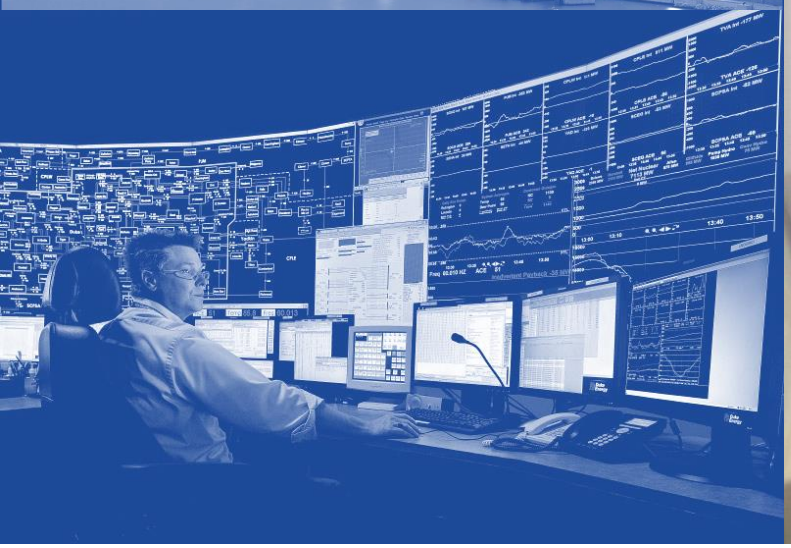
Deep Learning for Sequential Data

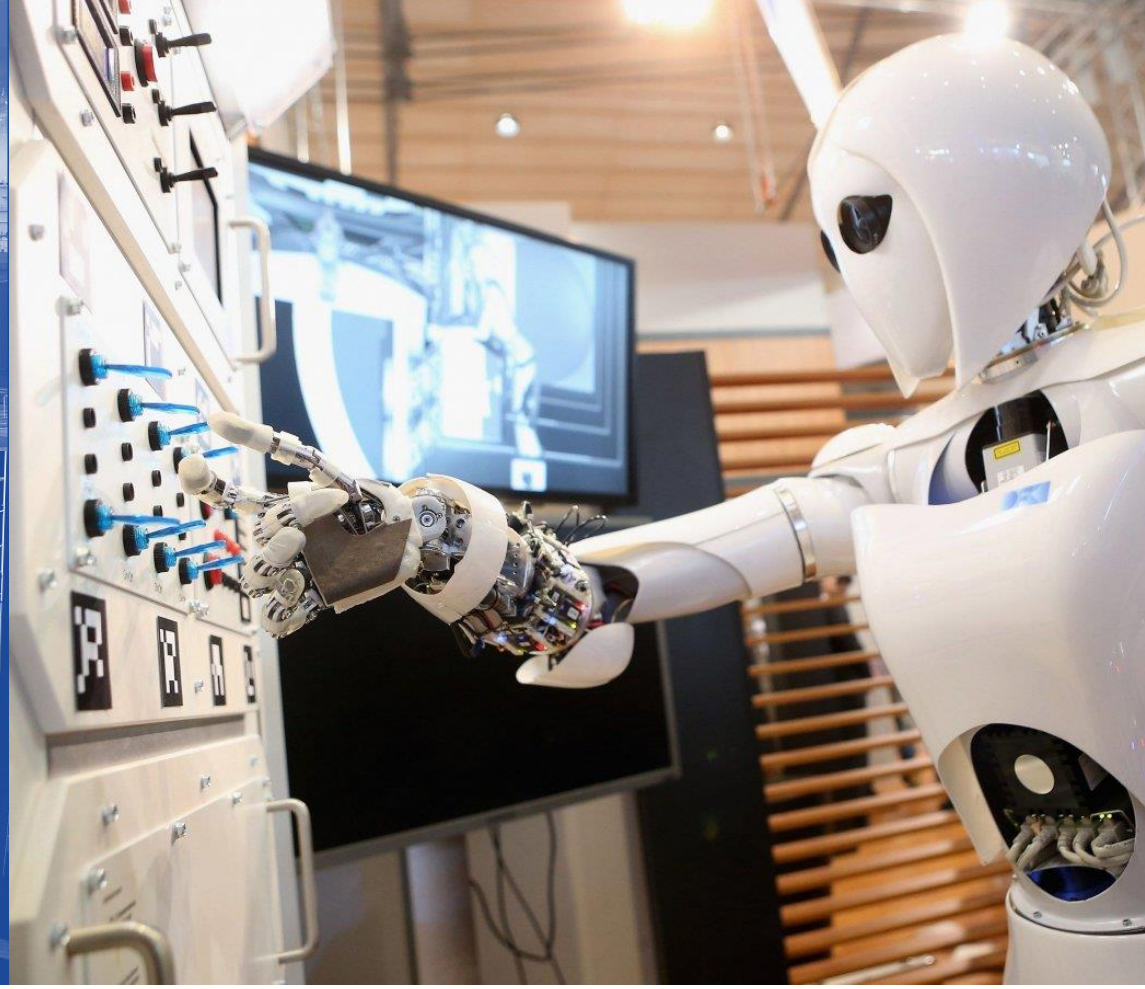
UNIST

Jaesik Choi

http://sail.unist.ac.kr/tutorials

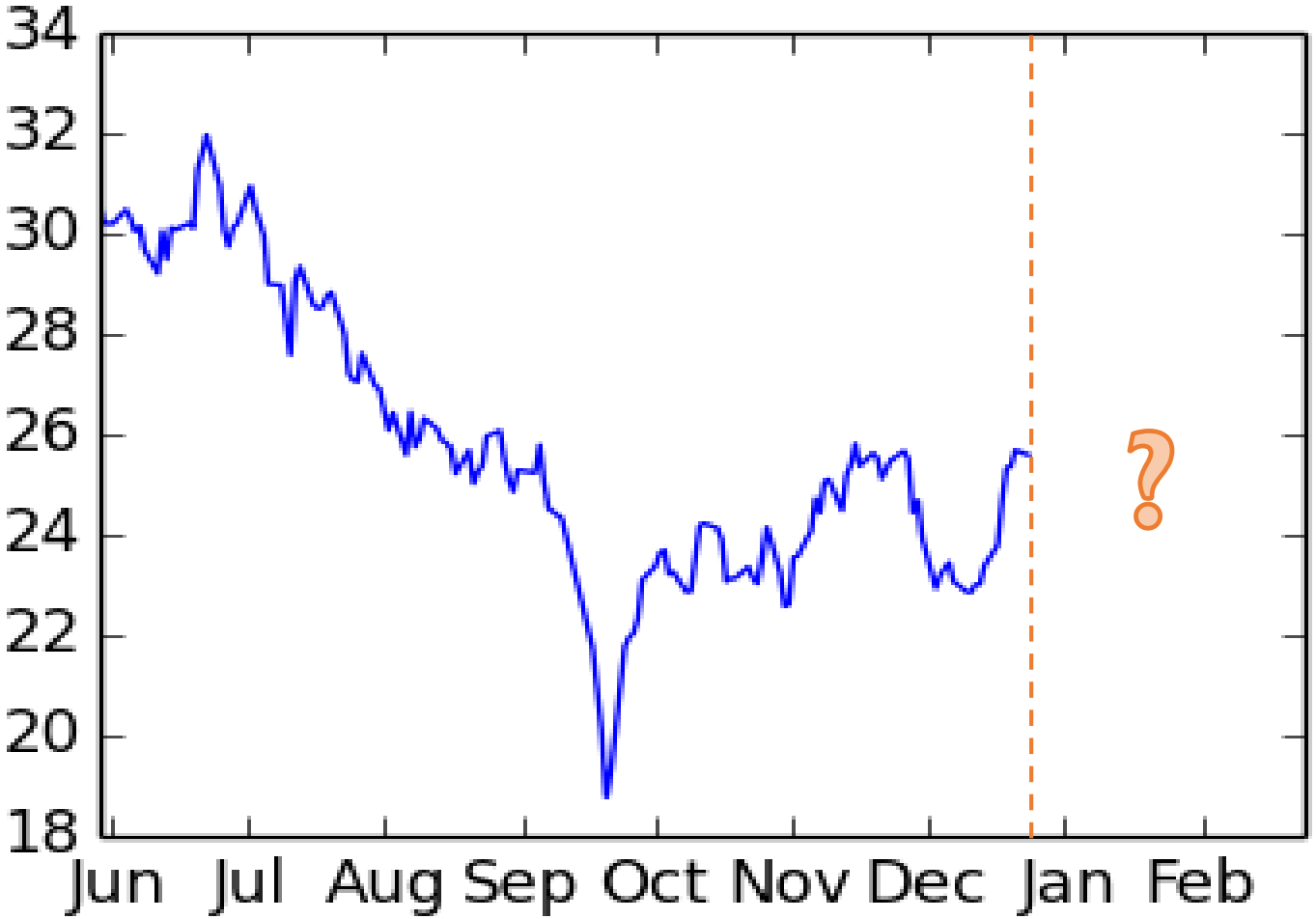
* Some slides courtesy of Prof. Injung Kim



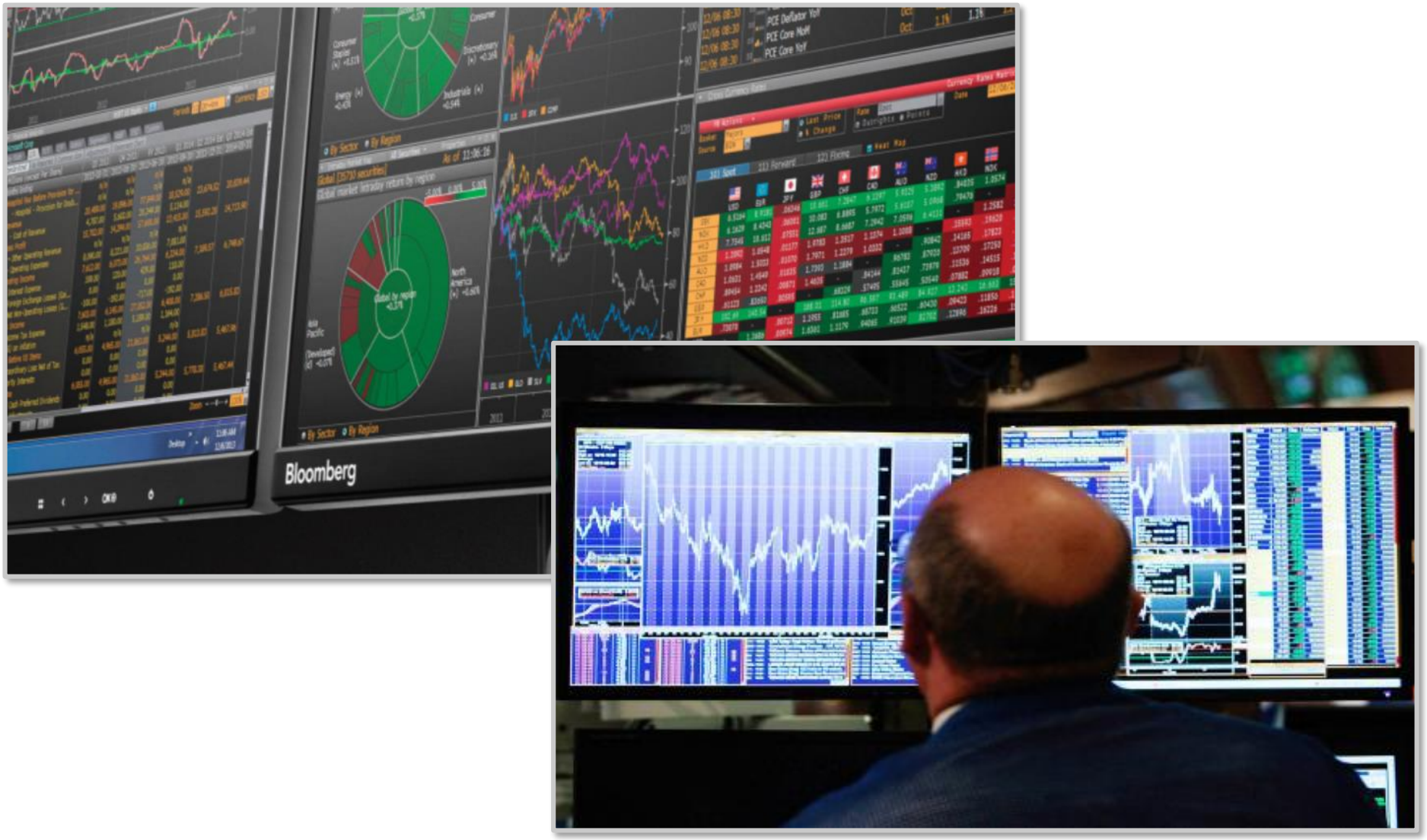


Sequential Data

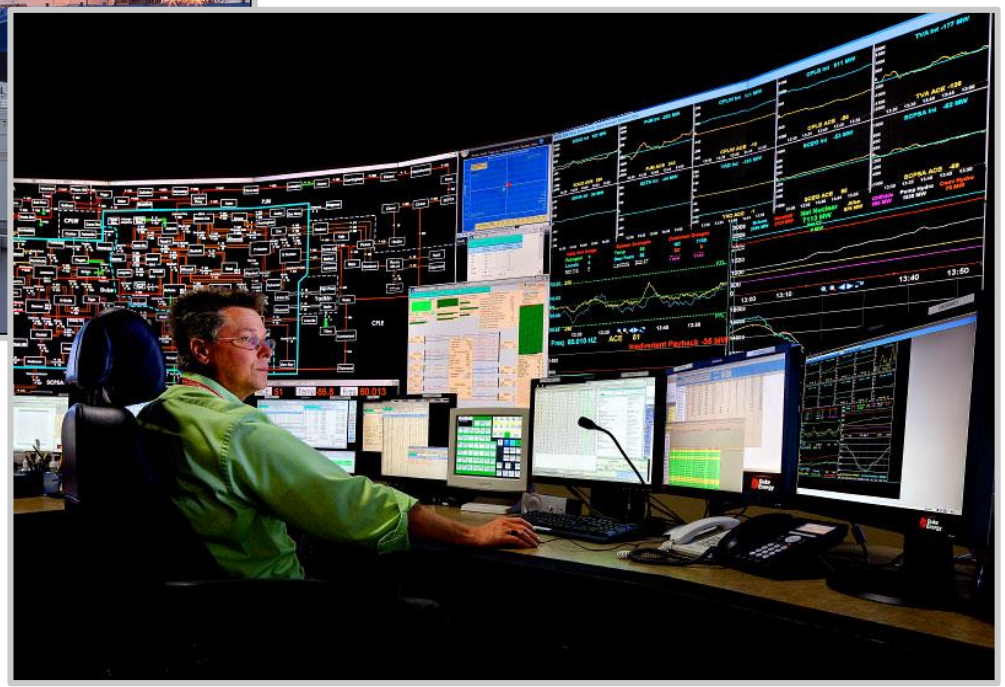
(Time Series Data)



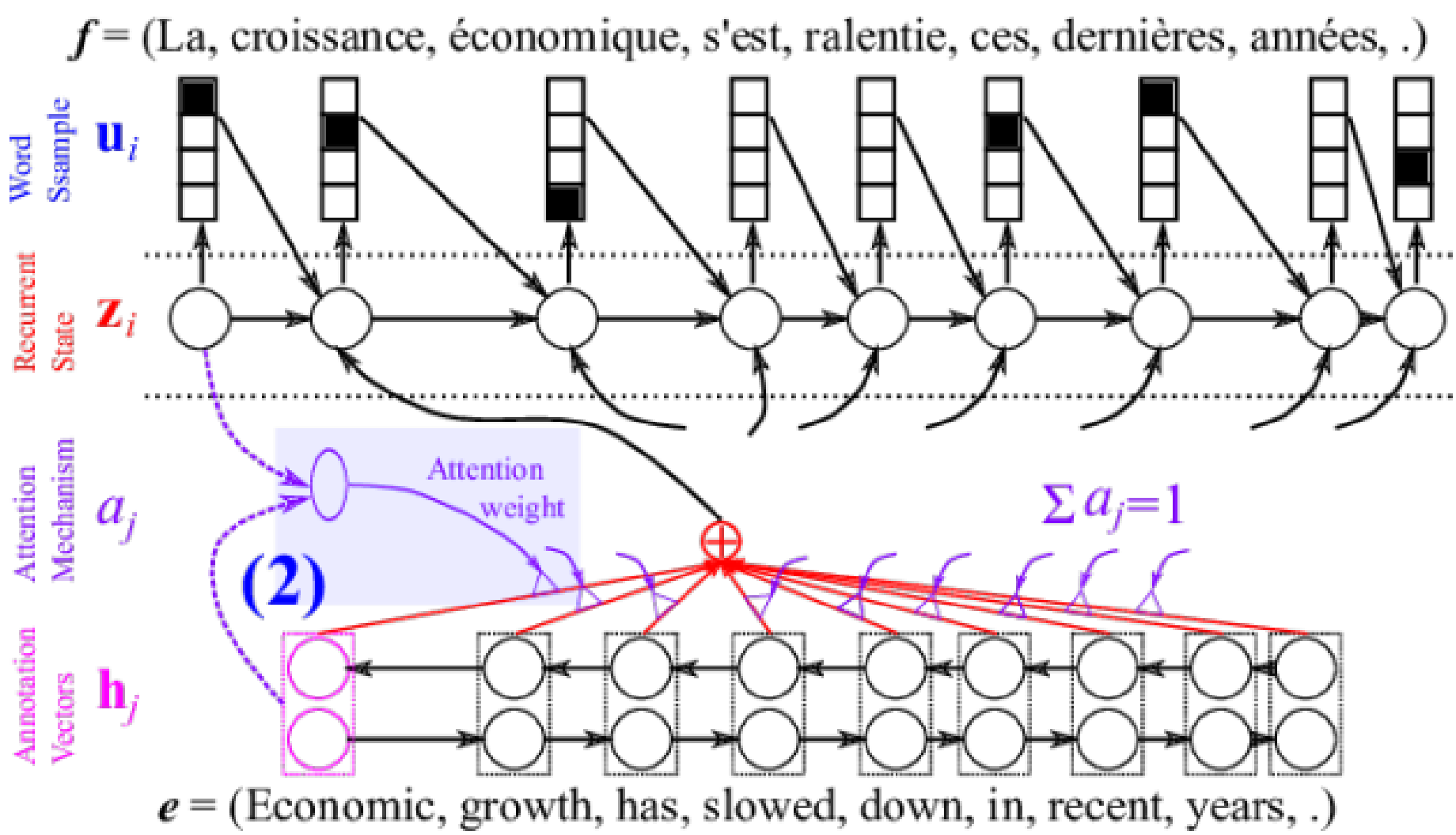
Prediction with Time Series Data



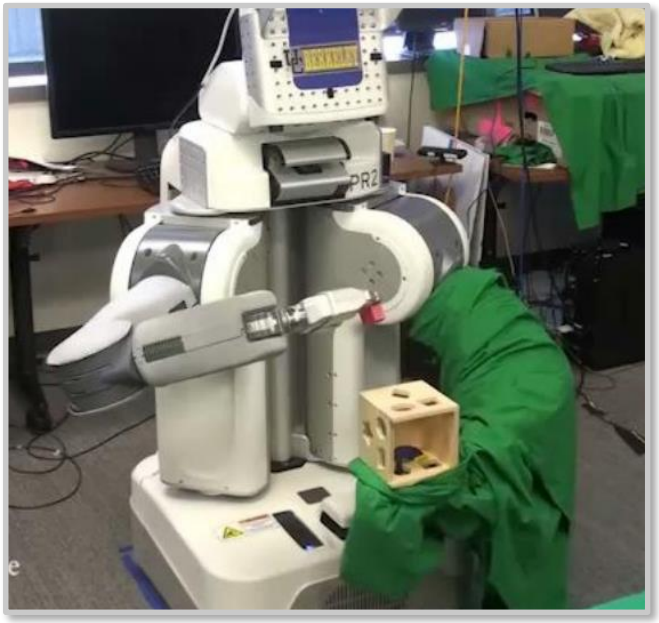
Financial Time Series Data

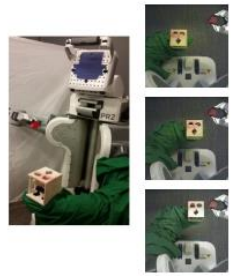


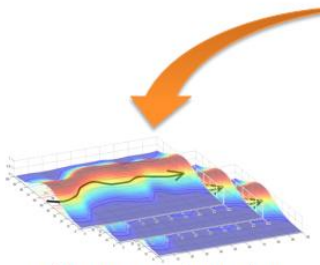
Industrial Time Series Data

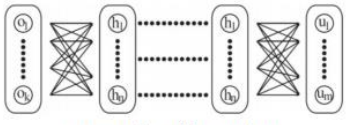


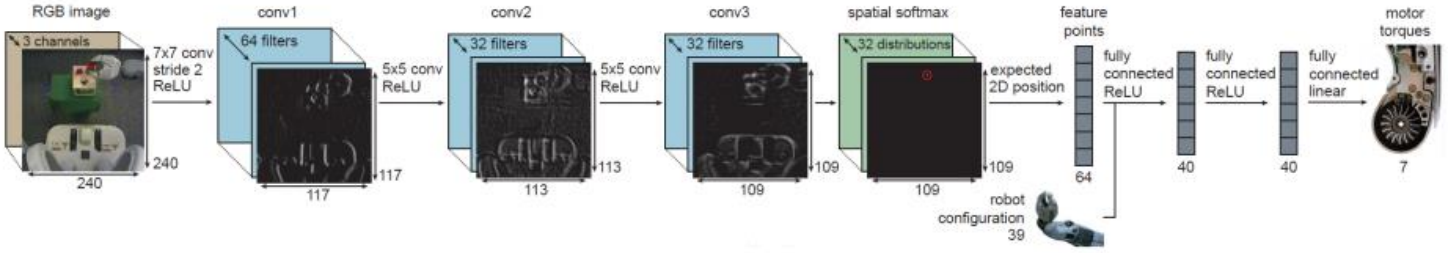
Machine Translation











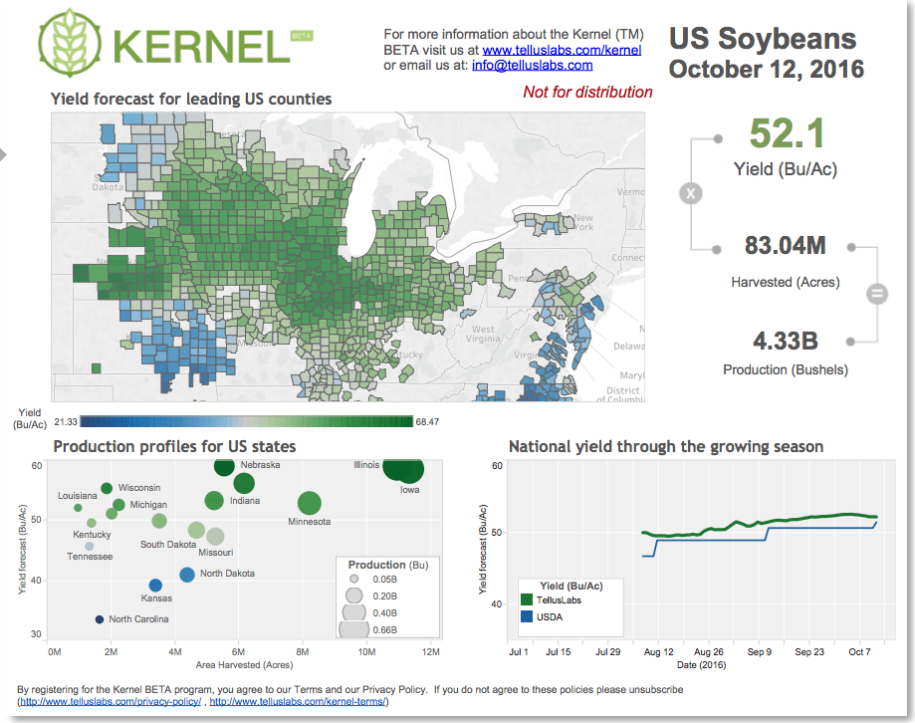
Robot Task (Manipulation) Learning (Sergey Levine, 2015)

Predictive Analysis

https://www.youtube.com/watch?v=0BAFOJbo4W4

위성사진

인공지능 기반 콩 생산량 예측(시/군구 단위)
[미국 USDA: 예측 주단위]



콩 생산량 예측 (Bu/Ac)

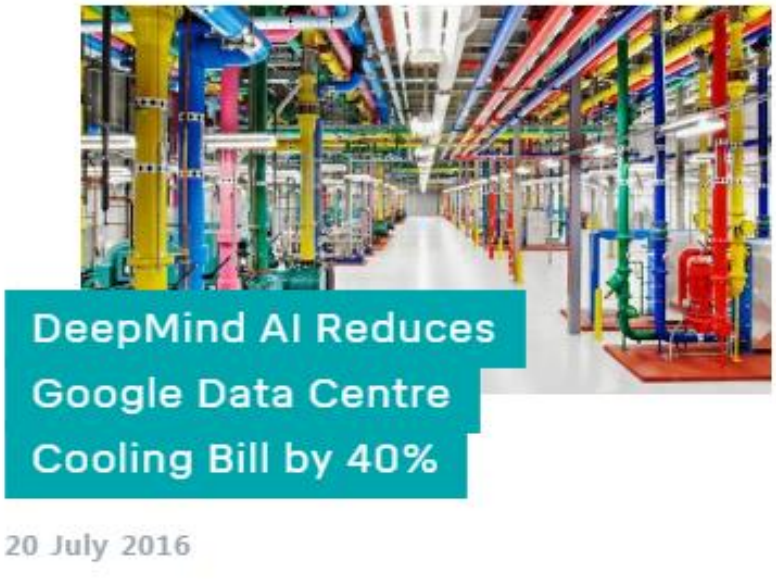
	2016/08	2016/09	2016/10
KERNEL	50 >	51.0	52.1
USDA	48.9	50.6	51.4

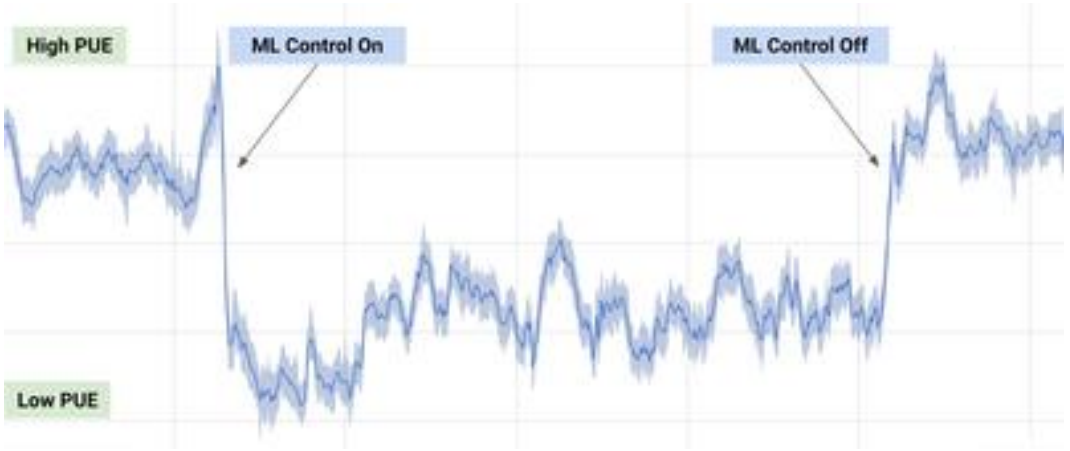
http://www.telluslabs.com/2016/10/12/telluslabs-forecasts-lead-usda-reports-corn-soy/

위성사진 기반 콩 생산량 예측(미국)

Google AI for Controlling Data Centers

https://www.youtube.com/watch?v=jVo6toAnE-s





Google says that it emits 1.5m tonnes of carbon annually but claims that its data centres consume 50% less energy than the industry average.

DeepMind AI reduces Google Data Center Bill (탄소배출 저감)

https://environment.google/approach/

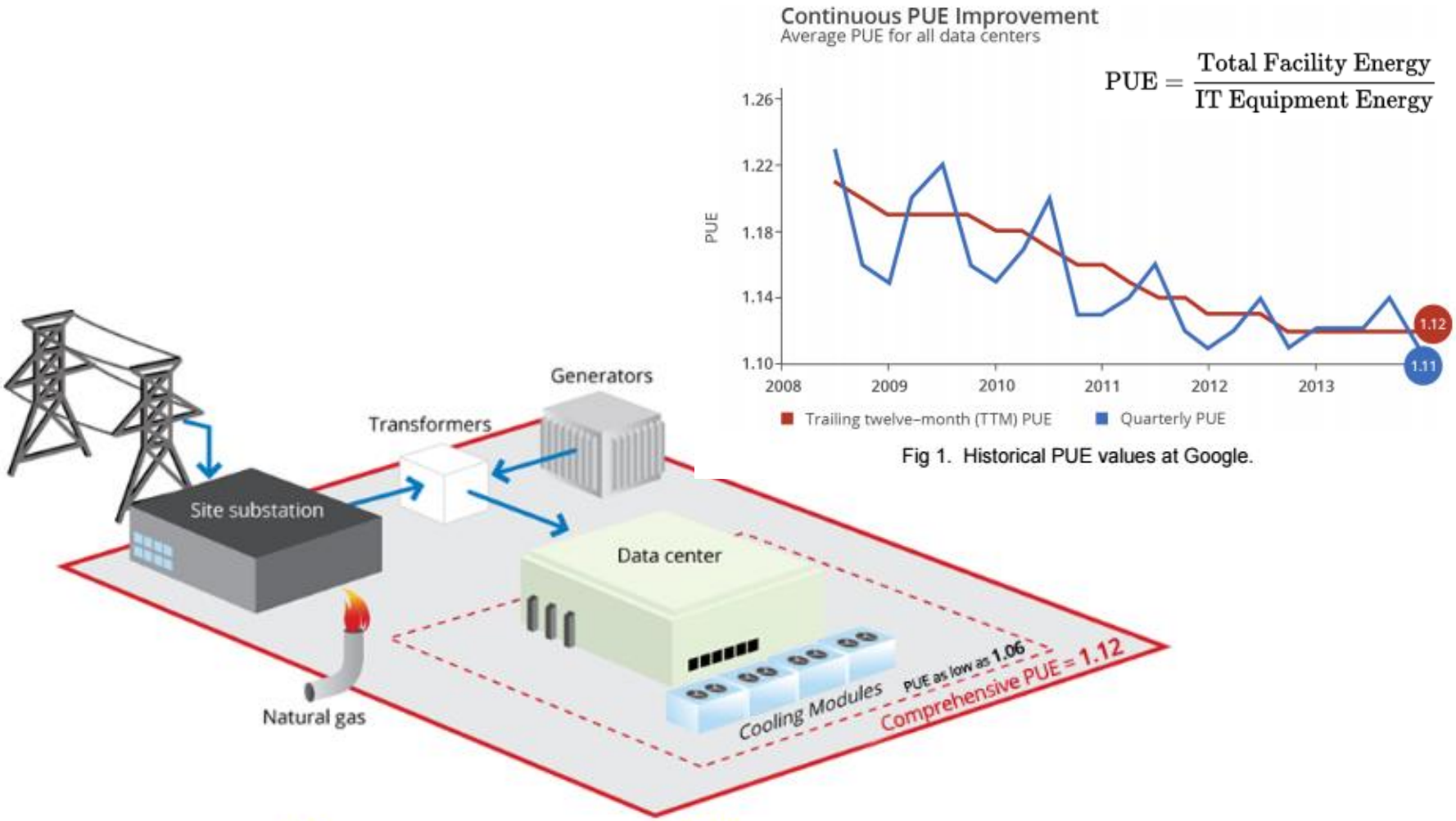


Fig 1. Historical PUE values at Google.

Figure 1: Google data center PUE measurement boundaries. The average PUE for all Google data centers is 1.12, although we could boast a PUE as low as 1.06 when using narrower boundaries.

DeepMind AI reduces Google Data Center Bill

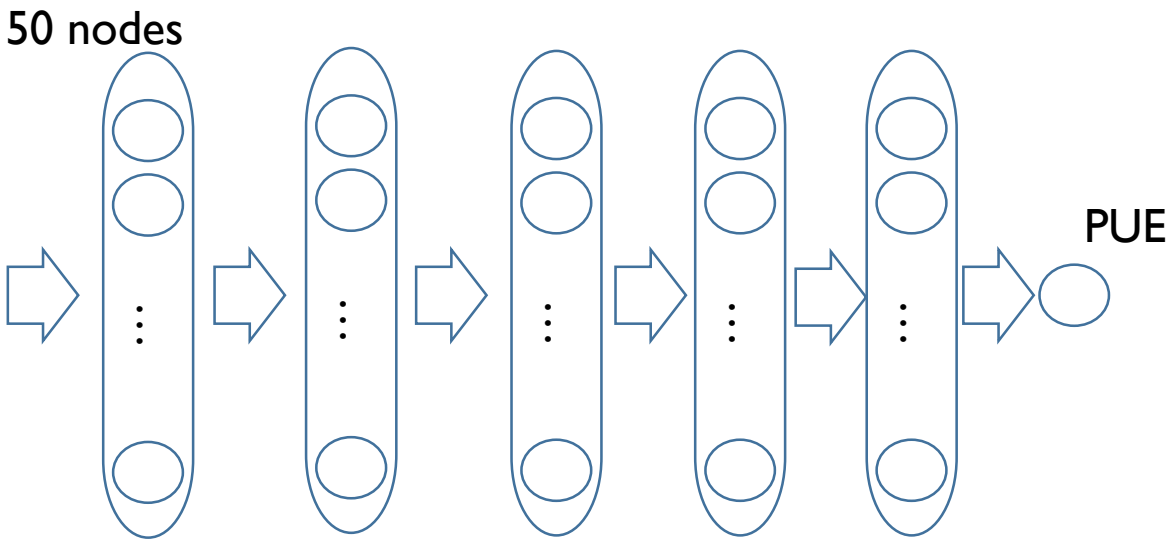
http://www.google.com/about/datacenters/efficiency/internal/assets/machine-learning-applicationsfor-datacenter-optimization-finalv2.pdf

Machine Learning Applications for Data Center Optimization

Jim Gao, Google

The neural network features are listed as follows:

1. Total server IT load [kW]
2. Total Campus Core Network Room (CCNR) IT load [kW]
3. Total number of process water pumps (PWP) running
4. Mean PWP variable frequency drive (VFD) speed [%]
5. Total number of condenser water pumps (CWP) running
6. Mean CWP variable frequency drive (VFD) speed [%]
7. Total number of cooling towers running
8. Mean cooling tower leaving water temperature (LWT) setpoint [F]
9. Total number of chillers running
10. Total number of drycoolers running
11. Total number of chilled water injection pumps running
12. Mean chilled water injection pump setpoint temperature [F]
13. Mean heat exchanger approach temperature [F]
14. Outside air wet bulb (WB) temperature [F]
15. Outside air dry bulb (DB) temperature [F]
16. Outside air enthalpy [kJ/kg]
17. Outside air relative humidity (RH) [%]
18. Outdoor wind speed [mph]
19. Outdoor wind direction [deg]



DeepMind AI reduces Google Data Center Bill

http://www.google.com/about/datacenters/efficiency/internal/assets/machine-learning-applicationsfor-datacenter-optimization-finalv2.pdf

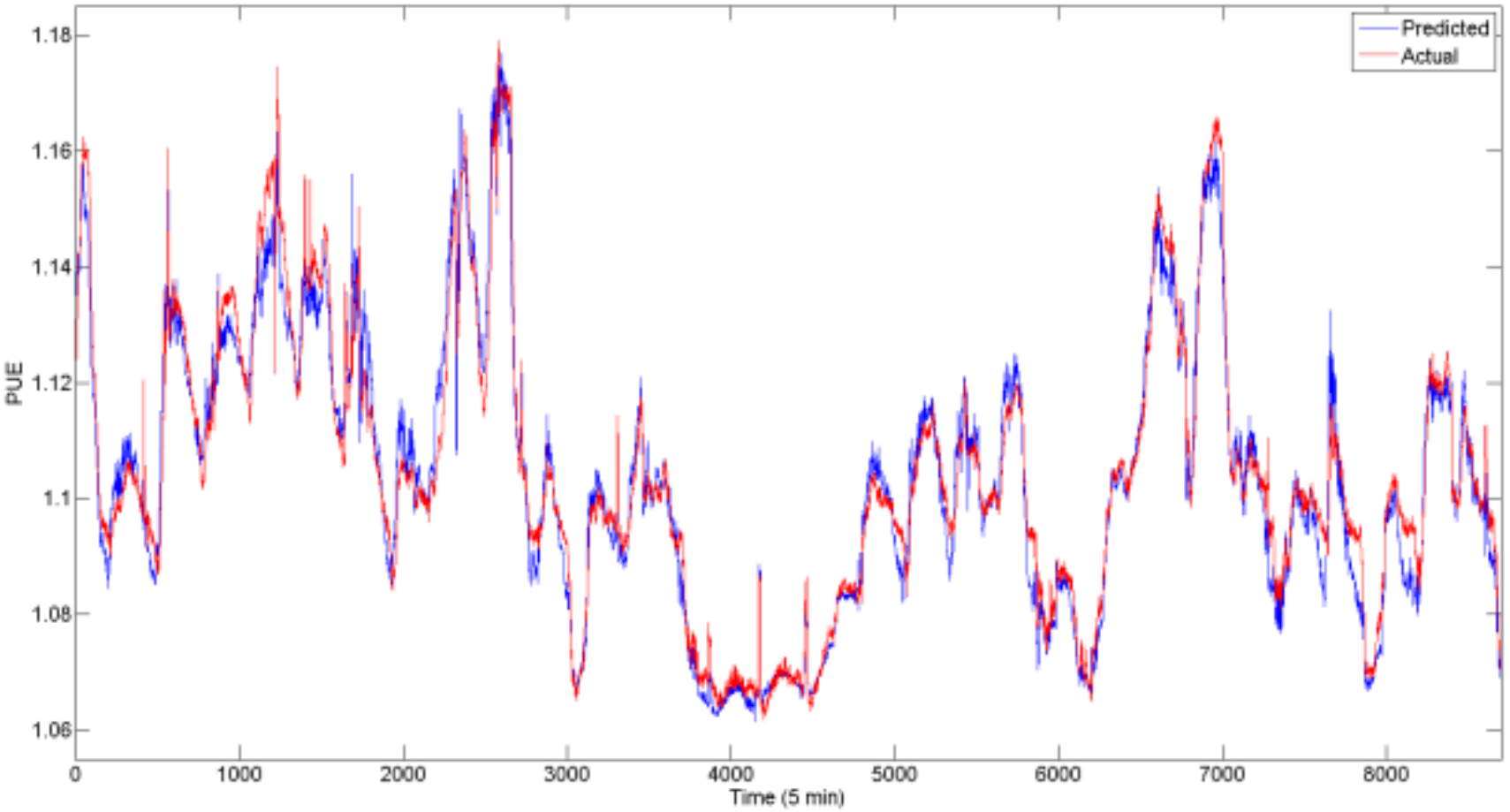
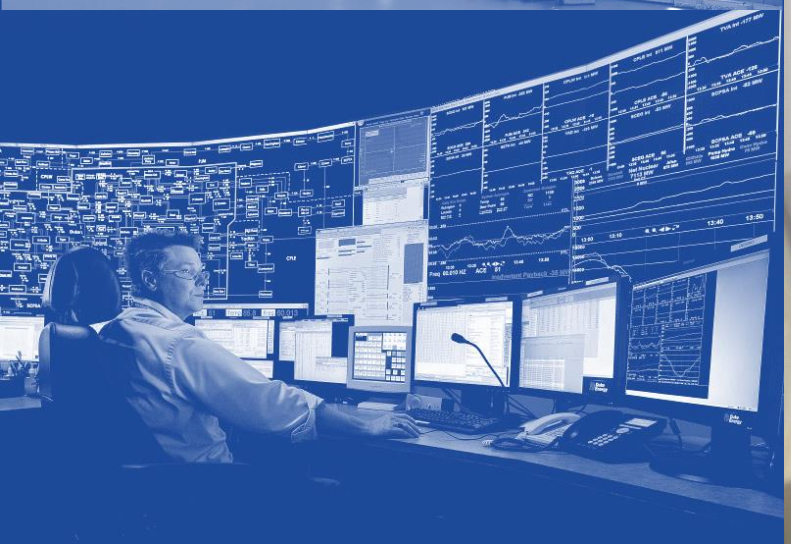
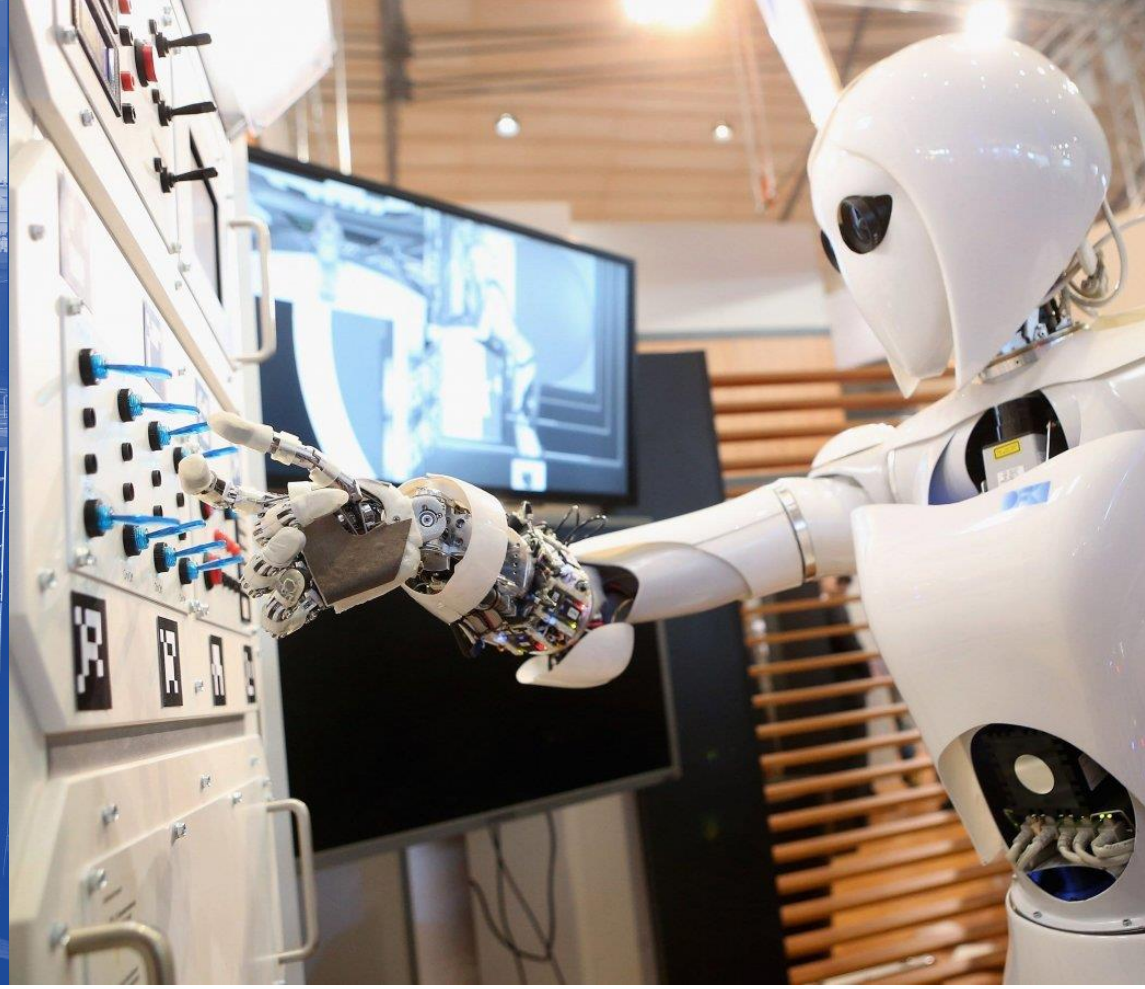


Fig. 3 Predicted vs actual PUE values at a major DC.

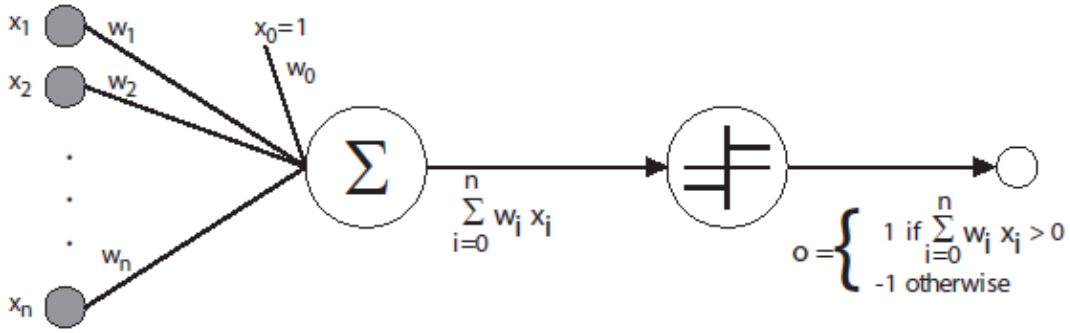
DeepMind AI reduces Google Data Center Bill

http://www.google.com/about/datacenters/efficiency/internal/assets/machine-learning-applicationsfor-datacenter-optimization-finalv2.pdf





What is Deep Learning?

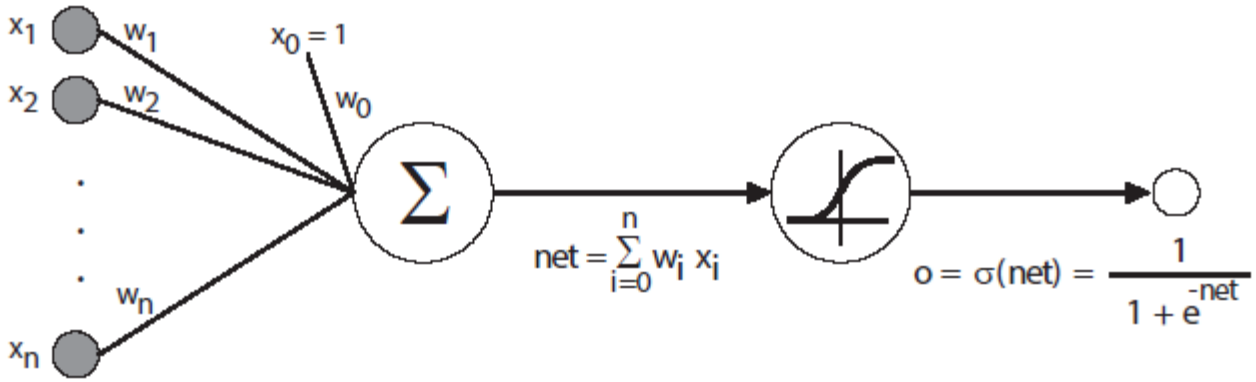


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Examples

- Given $(w_0, w_1, w_2, w_3) = (1, 1, 1, 0)$, what is $o(1, -3, 2) =$?

  (x_1, x_2, x_3)=(1, -3, 2)
  $w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 1 + 1 \cdot 1 + 1 \cdot (-3) + 0 \cdot 2 =$ -1
  $\rightarrow o(1, -3, 2) = -1$

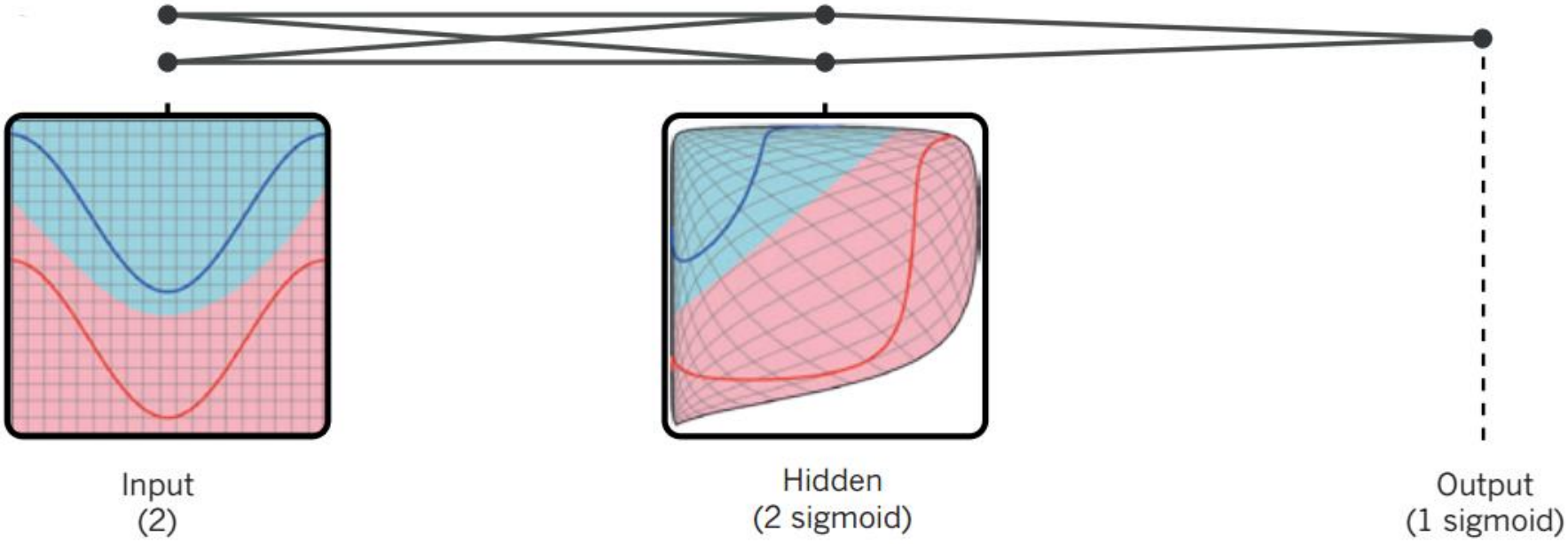


$\sigma(x)$ is the sigmoid function

$$\frac{1}{1+e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$

Nonlinear Transform

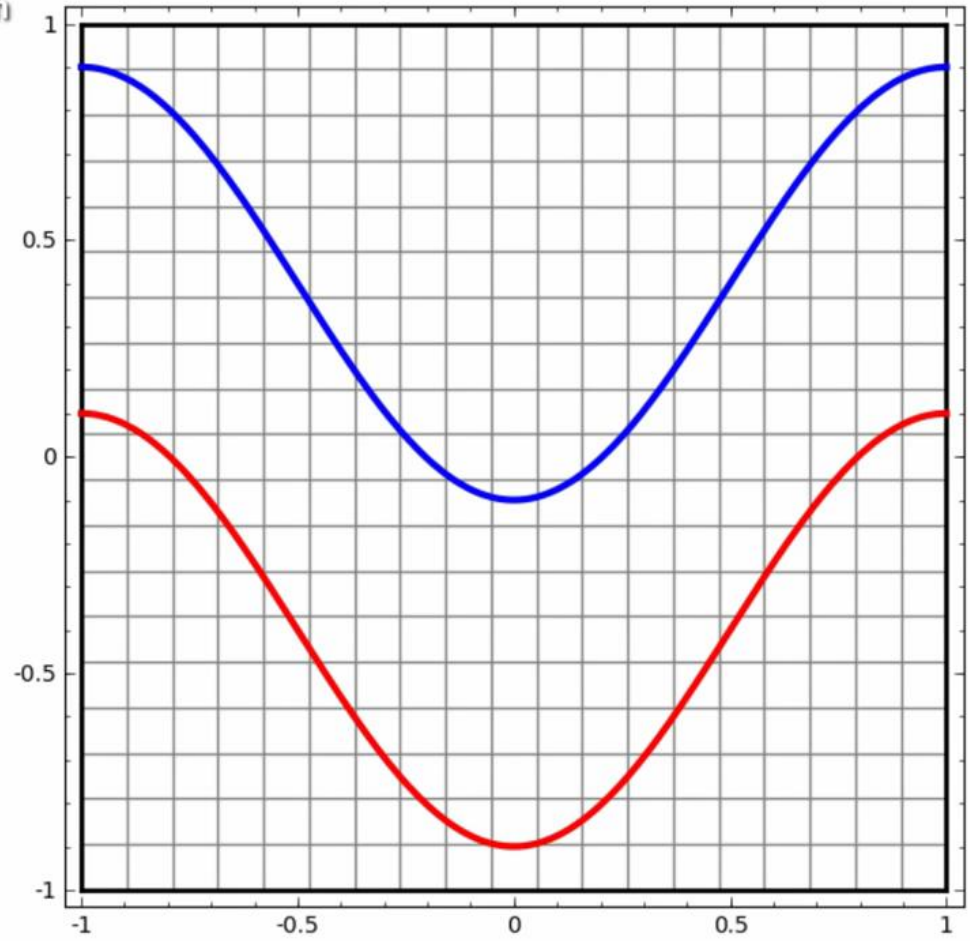


After a nonlinear transformation, red and blue are linear separable*

* Y. LeCun, Y. Bengio, G. Hinton (2015). Deep Learning. Nature 521, 436-444.

Linear Separable Classes in Multilayer Perceptron

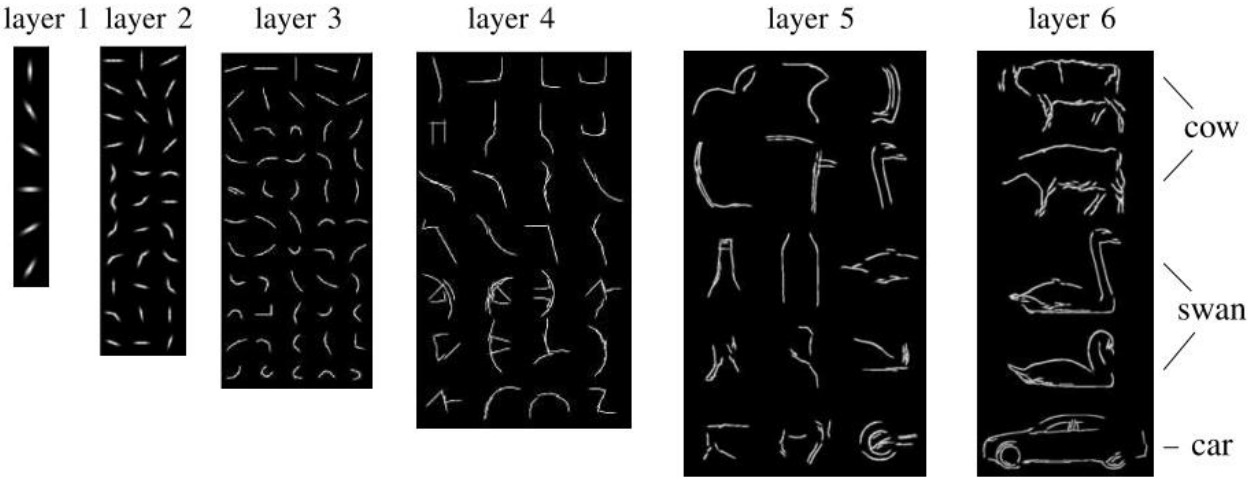
Source: Christopher Olah



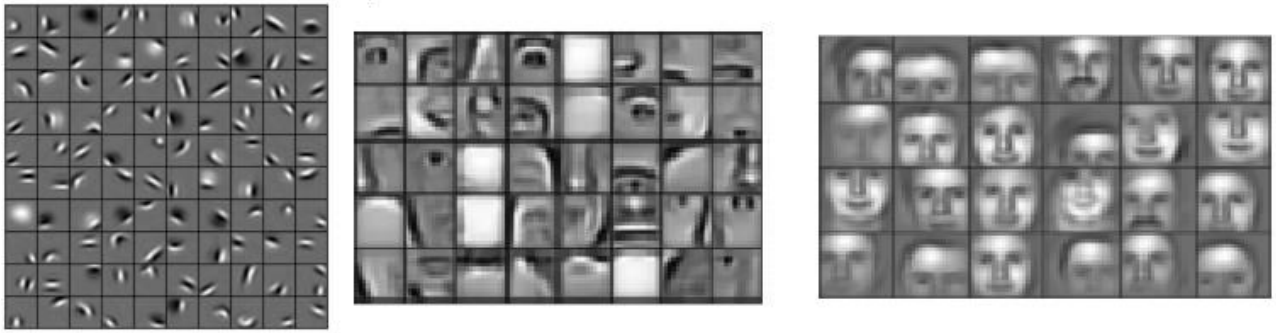
https://www.youtube.com/watch?v=He4t7Zekob0

How does the deep learning work

Recognizing drawing

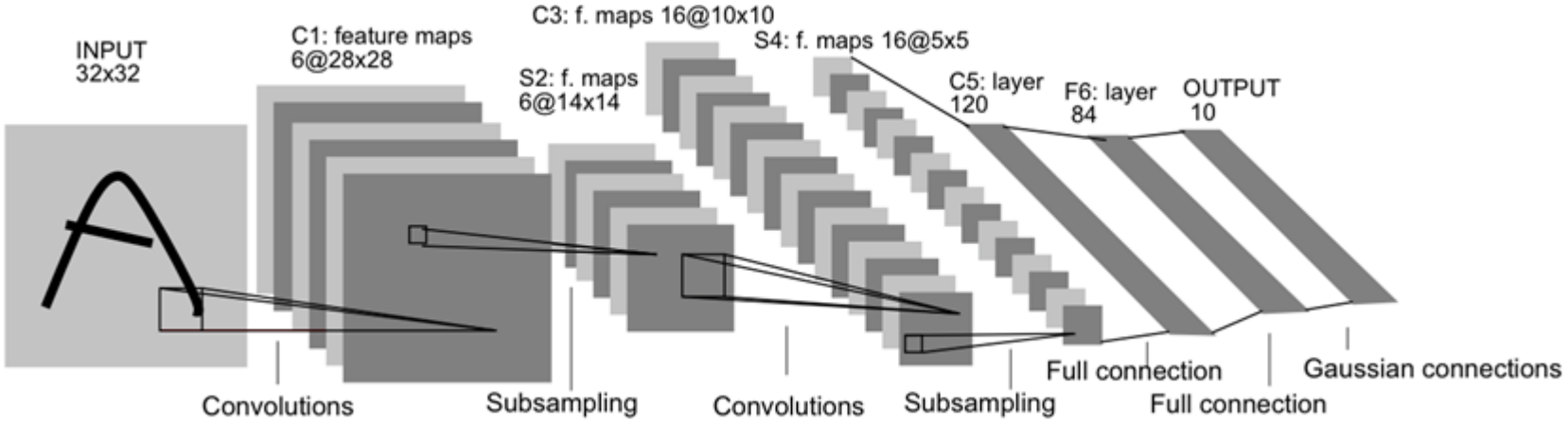


Recognizing human faces

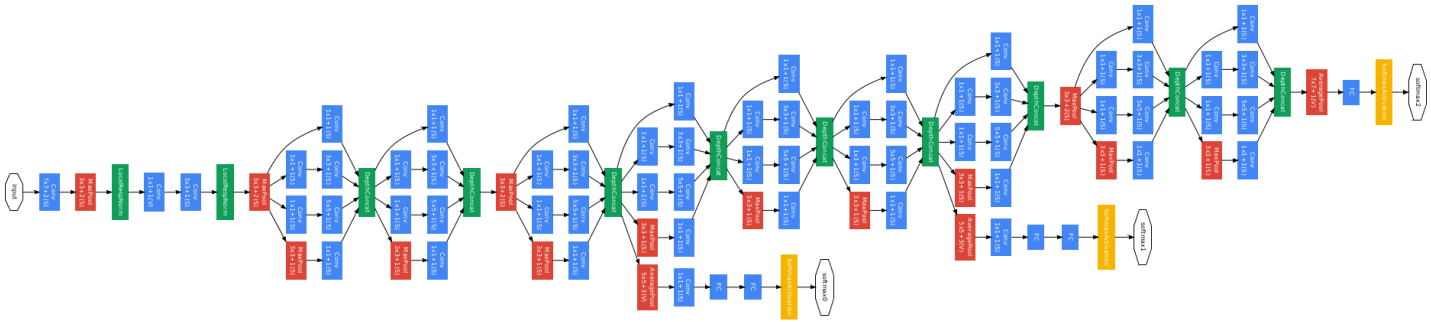


Learning Feature Hierarchy - Examples

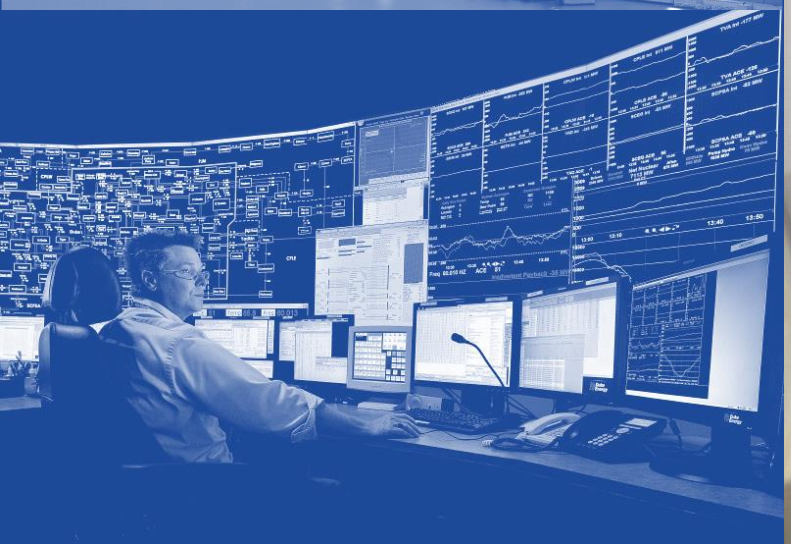
LeNet5: Recognizing digits using a neural network with 5 layers

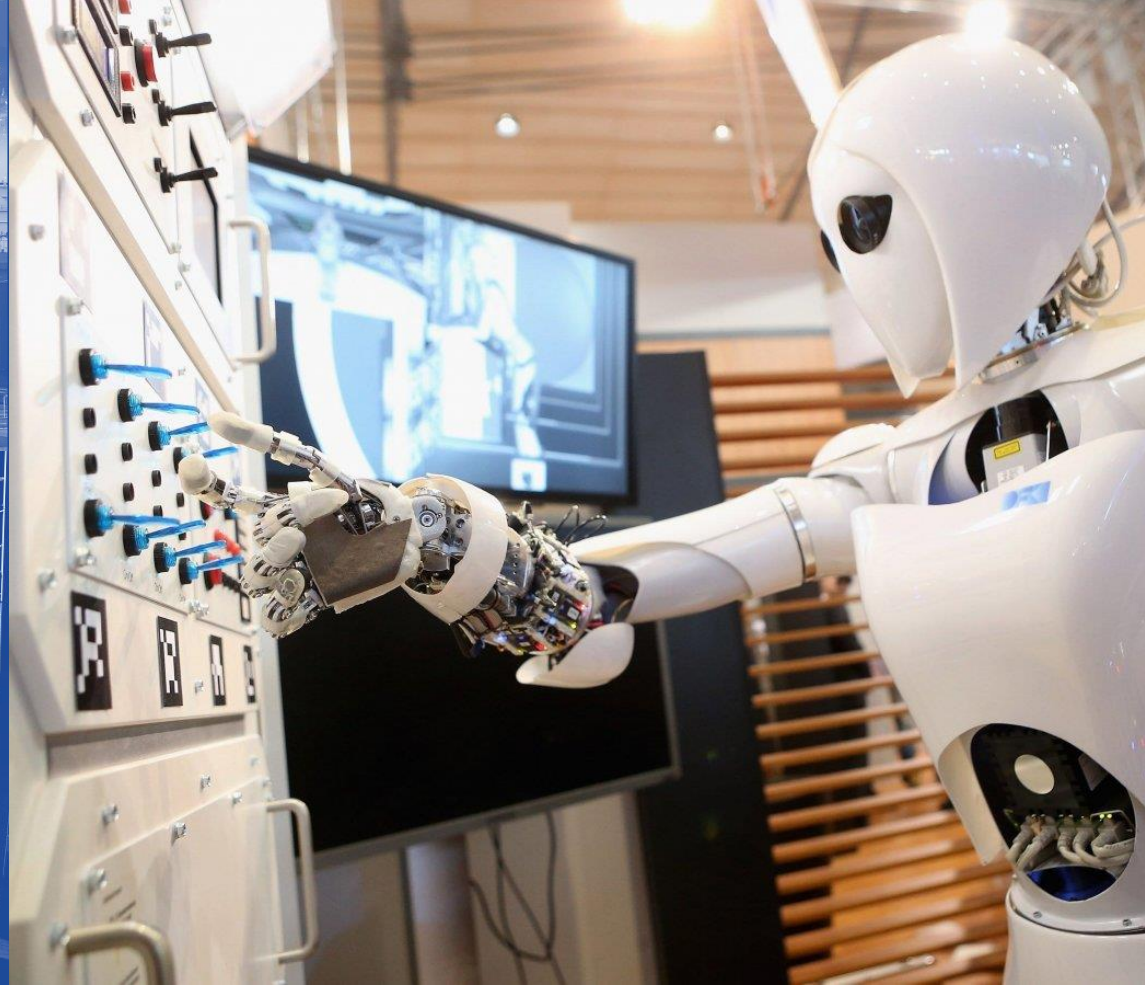


Recognizing human faces



LeNet 5 (1989) vs GoogLeNets (2014)

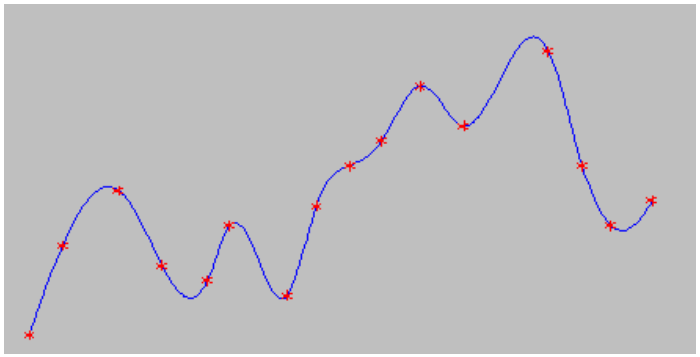


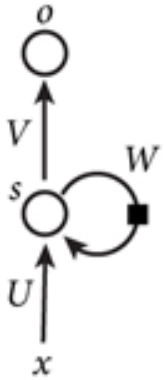


What are Recurrent Neural Networks?

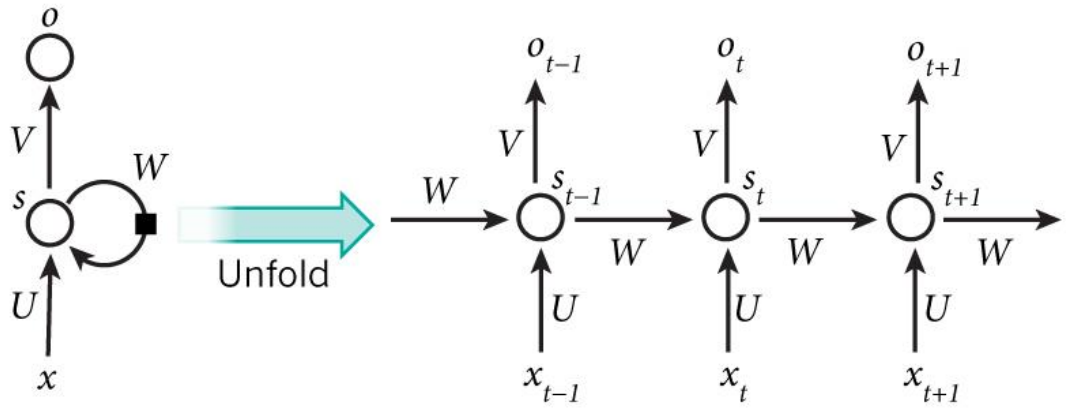
- Recurrent neural network is a neural network specialized for processing a sequence of values $x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}$.
  - Can scale to very long sequences

- Computational graph including cycle
  - Cycles represent the influence of the present value of a variable on its own value at a future time step

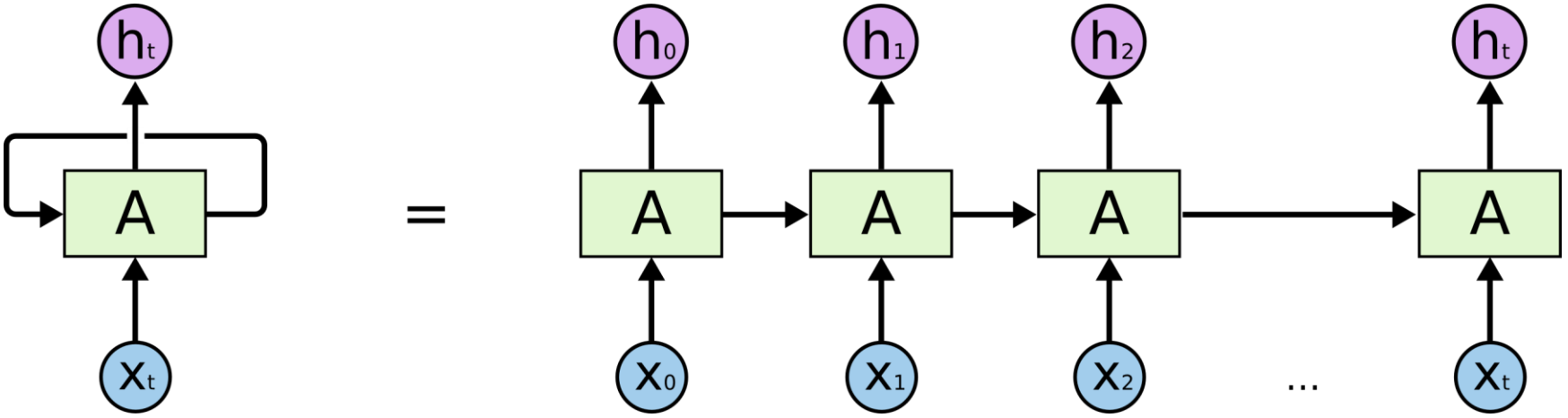




- Parameter sharing
  - Important when a specific piece of information can occur at multiple positions within the sequence.
    Ex) "I went to Nepal **in 2009**"
    "**In 2009**, I went to Nepal."

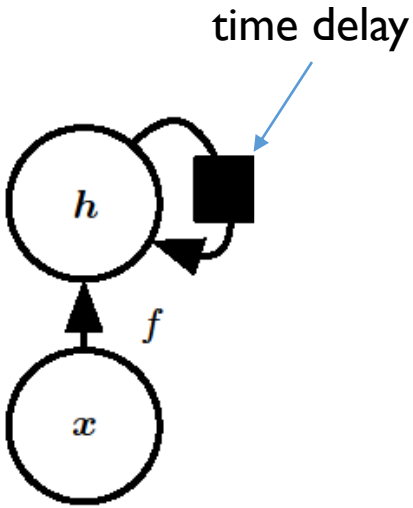


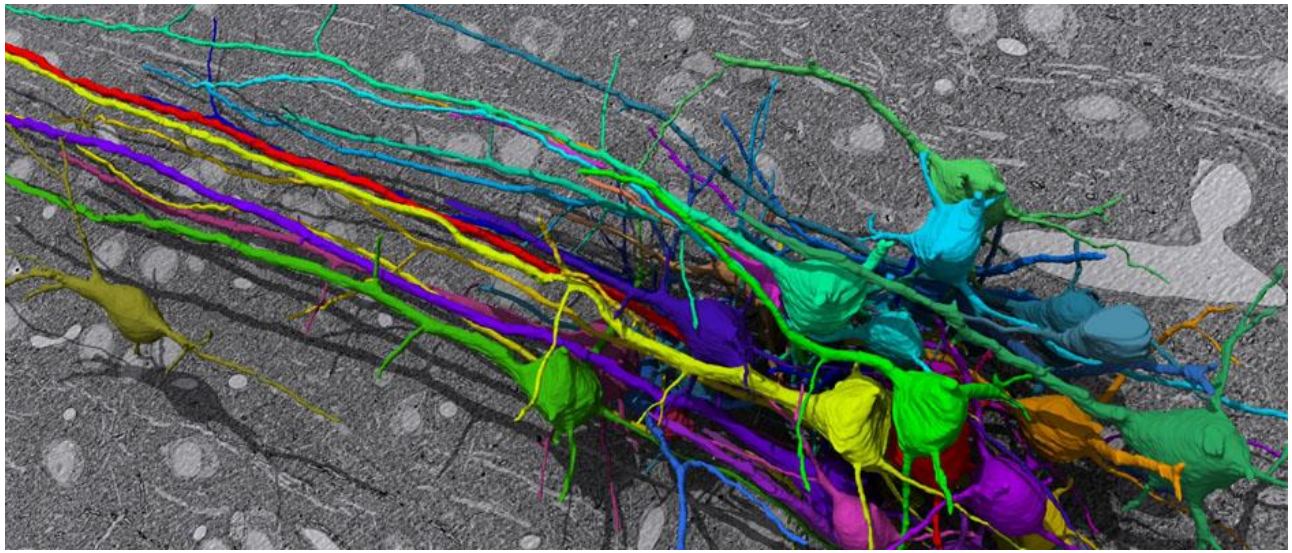
- Comparison
  - Traditional FFN: all rules **separately at each position**
  - CNN: shares the same weights **across position**
  - RNN: shares the same weights **across time steps**



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network (RNN)





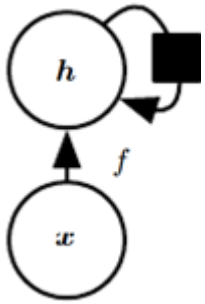
Recurrent Network - Circuit Graph vs Neuron

- A recurrent dynamic system with external input $x^{(t)}$
  - Input sequence: $x^{(1)}, x^{(2)}, \ldots, x^{(t)}, \ldots, x^{(\tau)}$

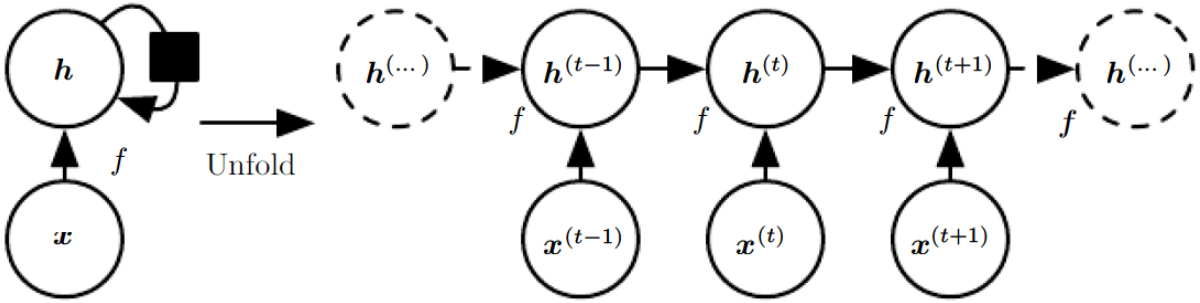
$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

  - When the state is hidden

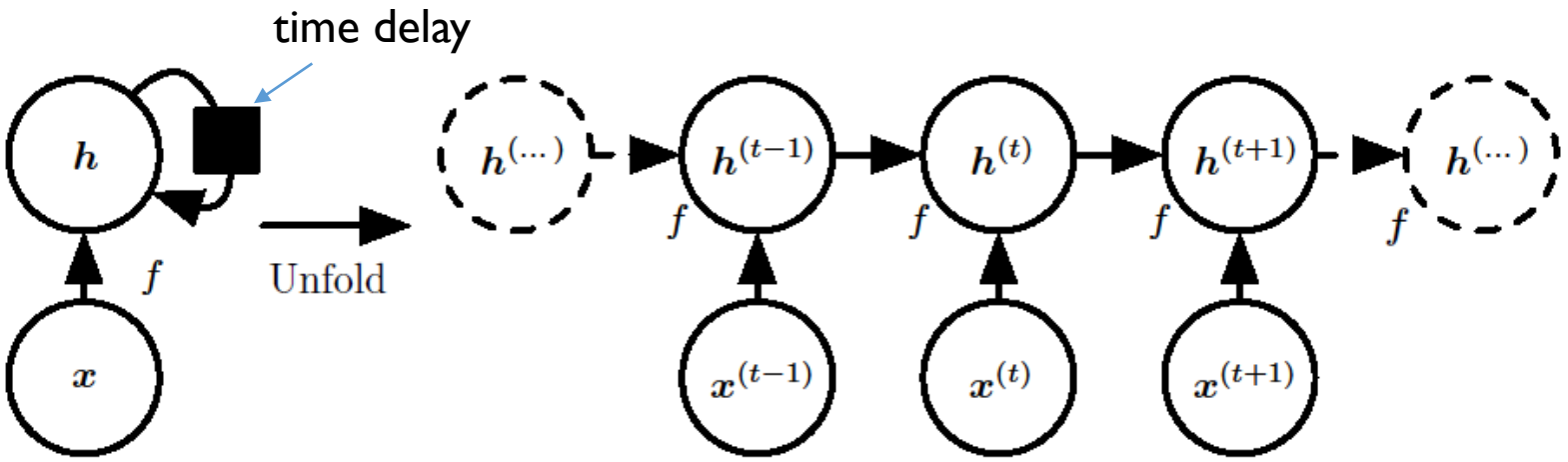
$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$



$h^{(t)}$: lossy summary of task-relevant aspects of the past sequence



Unfolding Computational Graphs



x: input
h: hidden unit
θ: parameter - parameter sharing (or parameter tying) over time t

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

Recurrent Network - Circuit Graph vs Unfold Computation Graph

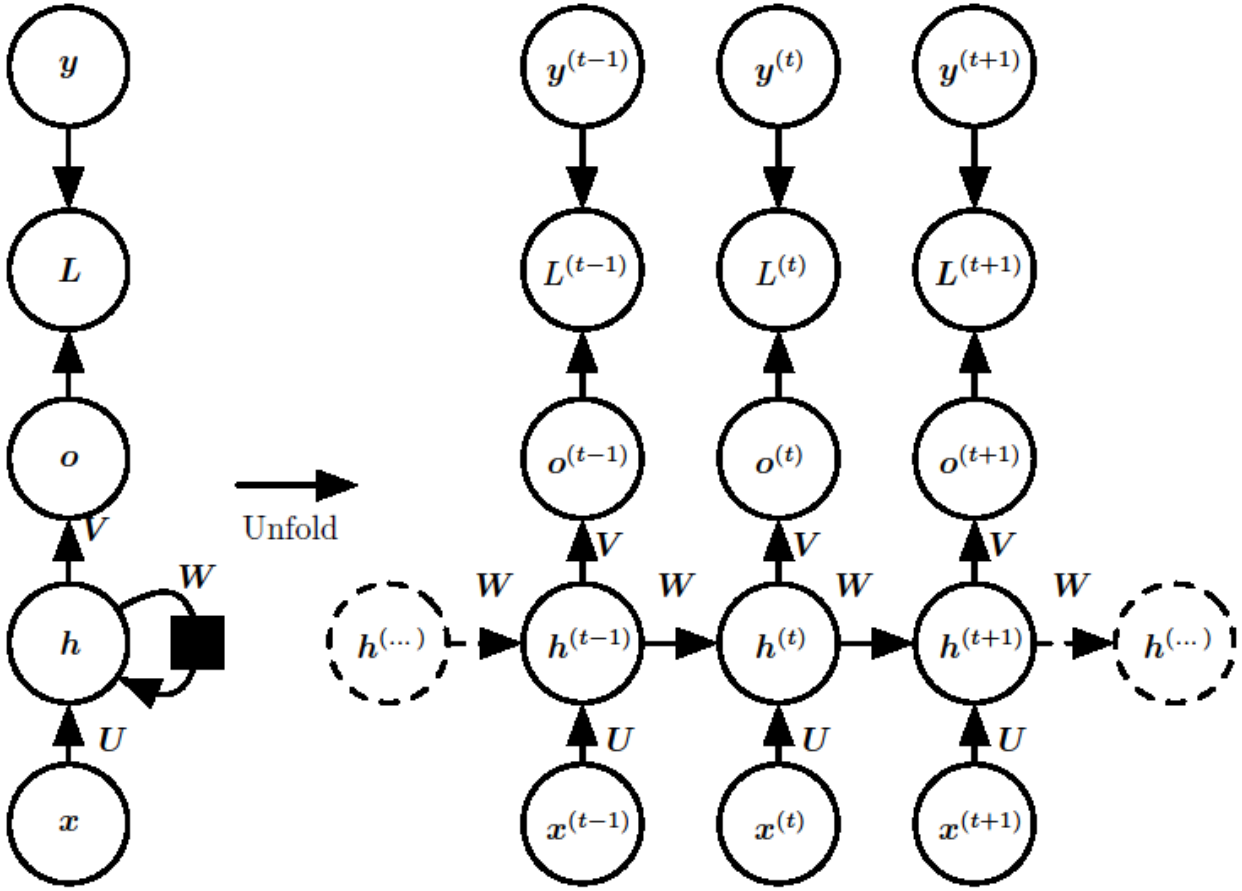
[Goodfellow et al., 2016] http://www.deeplearningbook.org

Advantages of Unfolding

- $g^{(t)}$ is factorized into repeated application of a function f.
  - $g^{(t)}$: separate model for all possible time steps
  - f: a single model that operates on all time steps and all sequence lengths
  - RNN learns f rather than $g^{(t)}$.
- Advantages
  - Regardless of the sequence length, the learned model always has the same input size.
    - Specified in terms of transition from one state to another state, rather than specified in terms of a variable-length history of states.
  - The same transition function f with the same parameters at every time step.

x: input
h: hidden unit
o: output
L: loss function
y: true label

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}
\end{aligned}
$$



Parameter Learning: Back-propagation through time (BPTT)
* Unable to be parallelized

Recurrent Neural Networks

[Goodfellow et al., 2016] http://www.deeplearningbook.org

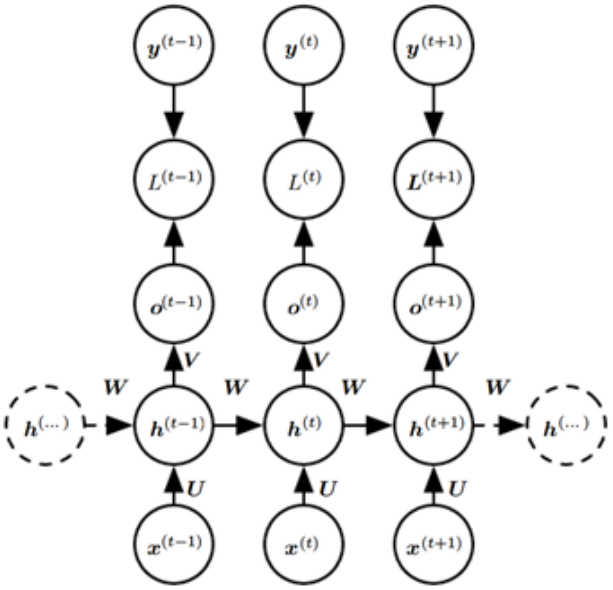
- Loss of RNN
  - $L^{(t)}$: loss at time t

    Ex) negative log-likelihood of true target $y^{(t)}$

$$-\log p_{model}(y^{(t)}|\{x^{(1)}, \dots, x^{(t)}\})$$

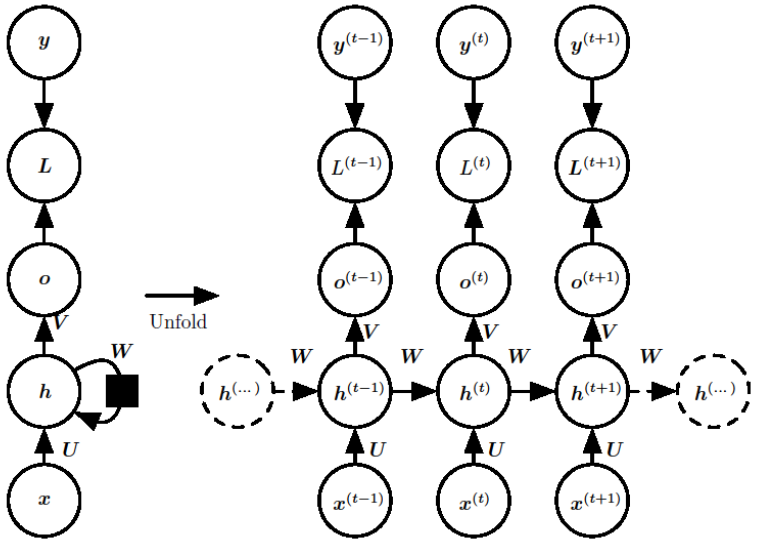
  - Total loss for a sequence

$$L\left(\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(\tau)}\}\right)$$
$$= \sum_t L^{(t)}$$
$$= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(t)}\}\right)$$



Loss Function for RNN

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}
\end{aligned}
$$



Remark: the recurrent neural network is universal in the sense that any function computable by a Turing machine can be computed by such a recurrent network of a finite size. (RNNs are Turing complete)

The output can be read from the RNN after a number of time steps that is asymptotically linear in the number of time steps used by the Turing machine and in the length of the input (Siegelmann and Sontag, 1991; Hyotyniemi, 1996)

A Turing machine = A finite state machine + An external tape

Recurrent Hidden Units

[Goodfellow et al., 2016] http://www.deeplearningbook.org

Remark: Computable functions can be implemented using a Turing machine

Following language L can be used to represent any function which is Turing computable.

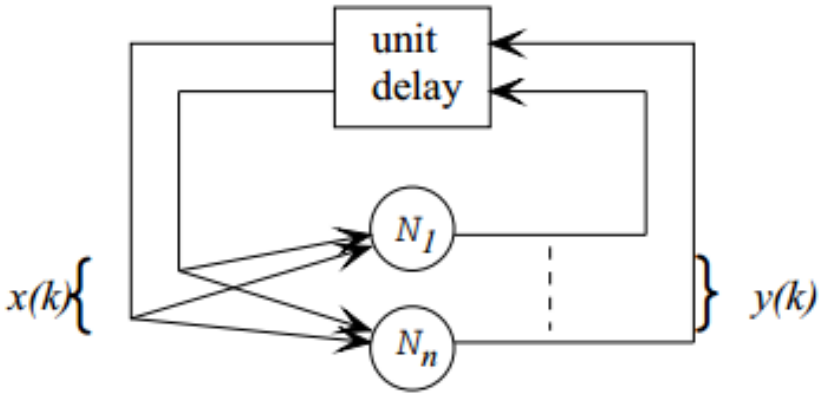
L is composed of the following basic operations

- No operation: $V \leftarrow V$
- Increment: $V \leftarrow V + 1$
- Decrement: $V \leftarrow \max\{0, V-1\}$
- Conditional branch: IF $V \neq 0$ GOTO j.

Recurrent neural network structure

$$y_q(k) = f\left(\sum_{p=1}^{n} w_{qp} x_p(k)\right),$$

$$f(x) = \begin{cases} x, & \text{if } x > 0, \text{ and} \\ 0, & \text{otherwise,} \end{cases}$$



Turing Recurrent Network

[Hyotyniemi 1996]

Given a program, make the following network

- For each variable V in the program, augment the network with the following link:

$$N_V \longrightarrow \boxed{1} \longrightarrow N_V$$

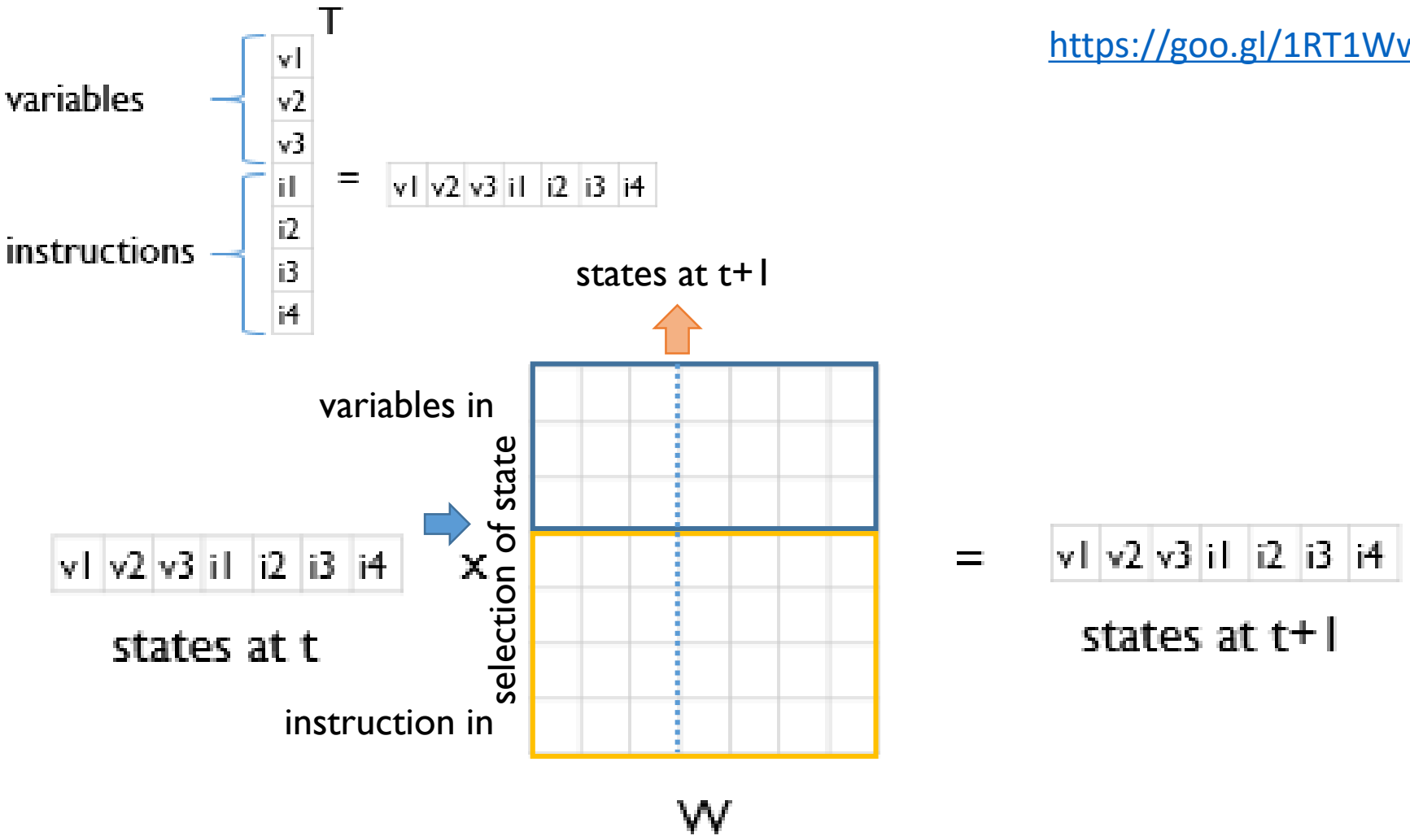
- If there is no operation $(V \leftarrow V)$ on row i of the program code then augment the network with the following link (assuming that node N_{i+1} exists):

$$N_i \longrightarrow \boxed{1} \longrightarrow N_{i+1}$$

Turing Recurrent Network

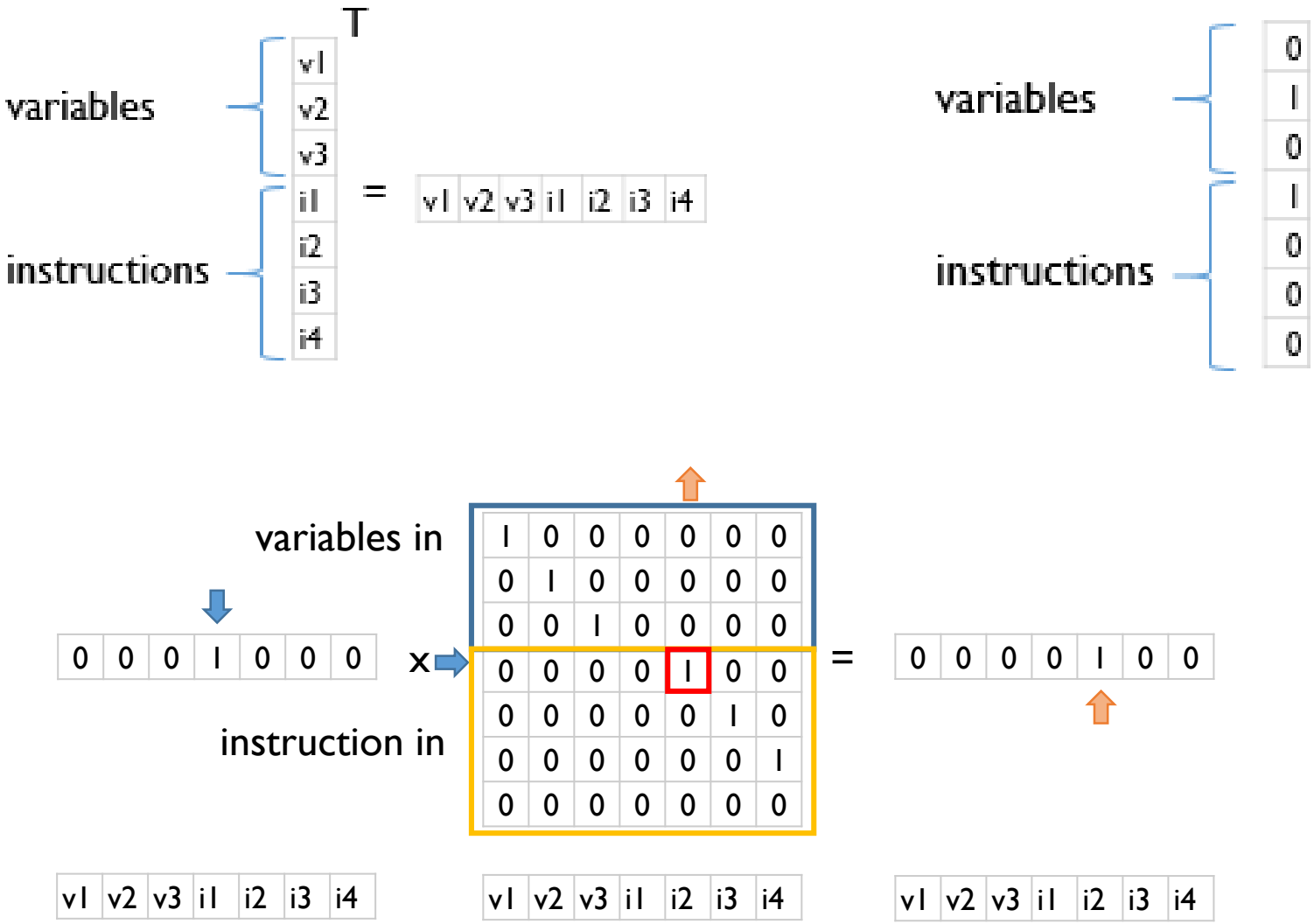
[Hyotyniemi 1996]

https://goo.gl/1RT1Ww



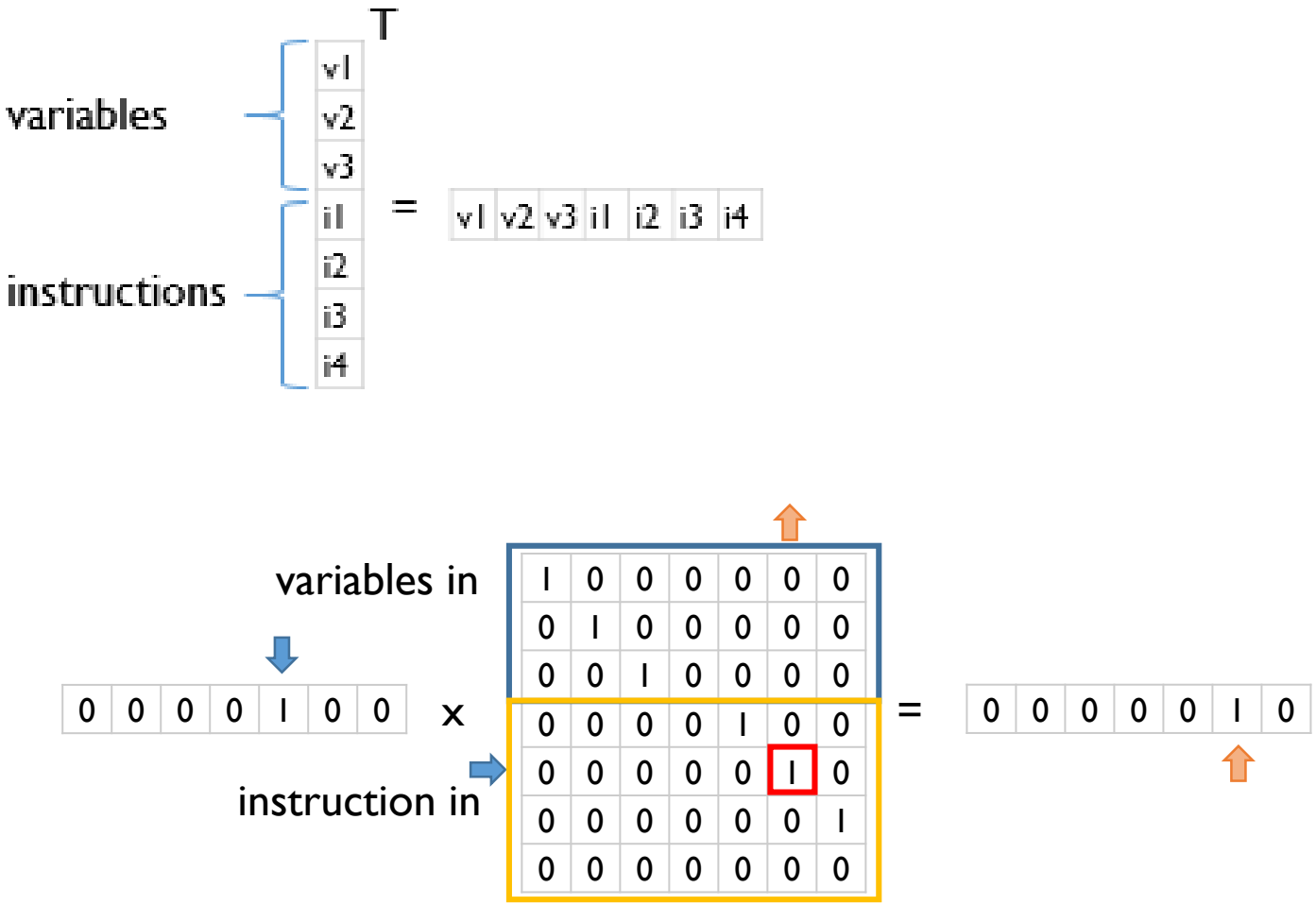
Turing Recurrent Network

[Hyotyniemi 1996]



Turing Recurrent Network

[Hyotyniemi 1996]



Turing Recurrent Network

[Hyotyniemi 1996]

Given a program, make the following network

- If there is an increment operation $(V \leftarrow V + 1)$ on row i then augment the network as follows:

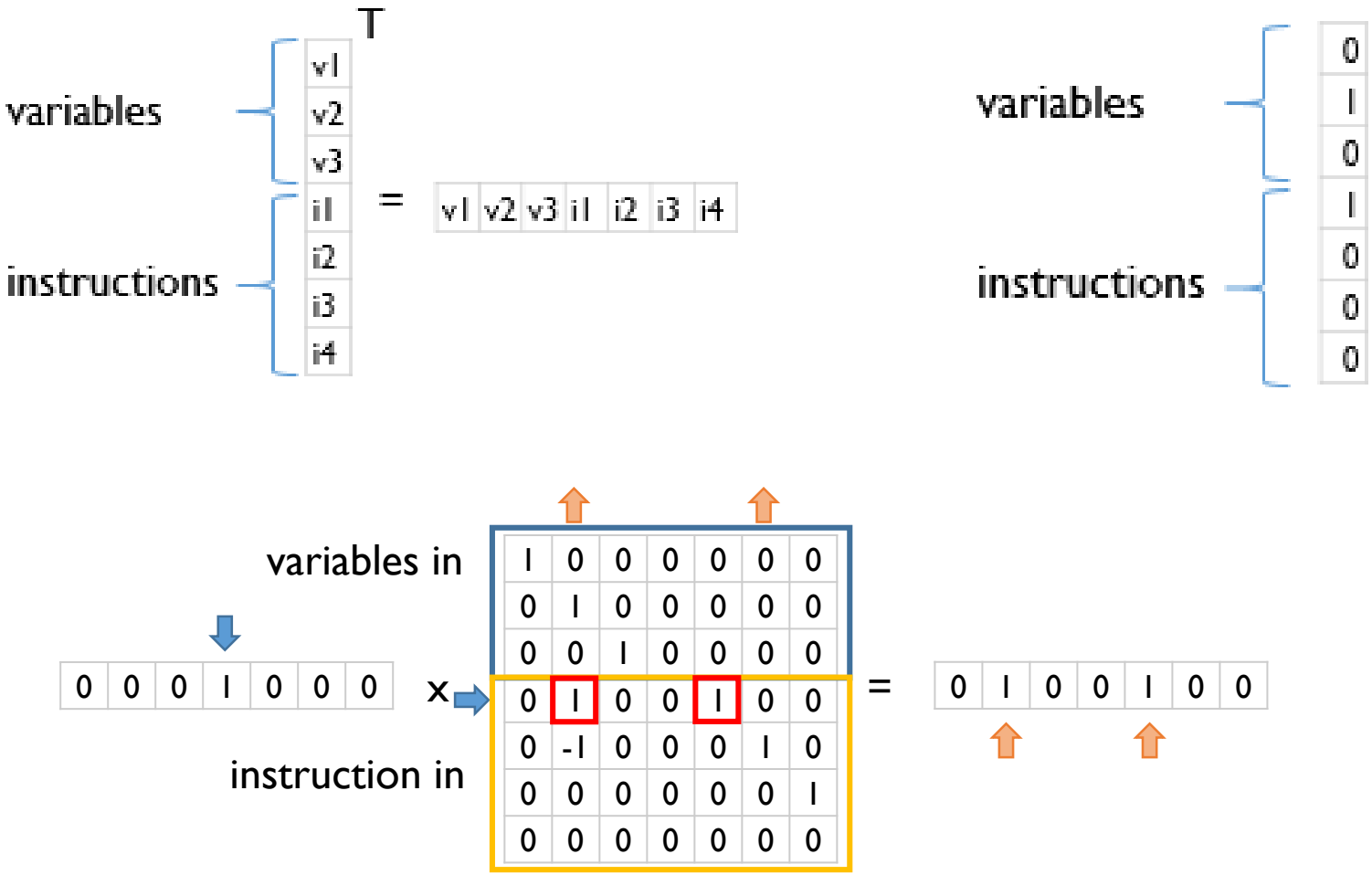
$$\begin{array}{lcccl} N_i & \longrightarrow & \boxed{1} & \longrightarrow & N_{i+1} \\ N_i & \longrightarrow & \boxed{1} & \longrightarrow & N_V \end{array}$$

- If there is a decrement operation $(V \leftarrow \max\{0, V - 1\})$ on row i then augment the network as follows:

$$\begin{array}{lcccl} N_i & \longrightarrow & \boxed{1} & \longrightarrow & N_{i+1} \\ N_i & \longrightarrow & \boxed{-1} & \longrightarrow & N_V \end{array}$$

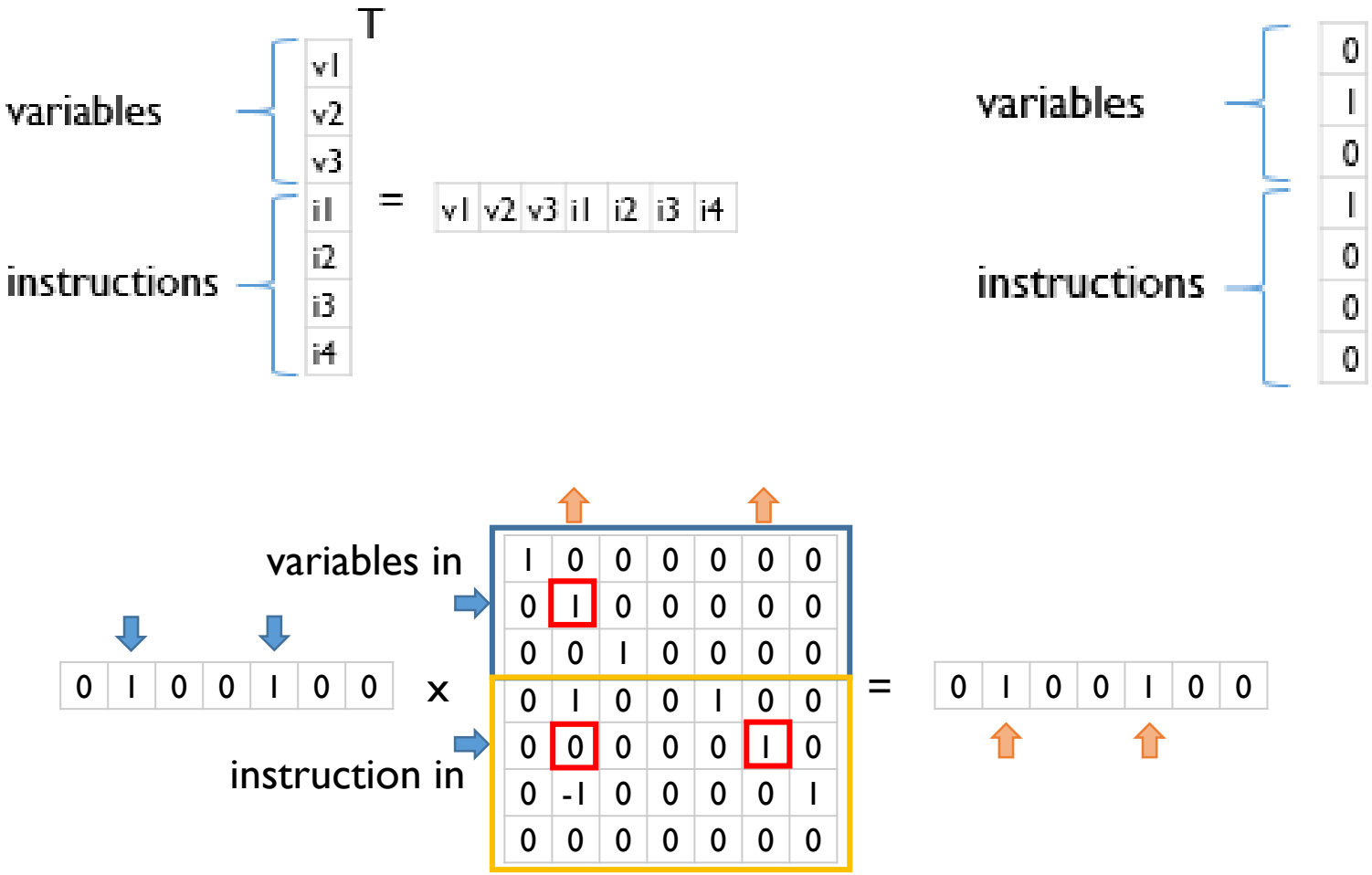
Turing Recurrent Network

[Hyotyniemi 1996]



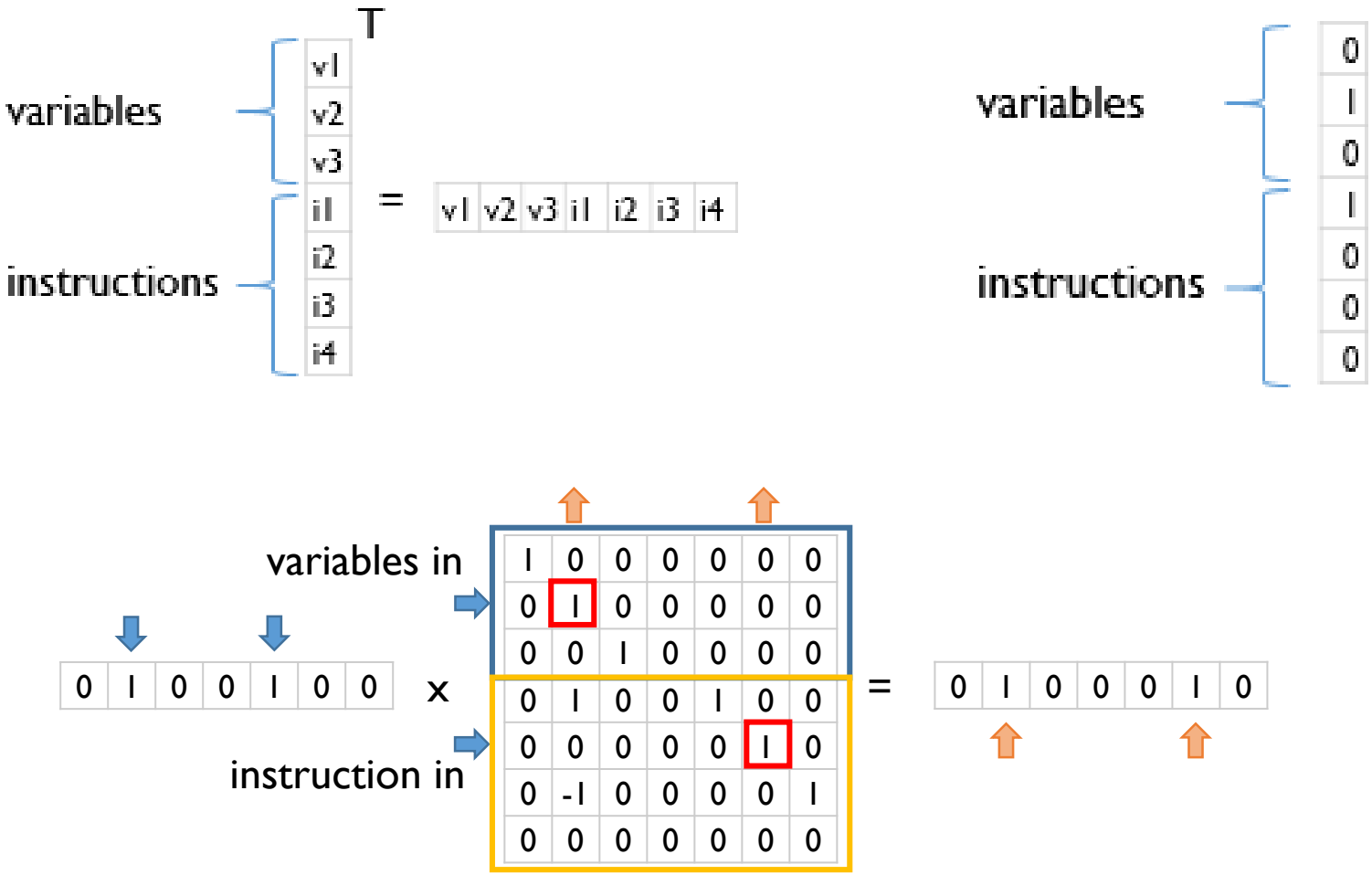
Turing Recurrent Network

[Hyotyniemi 1996]



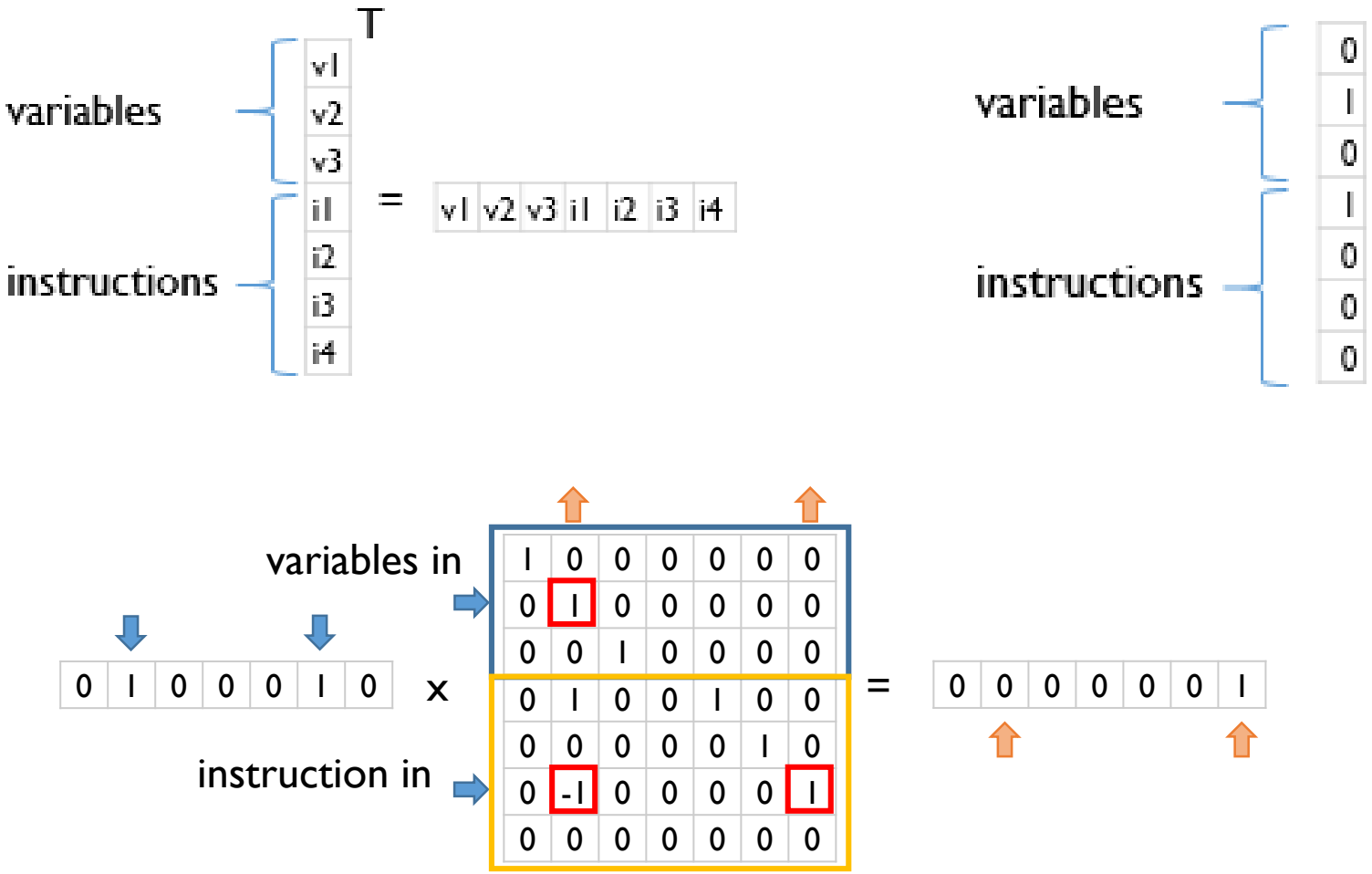
Turing Recurrent Network

[Hyotyniemi 1996]



Turing Recurrent Network

[Hyotyniemi 1996]



Turing Recurrent Network

[Hyotyniemi 1996]

Given a program, make the following network

- If there is a conditional branch (`IF` $V \neq 0$ `GOTO` j) on row i then augment the network as follows:

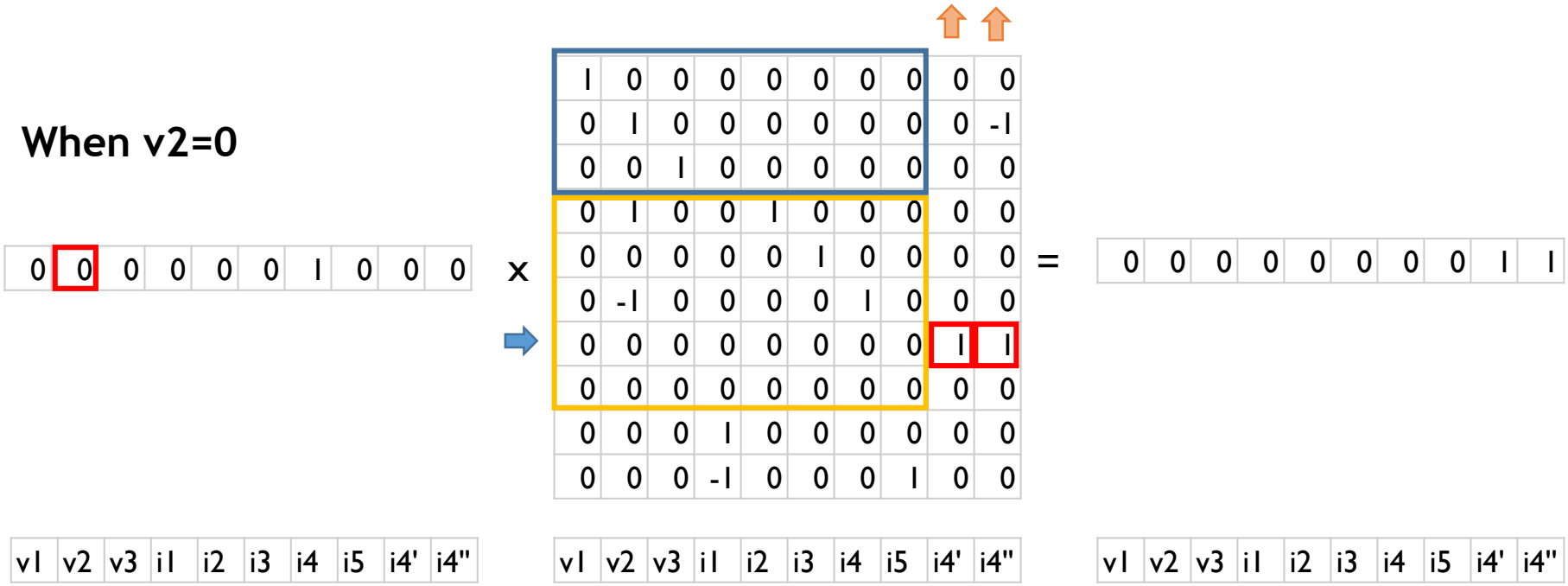
$$
\begin{array}{lcccl}
N_i & \longrightarrow & \boxed{1} & \longrightarrow & N_{i'} \\
N_i & \longrightarrow & \boxed{1} & \longrightarrow & N_{i''} \\
N_V & \longrightarrow & \boxed{-1} & \longrightarrow & N_{i''} \\
N_{i''} & \longrightarrow & \boxed{1} & \longrightarrow & N_{i+1} \\
N_{i'} & \longrightarrow & \boxed{1} & \longrightarrow & N_j \\
N_{i''} & \longrightarrow & \boxed{-1} & \longrightarrow & N_j .
\end{array}
$$

Turing Recurrent Network

[Hyotyniemi 1996]

i4: if v2 ≠ 0 GOTO i1

When v2=0

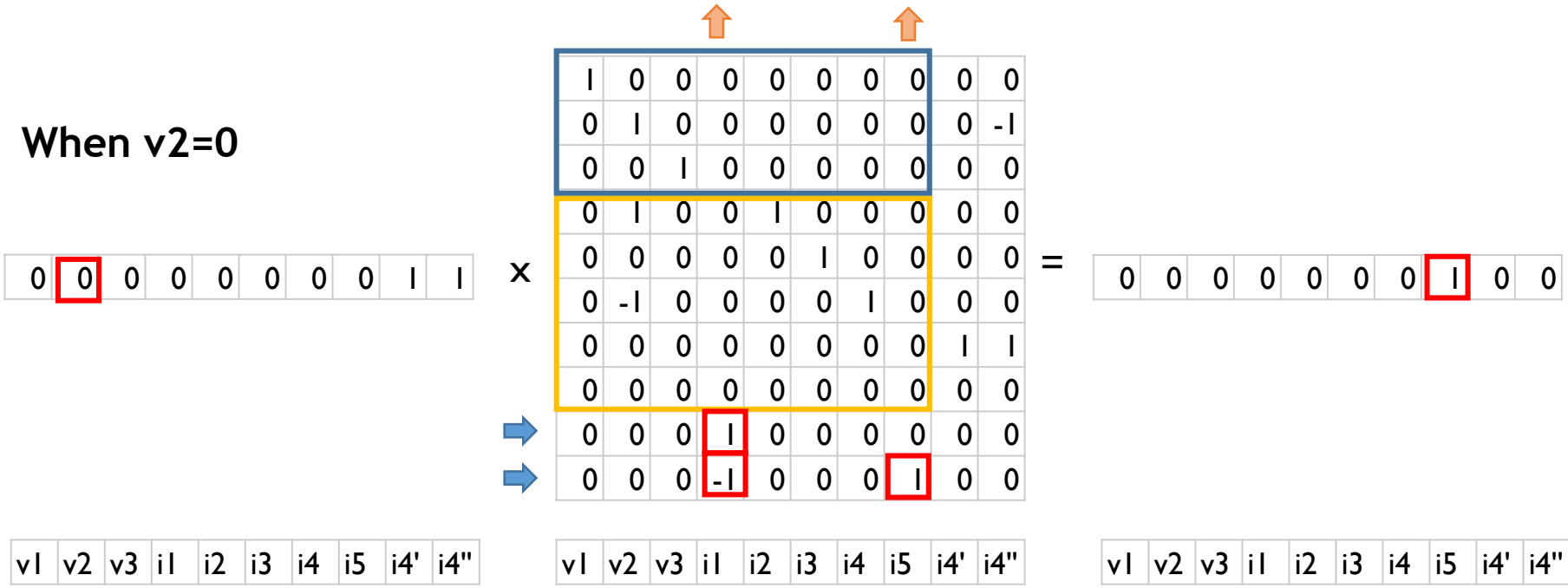


Turing Recurrent Network

[Hyotyniemi 1996]

i4: if v2 ≠ 0 GOTO i1

When v2=0

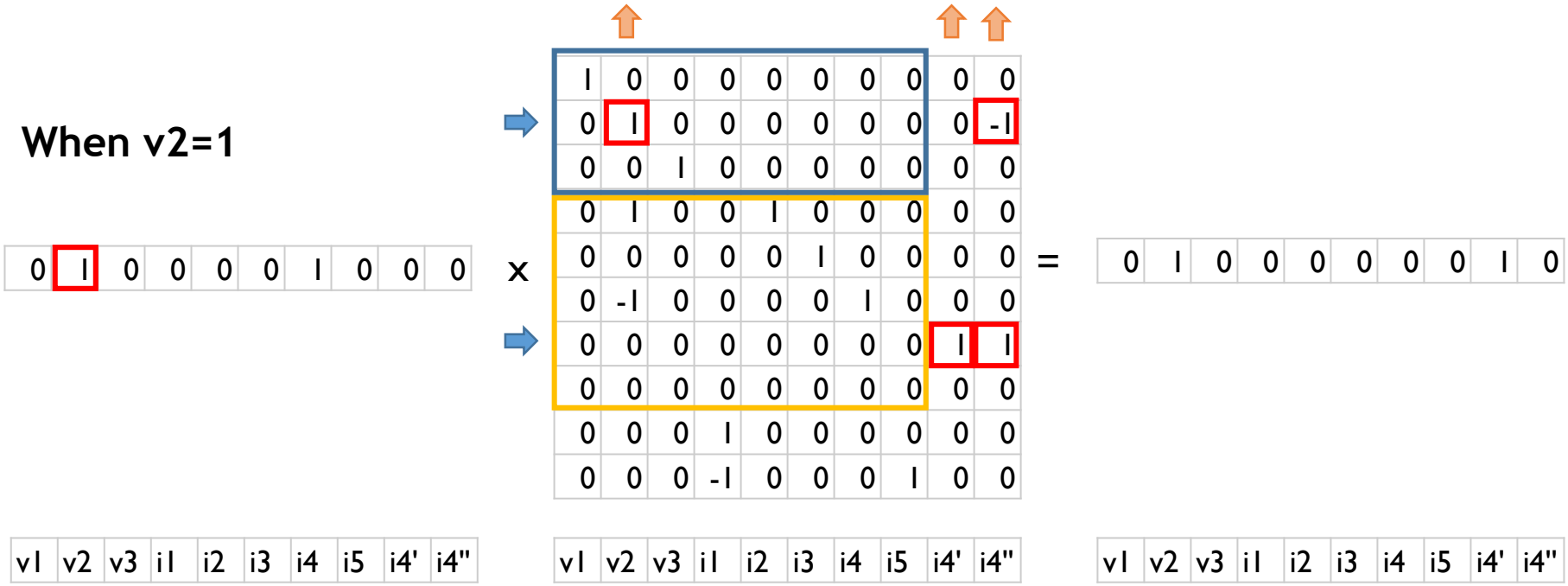


Turing Recurrent Network

[Hyotyniemi 1996]

i4: if v2 ≠ 0 GOTO i1

When v2=1

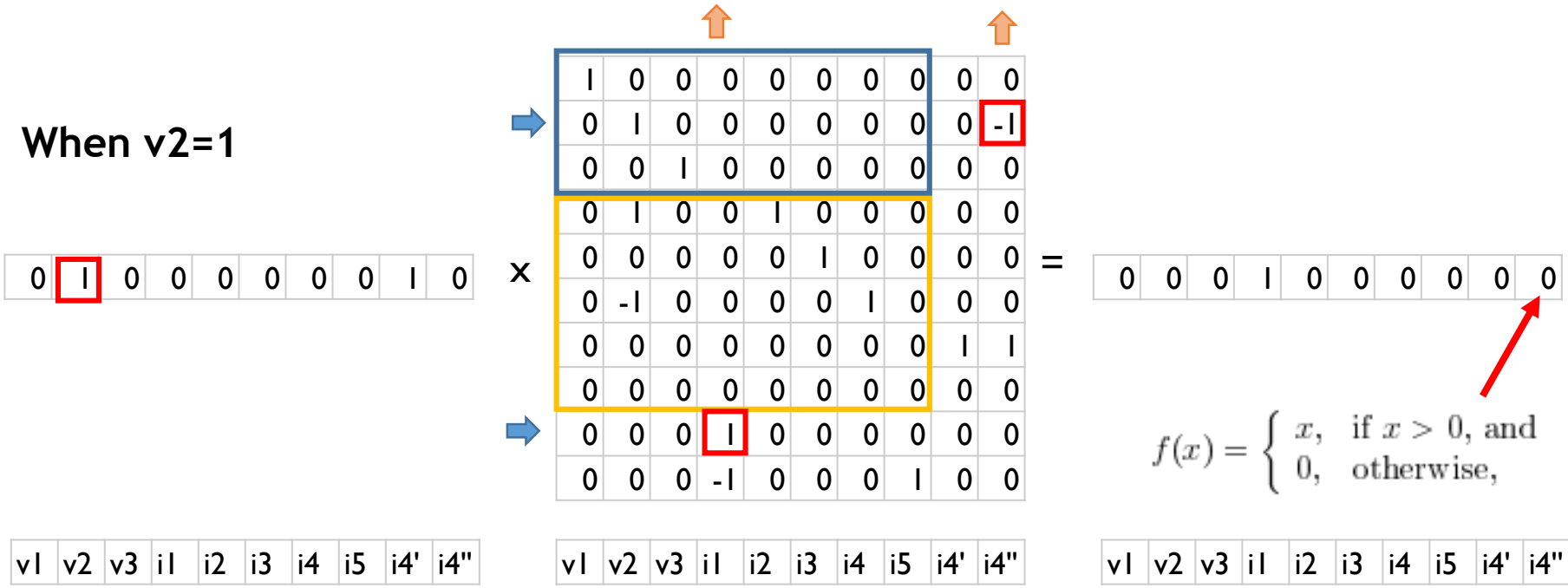


Turing Recurrent Network

[Hyotyniemi 1996]

i4: if v2 ≠ 0 GOTO i1

When v2=1



Turing Recurrent Network

[Hyotyniemi 1996]

1. V0 ← V0 - 1
2. V1 ← V1 - 1
3. IF V0 ≠ 0 GOTO 6
4. V0 ← V0 + 1
5. GOTO 9
6. V0 ← V0 - 1
7. IF V1 ≠ 0 GOTO 10
8. V1 ← V1 + 1
9. GOTO 3
10. END (Do Nothing)

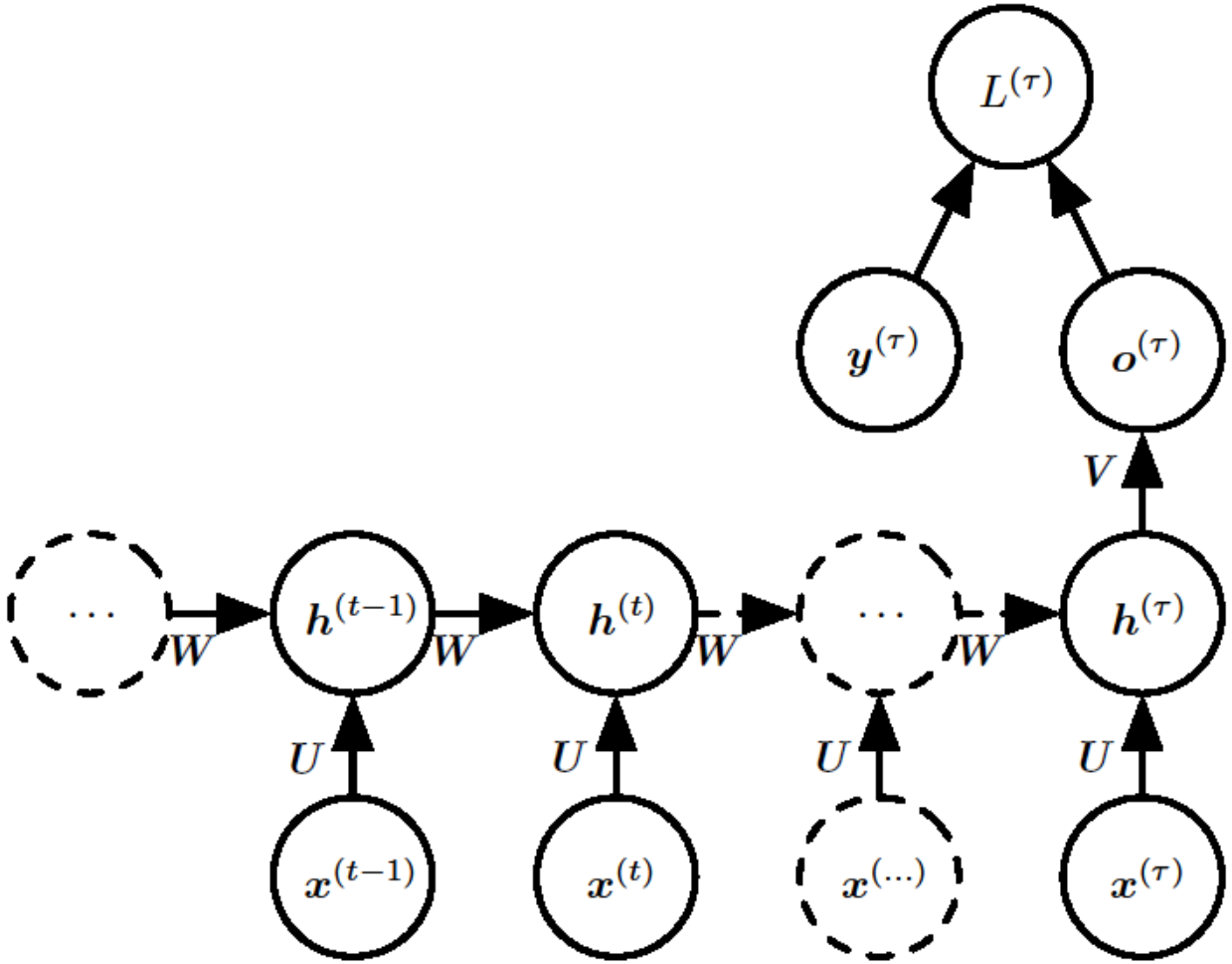
	V0	V1	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i3'	i3"	i7'	i7"
V0																
V1																
i1																
i2																
i3																
i4																
i5																
i6																
i7																
i8																
i9																
i10																
i3'																
i3"																
i7'																
i7"																

Find the Transition Matrix in the Turing Recurrent Network

1. V0 ← V0 - 1
2. V1 ← V1 - 1
3. IF V0 ≠ 0 GOTO 6
4. V0 ← V0 + 1
5. GOTO 9
6. V0 ← V0 - 1
7. IF V1 ≠ 0 GOTO 10
8. V1 ← V1 + 1
9. GOTO 3
10. END (Do Nothing)

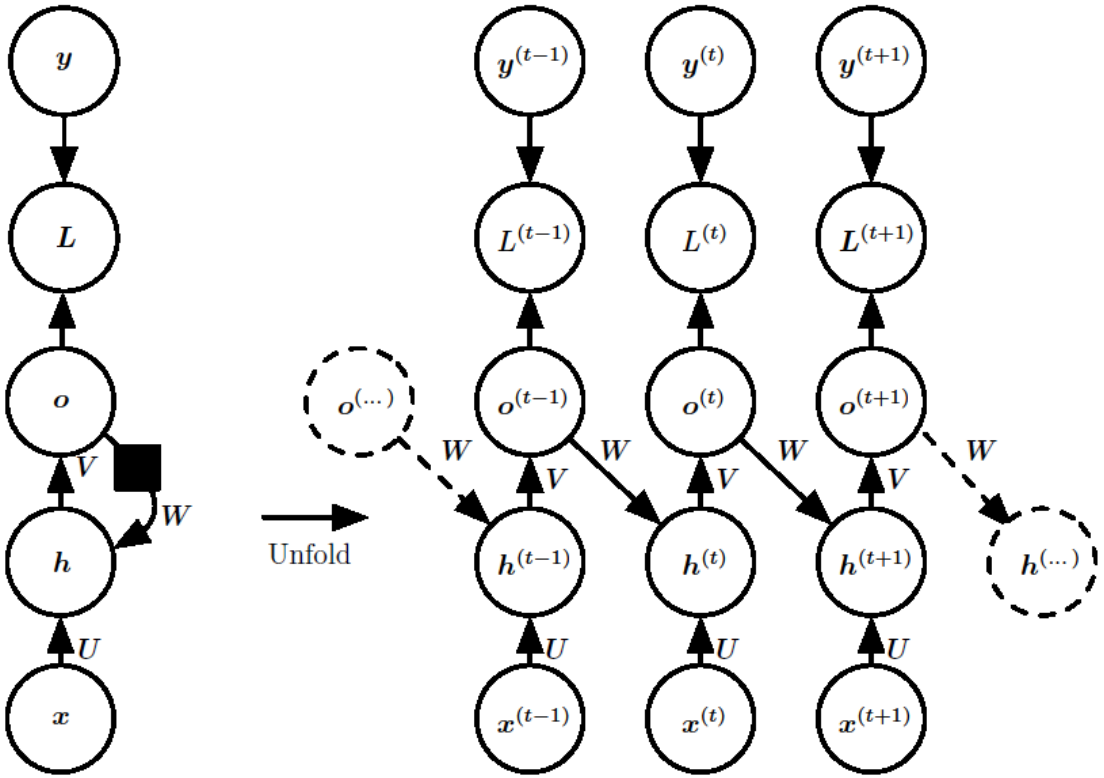
	V0	V1	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i3'	i3"	i7'	i7"
V0	1													-1		
V1		1														-1
i1	-1			1												
i2		-1			1											
i3													1	1		
i4	1						1									
i5											1					
i6	-1								1							
i7															1	1
i8		1									1					
i9					1											
i10																
i3'								1								
i3"						1		-1								
i7'												1				
i7"										1		-1				

Find the Transition Matrix in the Turing Recurrent Network



RNN with single output

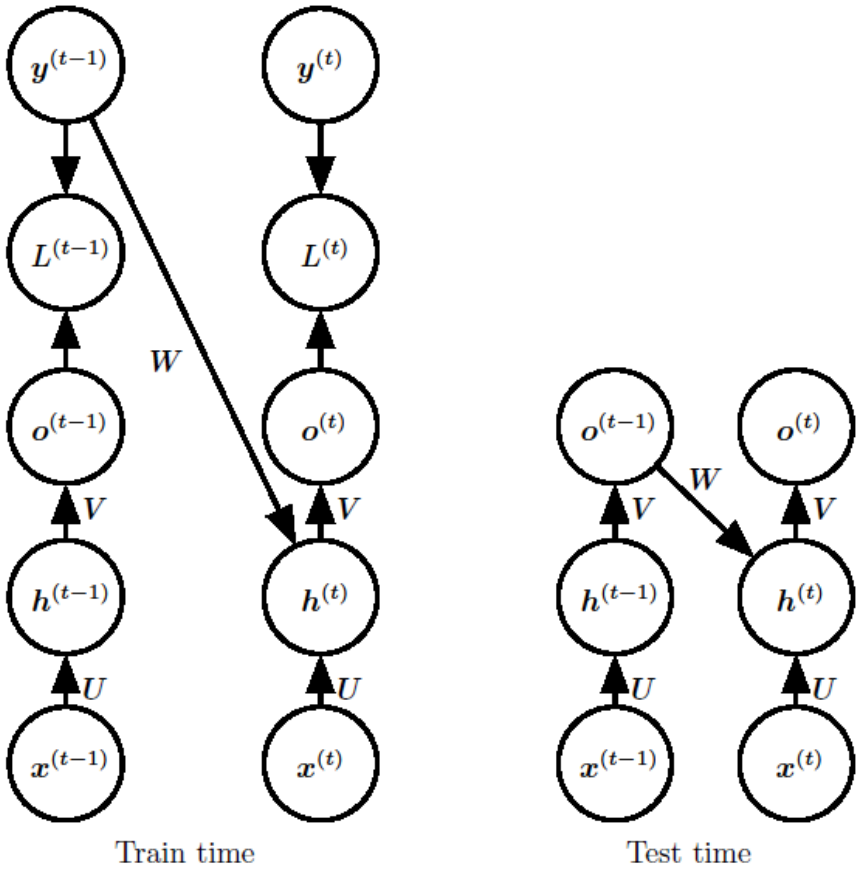
[Goodfellow et al., 2016] http://www.deeplearningbook.org



This RNN (called Jordan Net) is not Turing complete
unless output o^t is expressive enough to include all the information in h^t.

Recurrence through only the Output

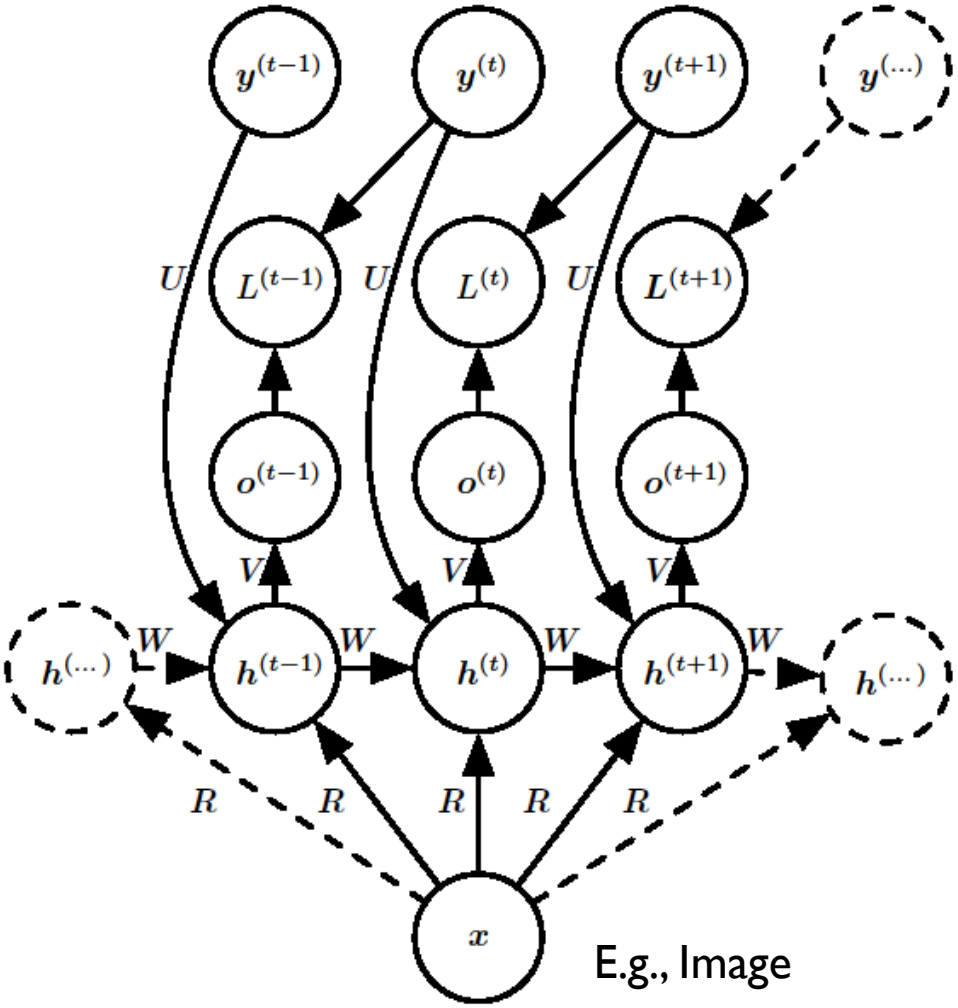
[Goodfellow et al., 2016] http://www.deeplearningbook.org



However, this can be trained parallel, without using the BPTT algorithm.

Recurrence through only the output: Teacher Forcing

[Goodfellow et al., 2016] http://www.deeplearningbook.org



E.g., Image Caption

E.g., Image

Vector to Sequence

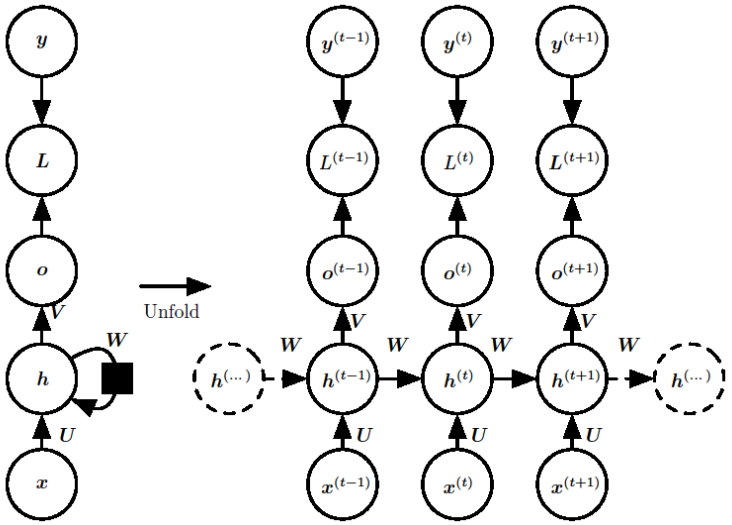
[Goodfellow et al., 2016] http://www.deeplearningbook.org

Gradient of loss (L) at $o^{(t)}$ for ith output

$$(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}$$

Gradient of loss (L) at $h^{(\tau)}$ when τ is the last time step

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L.$$



Gradient of loss (L) at $h^{(t)}$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= \boldsymbol{W}^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t+1)}\right)^2\right) + \boldsymbol{V}^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

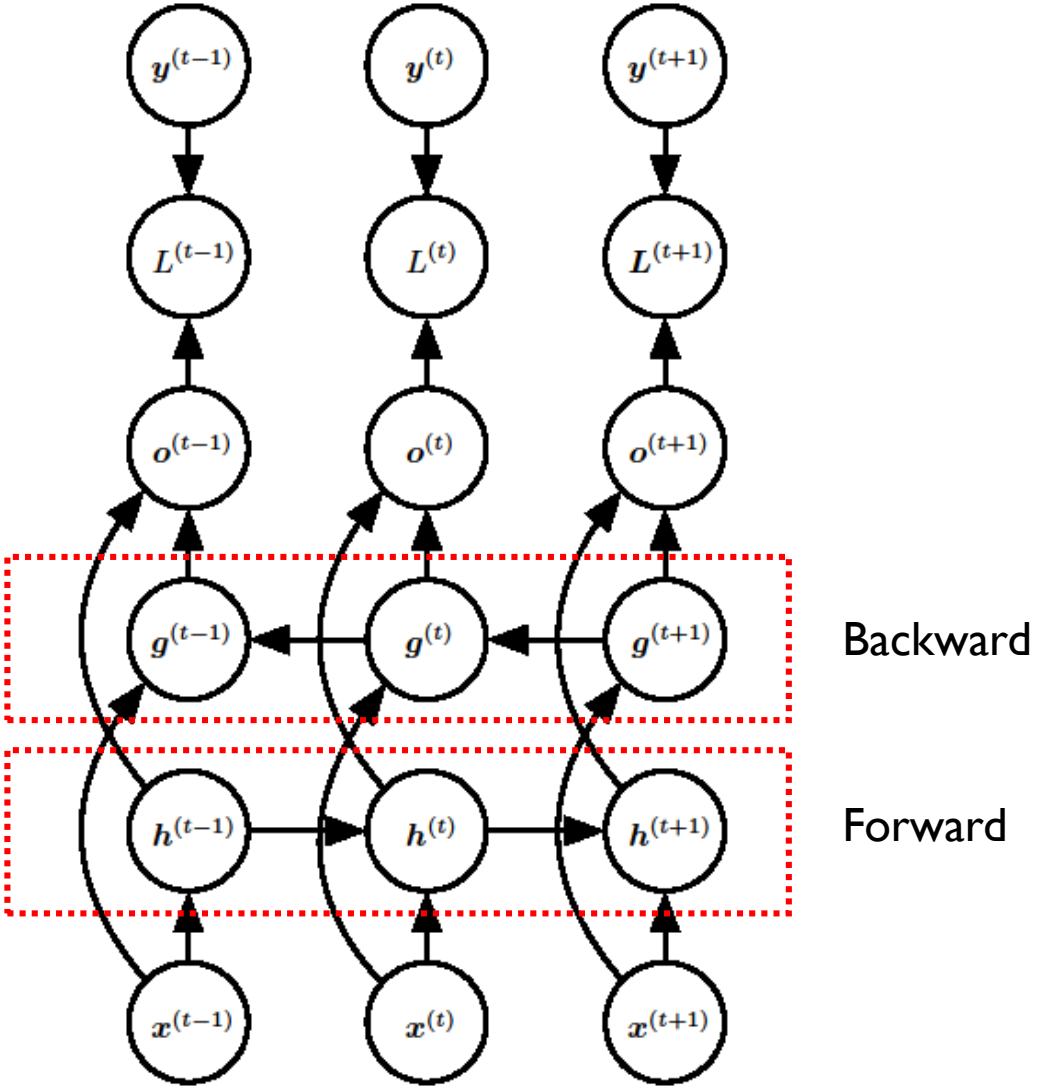
Gradient from future

Derivative of tanh()

Gradient from current output

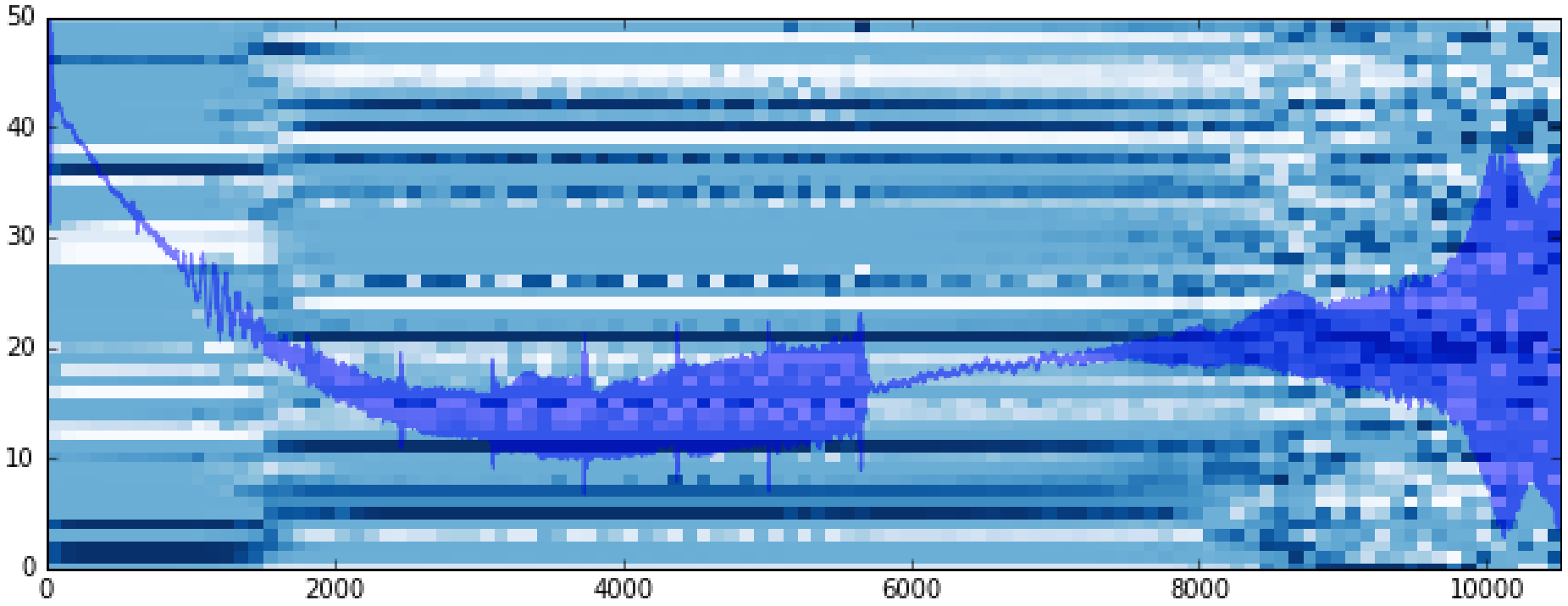
Backpropagation in RNN

[Goodfellow et al., 2016] http://www.deeplearningbook.org



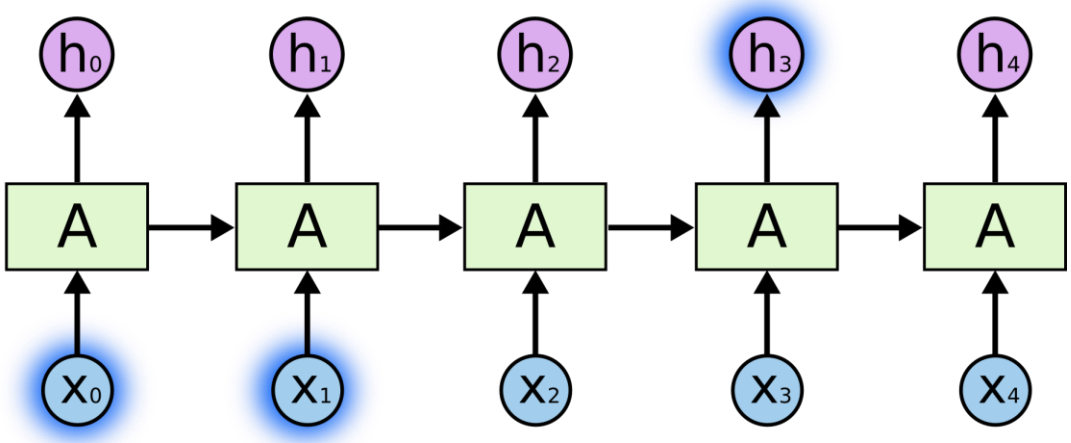
Bidirectional RNNs

[Goodfellow et al., 2016] http://www.deeplearningbook.org

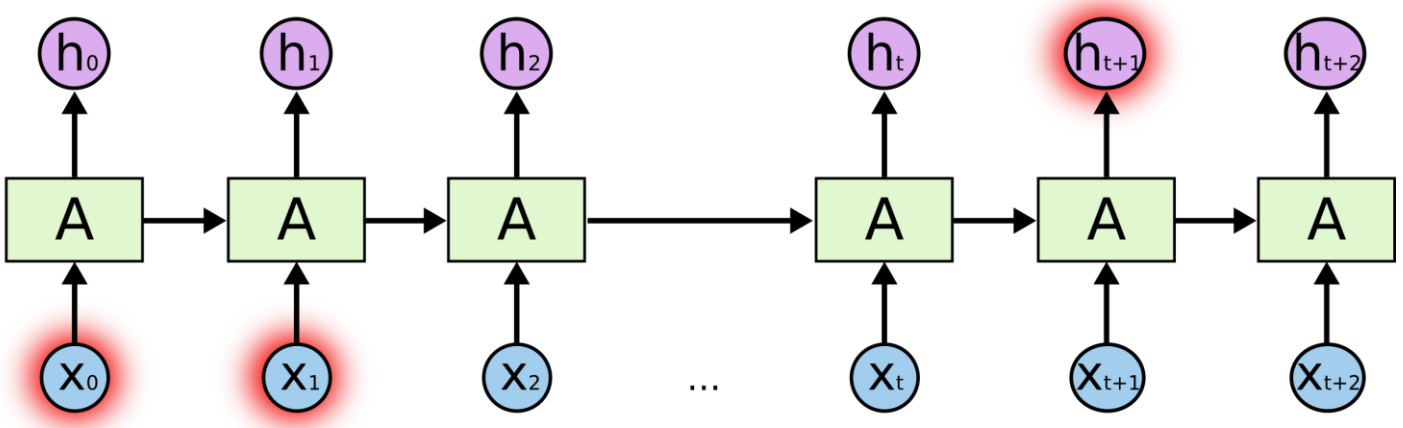


* Courtesy of Prof. Seungchul Lee (UNIST, http://isystems.unist.ac.kr/)

Recurrent Neural Network (RNN): A Case Study

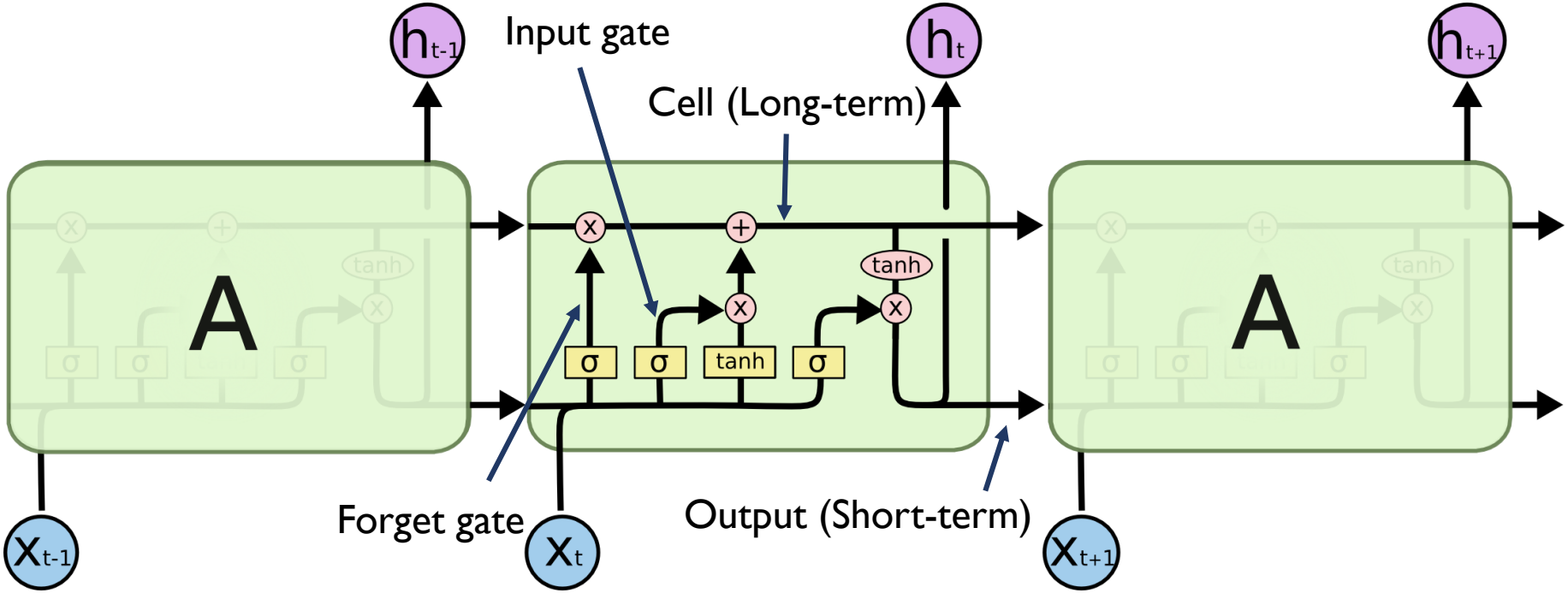


RNNs can learn a short-term dependencies.



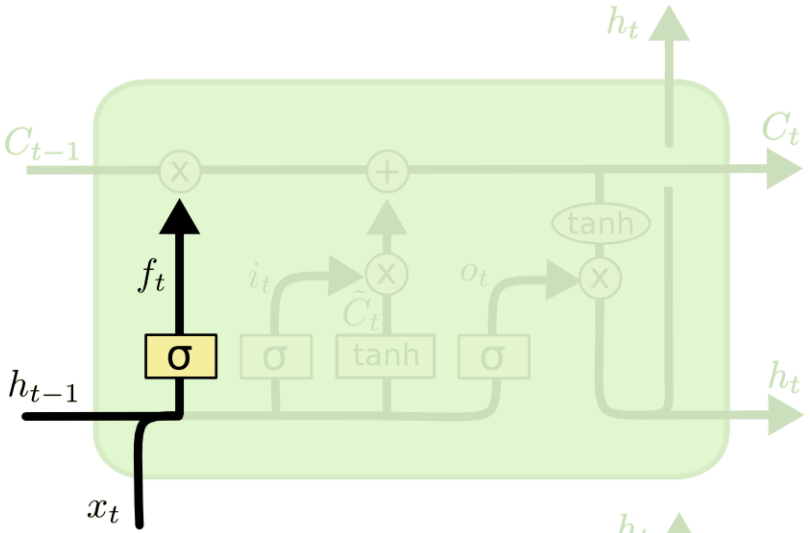
It is very hard to learn a long-term dependencies from a RNN model.

Recurrent Neural Network (RNN)
Long/short-term dependencies

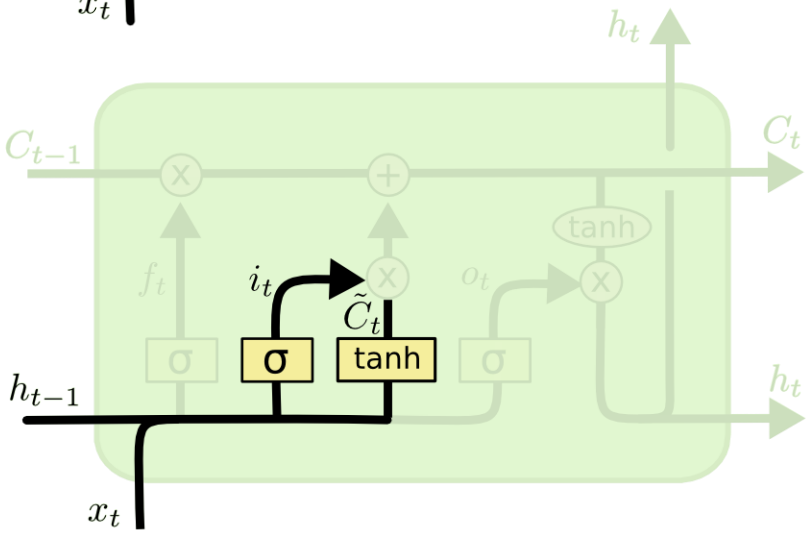


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



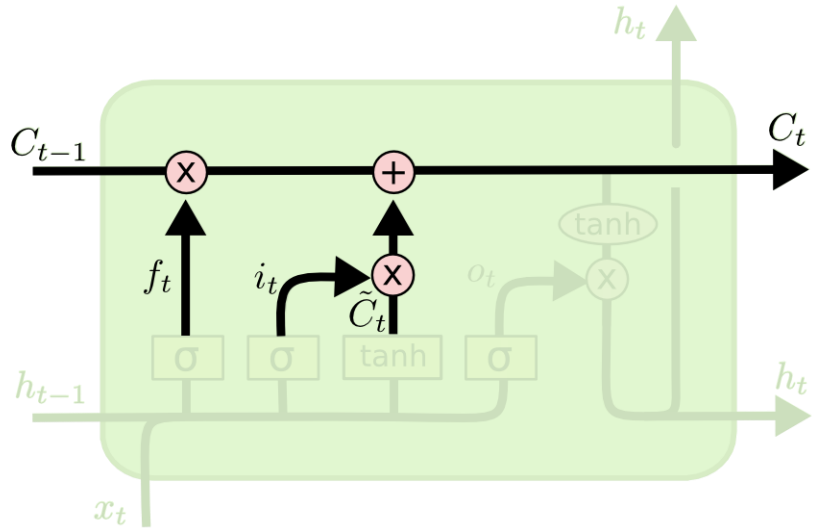
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



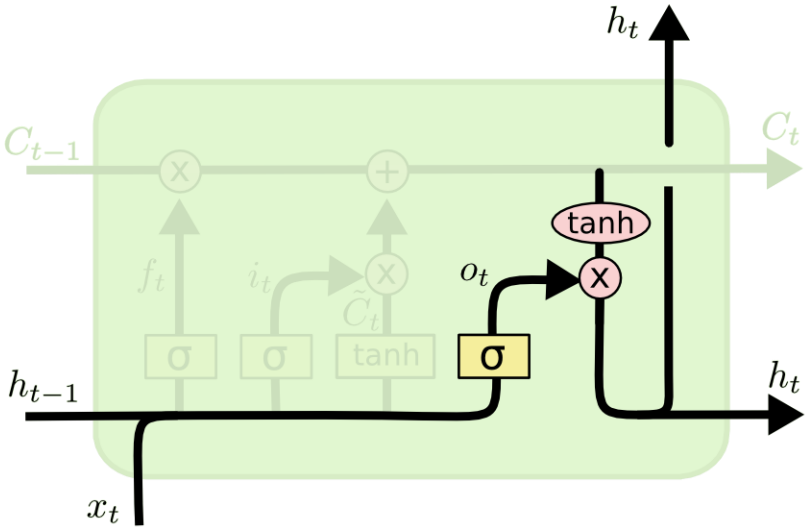
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Long Short Term Memory (LSTM)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

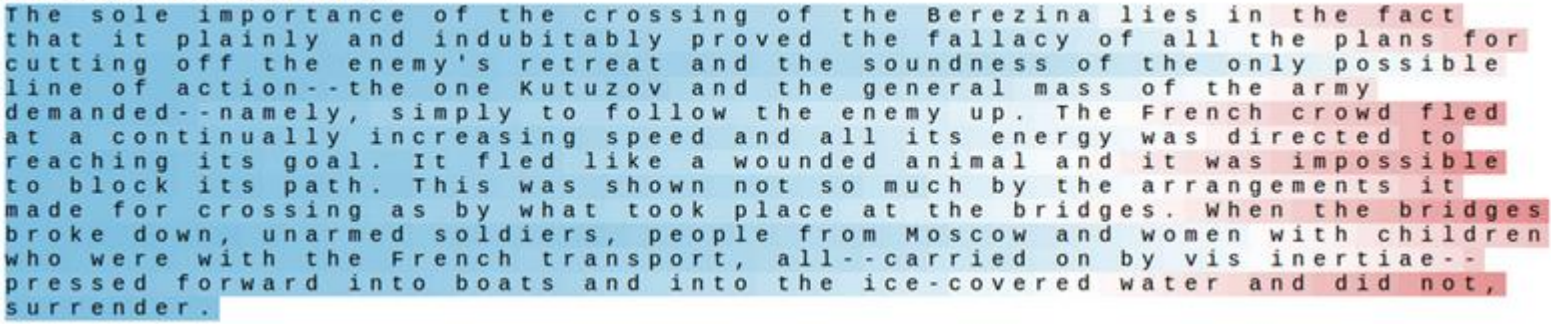


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

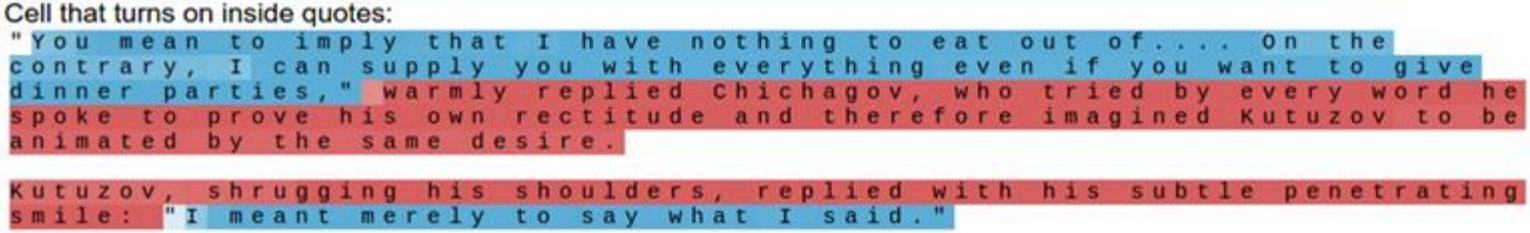
$$h_t = o_t * \tanh\left(C_t\right)$$

Long Short Term Memory (LSTM)

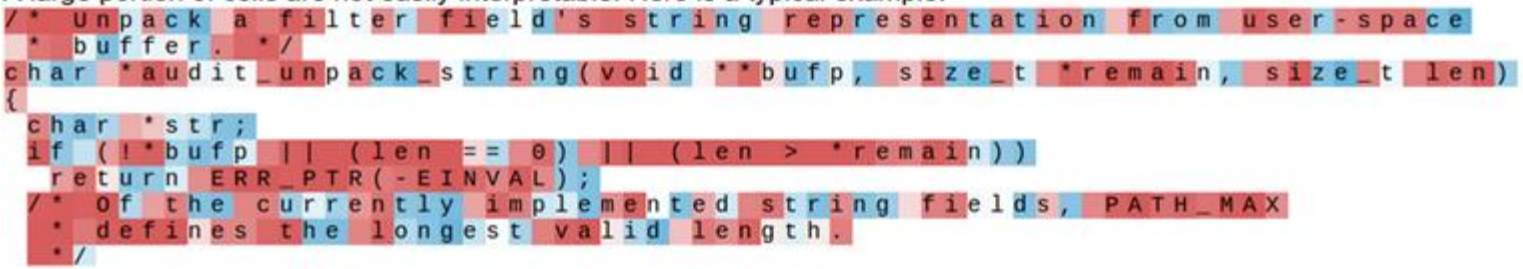
Cell sensitive to position in line:



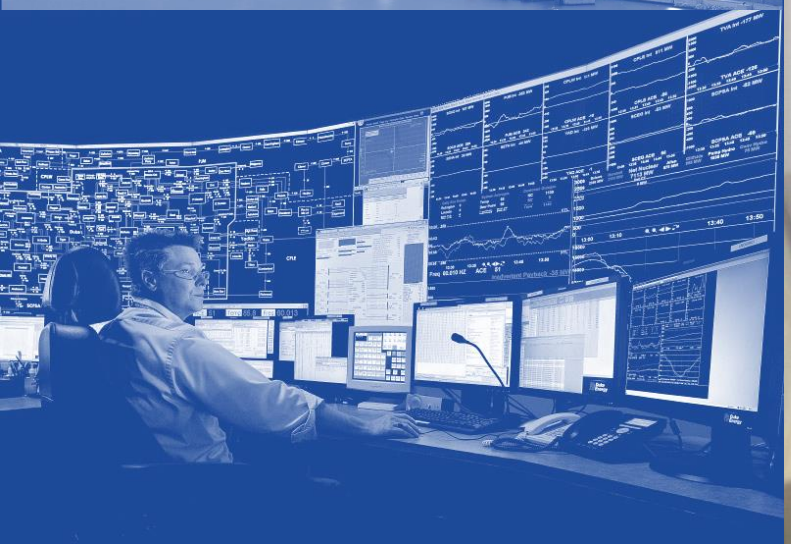
Cell that turns on inside quotes:

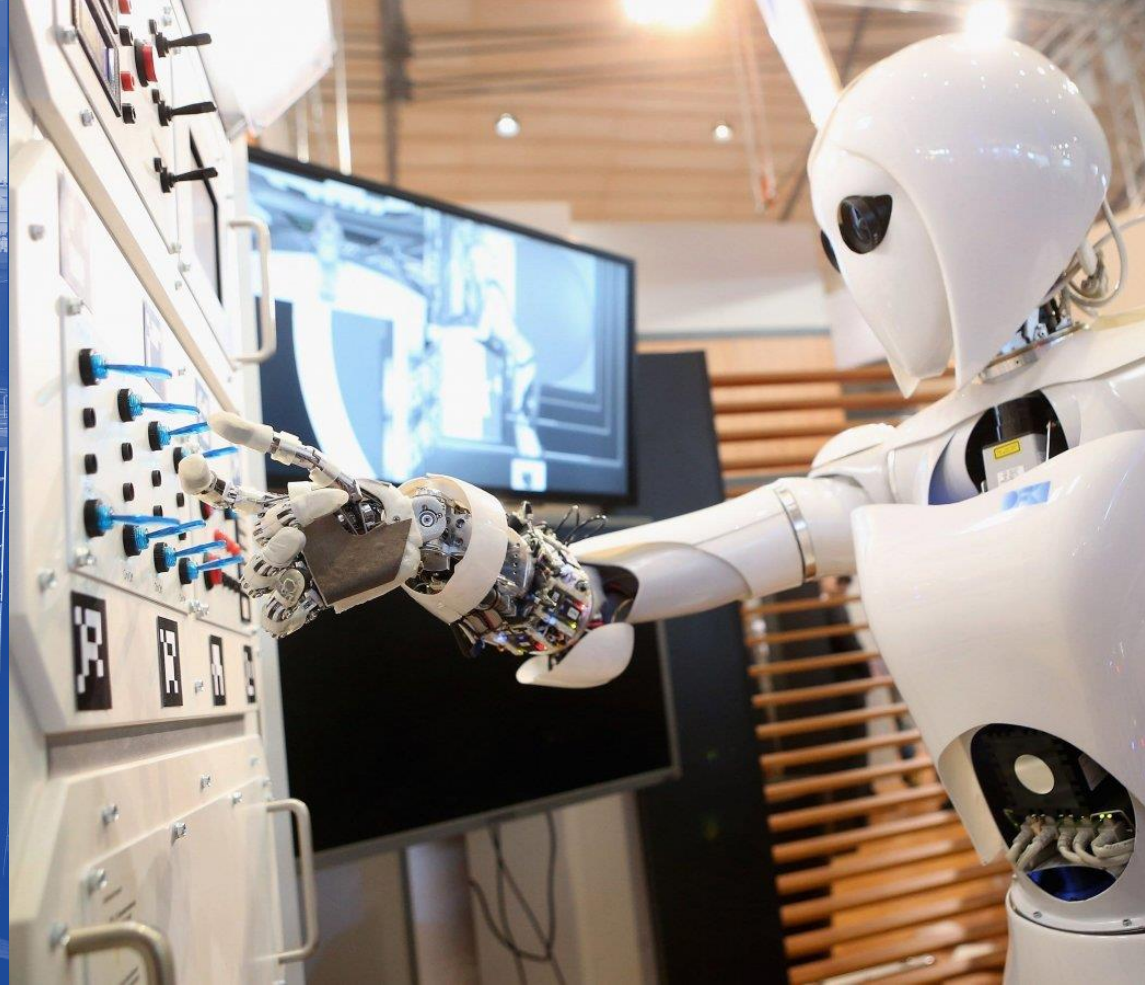


A large portion of cells are not easily interpretable. Here is a typical example:

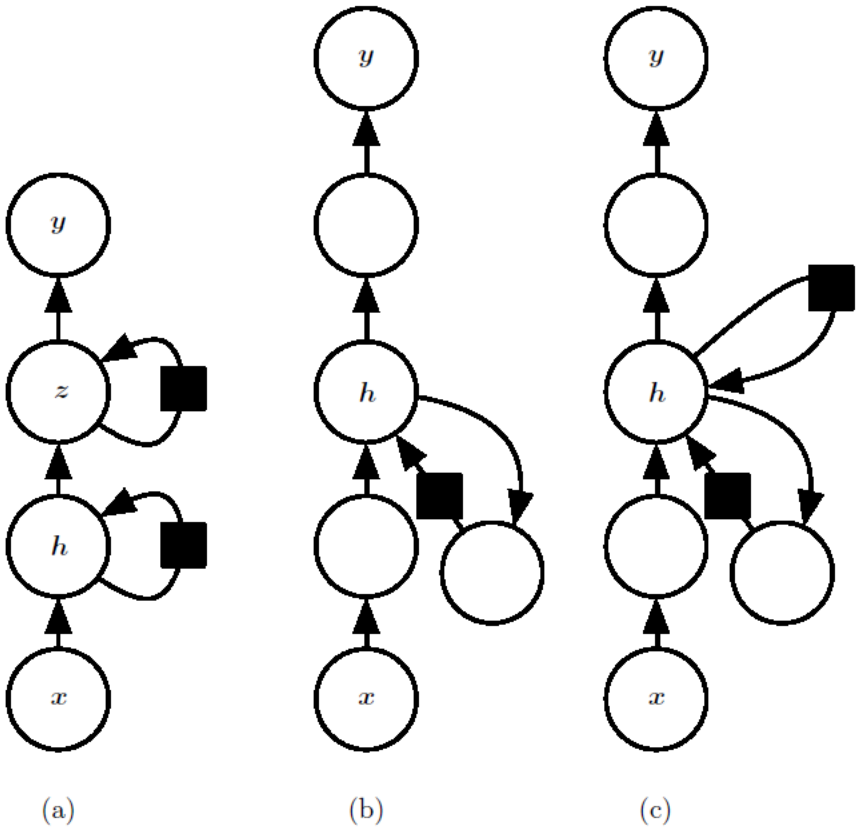


Long Short Term Memory (LSTM): A Case Study





What are the Current/Future Issues?



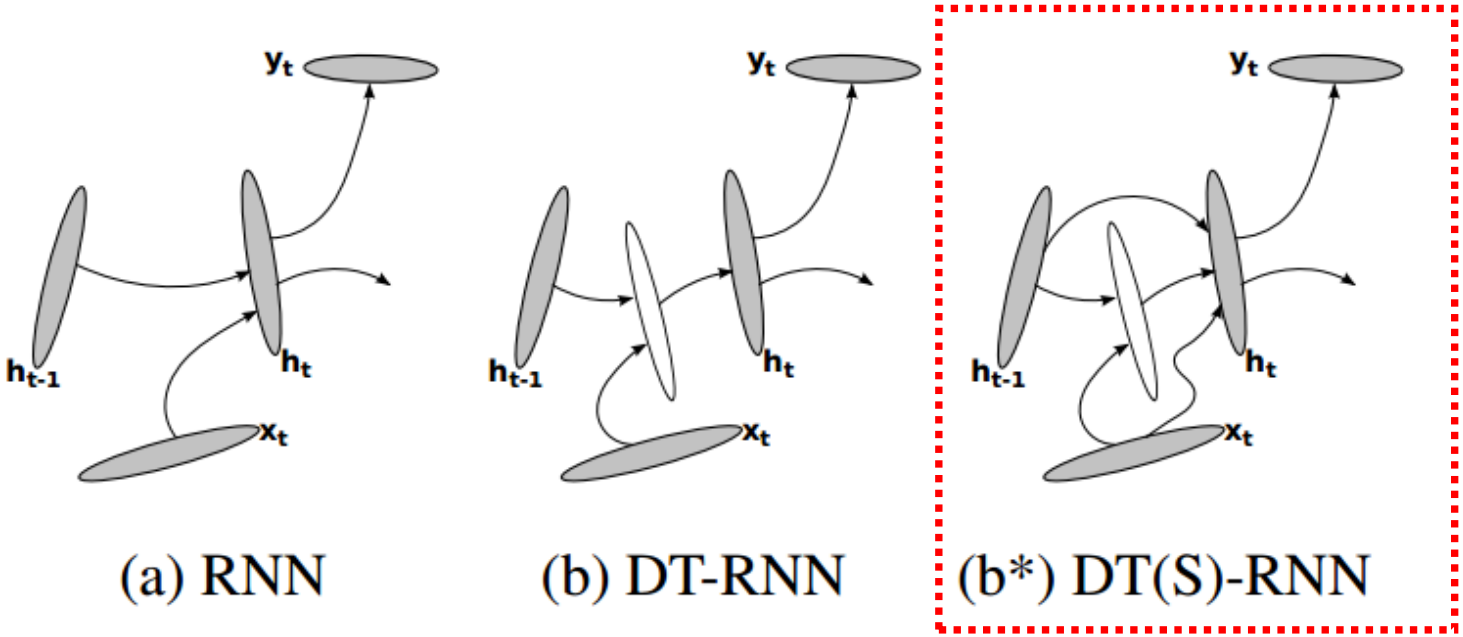
(a) Deep network for output

(b) Deep network for transition

(c) Deep network for transition with skip

Deep RNNs

[Goodfellow et al., 2016] http://www.deeplearningbook.org



Deep RNNs over Transition

[Pascanu et al., 2014]

RNN in a time step

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

RNN after t time steps

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

When W admits an eigendecomposition of the form

$$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}$$

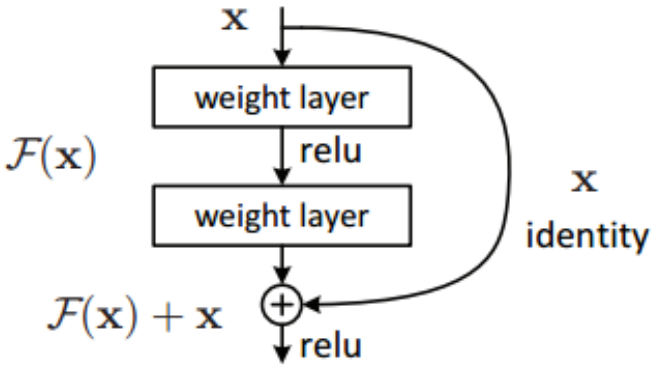
RNN after t time steps

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

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

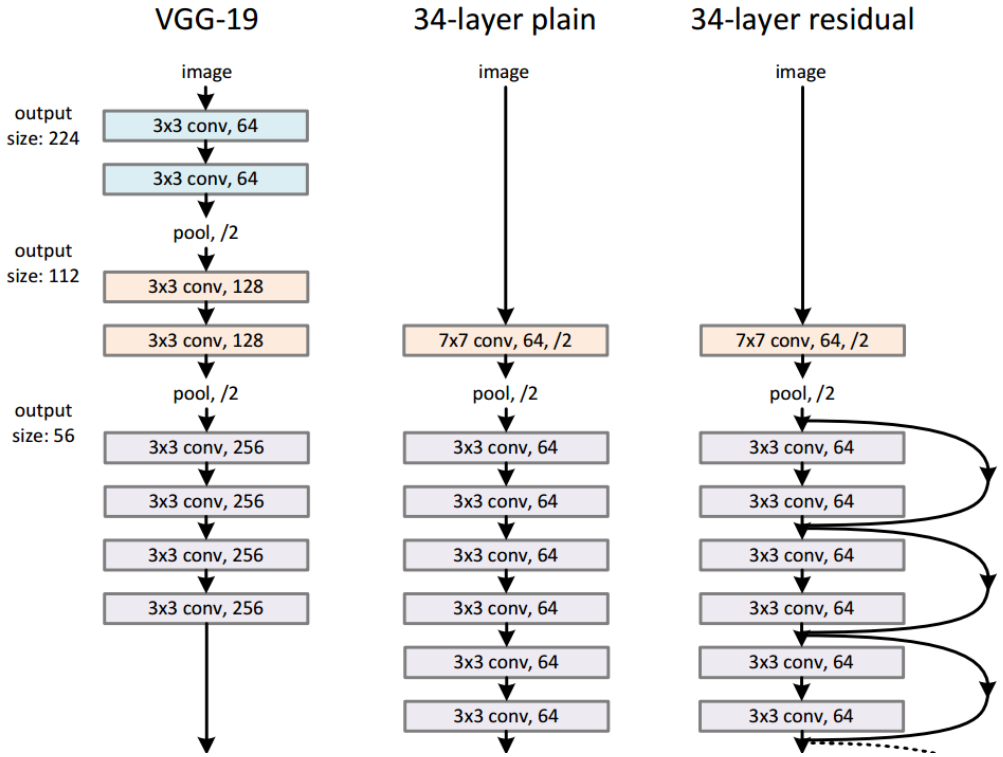
Challenges of Long-Term Dependencies

[Goodfellow et al., 2016] http://www.deeplearningbook.org



Residual learning

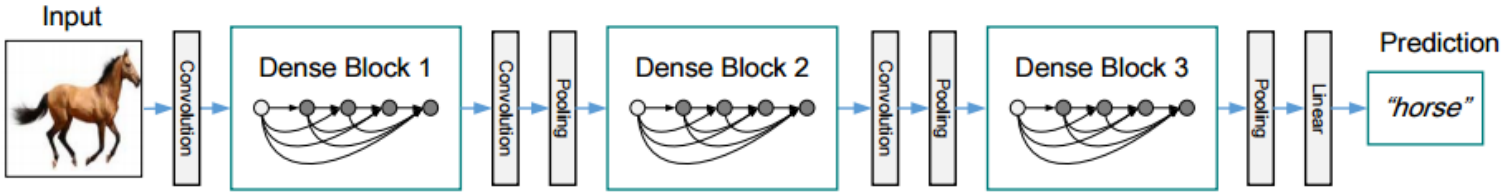
$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$$

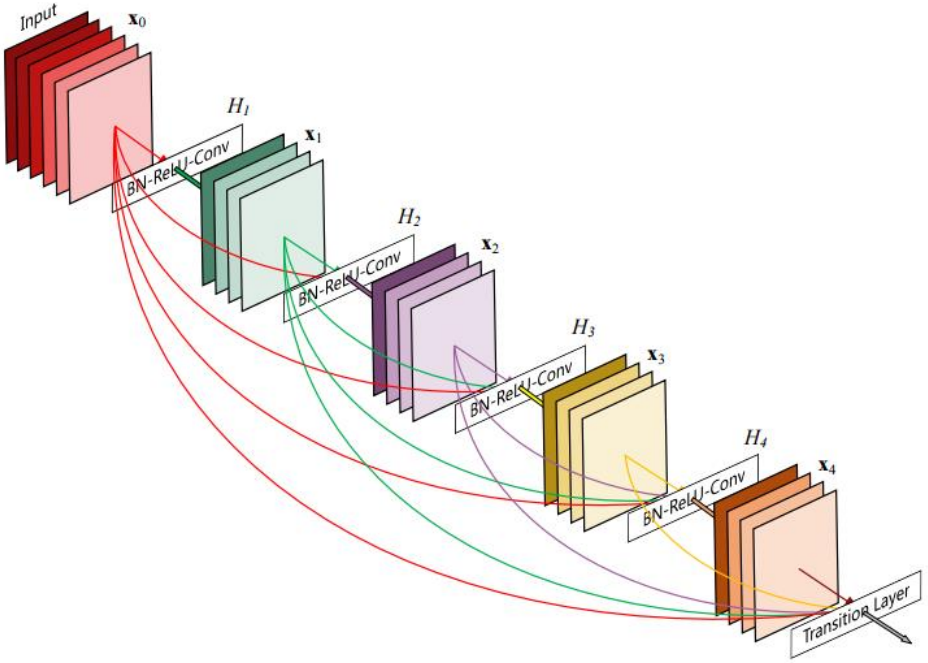


Comparison of Resnet

3.6% of error in ImageNet Challenge, 2015

Residual Network (ResNet, He et. al., 2015)



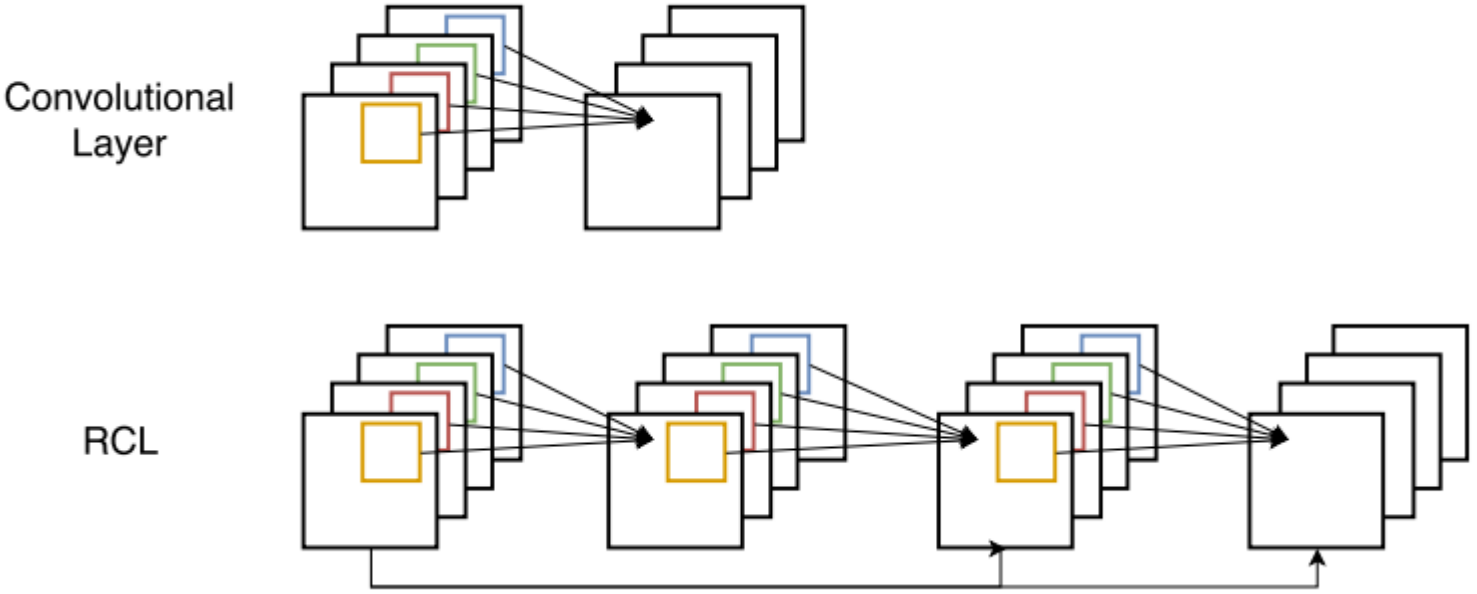


$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

Better Performance than ResNet

CIFAR10 (3.74% -ResNet 4.62%)
CIFAR 100 (19.25% - ResNet 22.71%)

Densely Connected Convolutional Networks (DensNet, Huang et. Al., 2016)

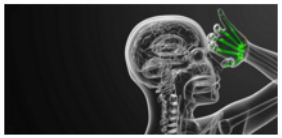


Recurrent Convolutional Layer (RCL)

$$x_l = x_{l-1} + H_l(x_{l-1}) + H_l(H_l(x_{l-1})) + H_l(H_l(H_l(x_{l-1})))$$

* Figure is drawn by Subin Yi

Recurrent Convolutional Neural Layers (RCNN, Liang and Hu, 2015)

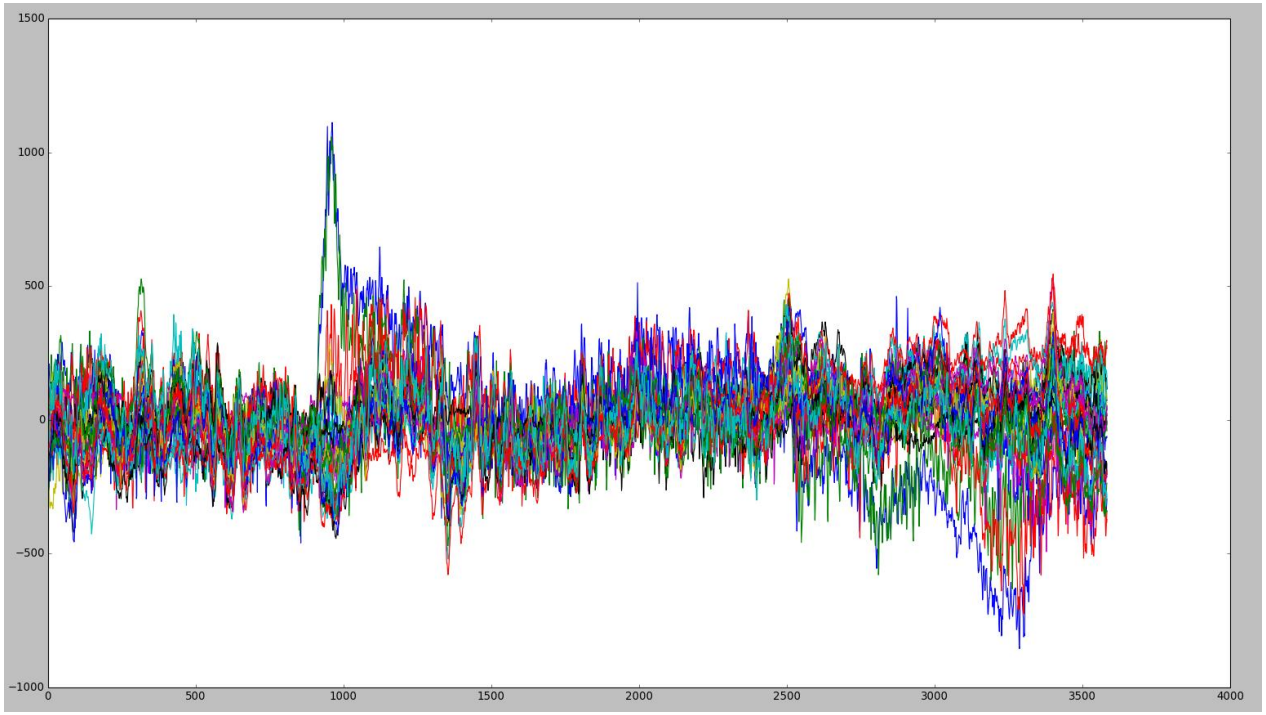


Completed • $10,000 • 379 teams

Grasp-and-Lift EEG Detection

Mon 29 Jun 2015 – Mon 31 Aug 2015 (4 months ago)

One chunk: Data: 3584,32



Hand Start

First Digit Touch

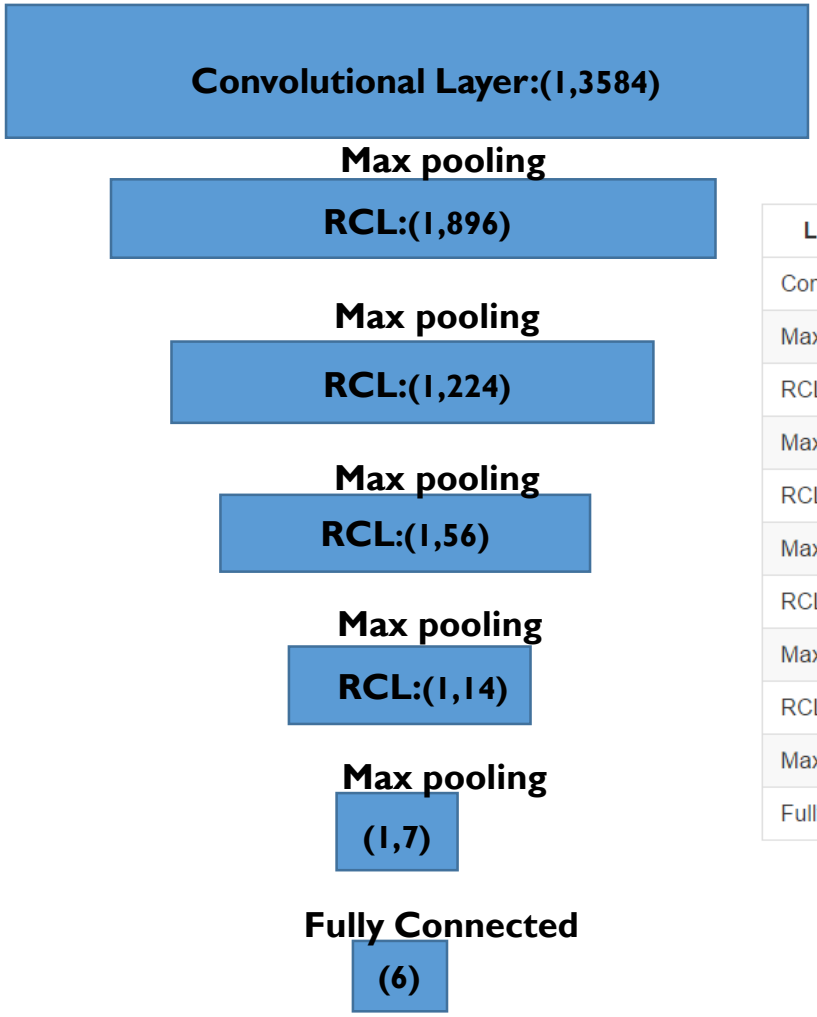
Lift off

Replace

Both Released

* Joint work with Azamatbek Akhmedov

RCNN on EEG Analysis



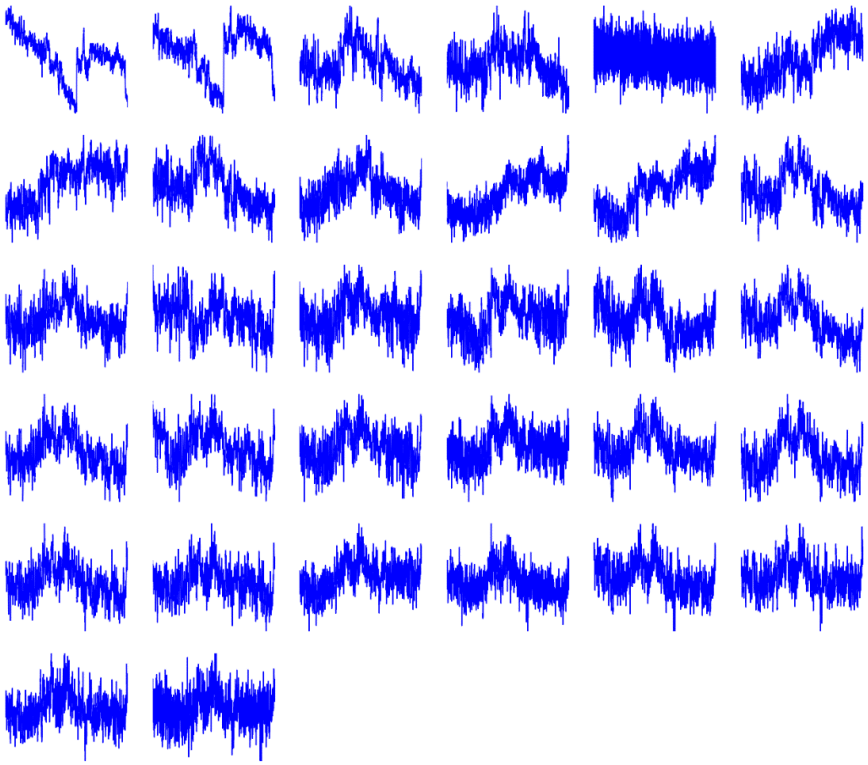
Applying RCL

Layer type	Size	Output shape
Convolutional	256 1×9 filters	(64, 256, 1, 3584)
Max pooling	Pool size 4, stride 4	(64, 256, 1, 896)
RCL	256 1×1 feed-forward filters, 256 1×9 filters, 3 iterations	(64, 256, 1, 896)
Max pooling	Pool size 4, stride 4	(64, 256, 1, 224)
RCL	256 1×1 feed-forward filters, 256 1×9 filters, 3 iterations	(64, 256, 1, 224)
Max pooling	Pool size 4, stride 4	(64, 256, 1, 56)
RCL	256 1×1 feed-forward filters, 256 1×9 filters, 3 iterations	(64, 256, 1, 56)
Max pooling	Pool size 4, stride 4	(64, 256, 1, 14)
RCL	256 1×1 feed-forward filters, 256 1×9 filters, 3 iterations	(64, 256, 1, 14)
Max pooling	Pool size 2, stride 2	(64, 256, 1, 7)
Fully connected	1792×6	(64, 6)

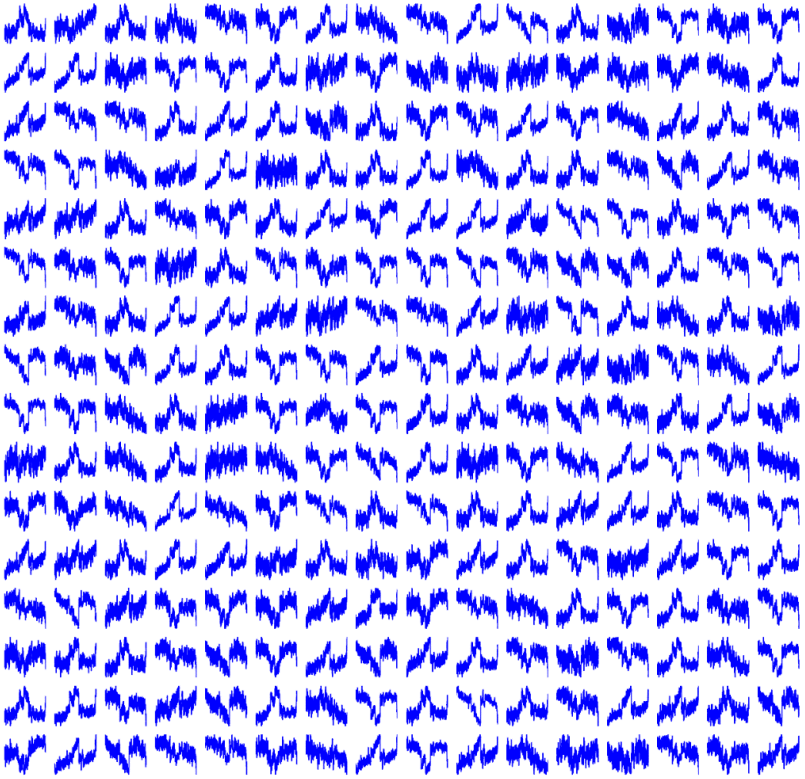
97.687%

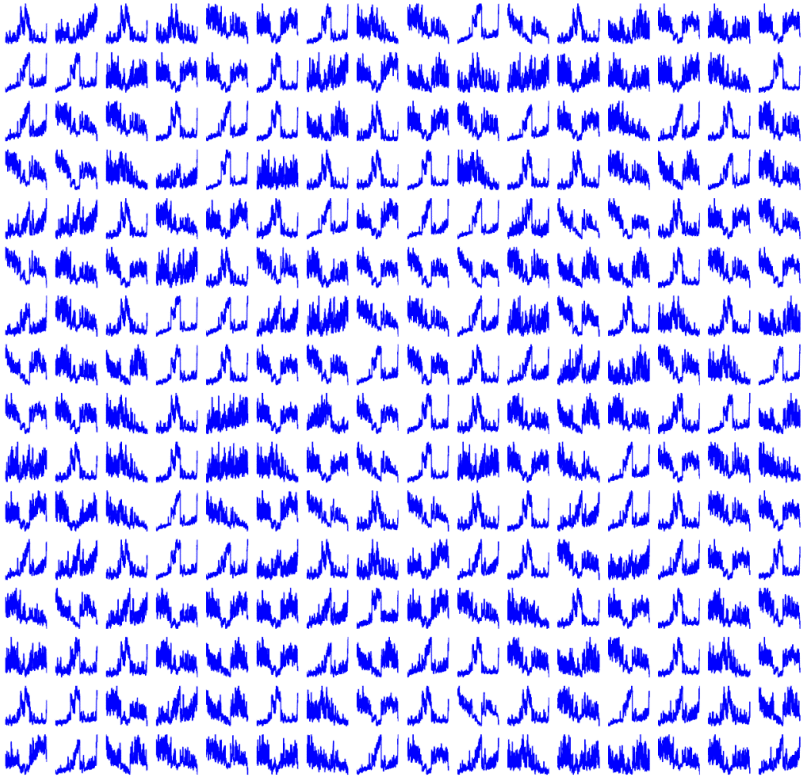
256 1x9 filters

Example: Hand Start

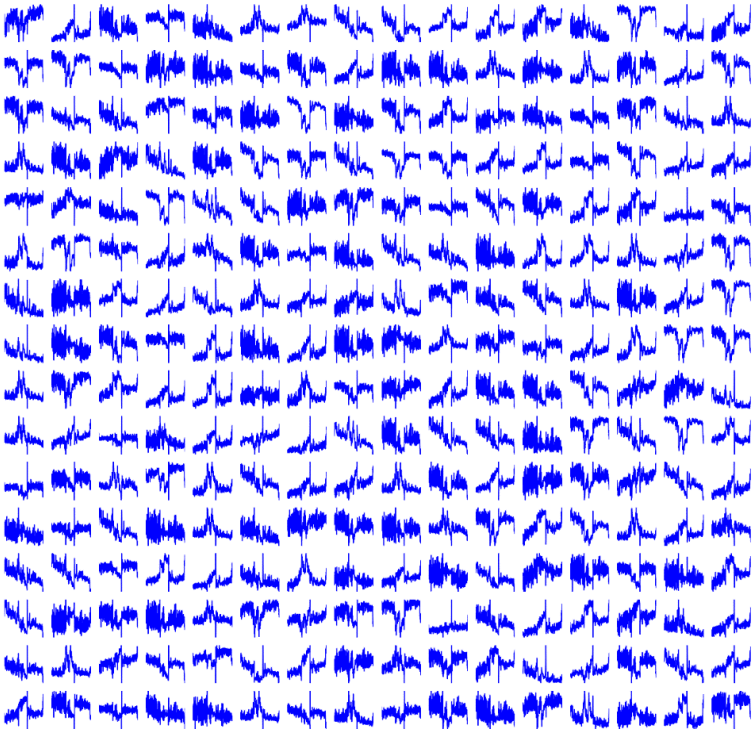


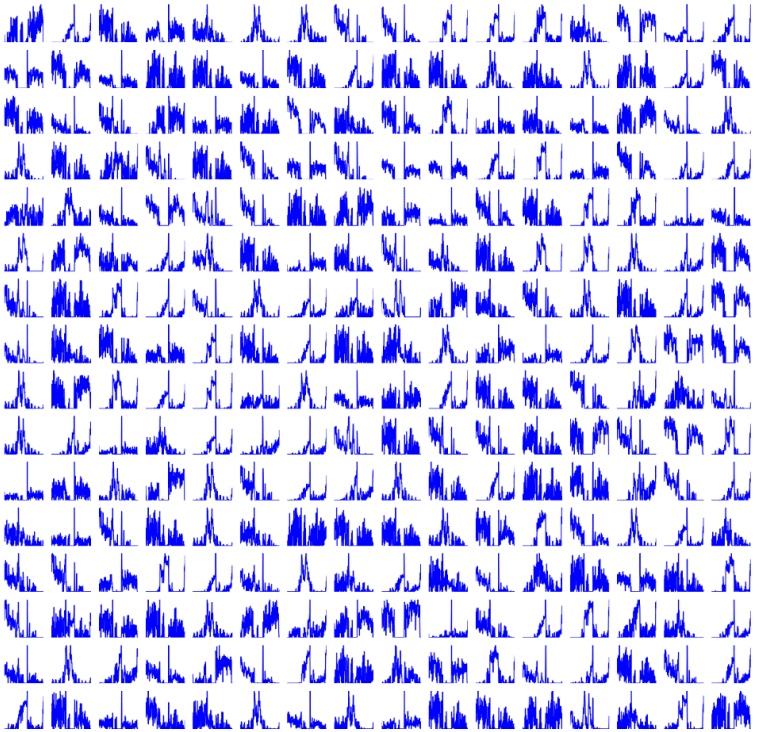
Example: Hand Start



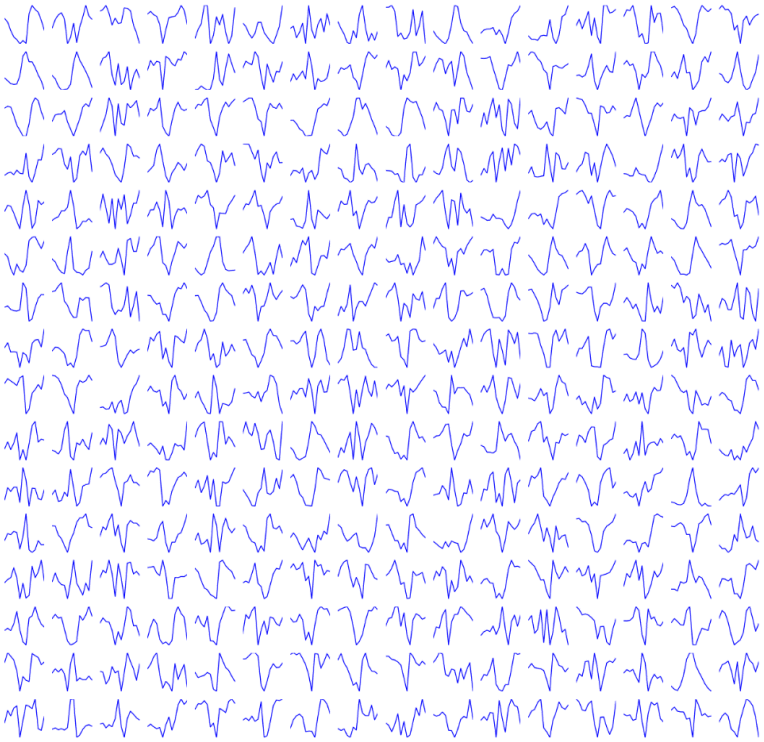


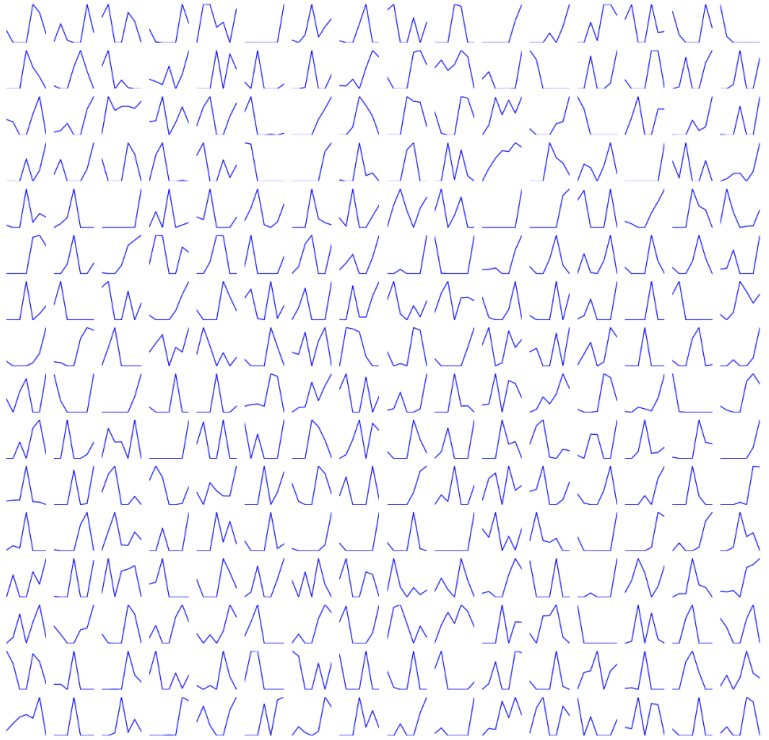
Example: Hand Start



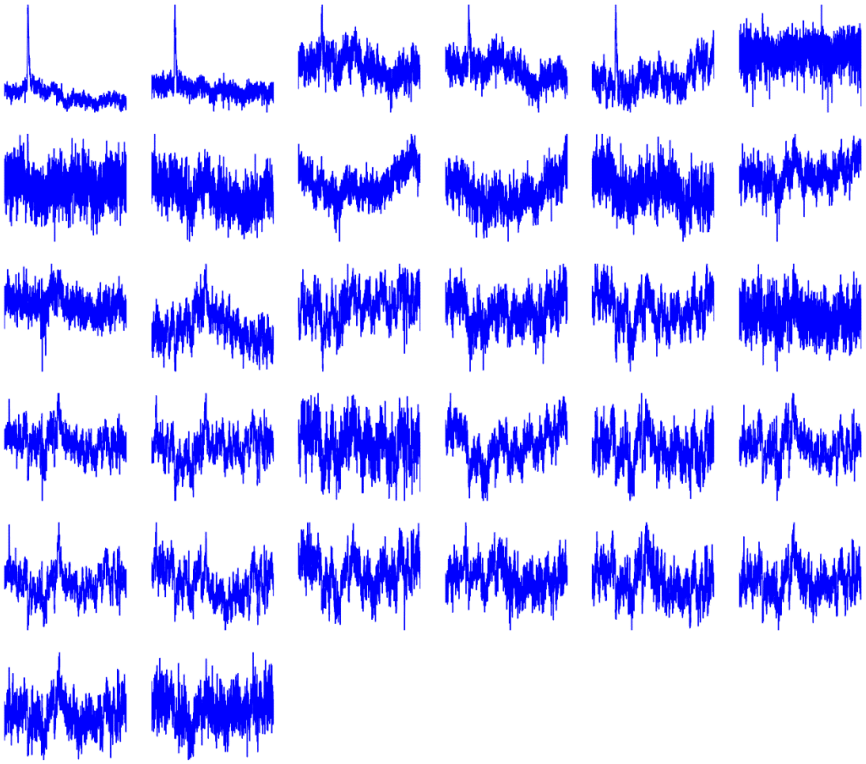


Example: Hand Start

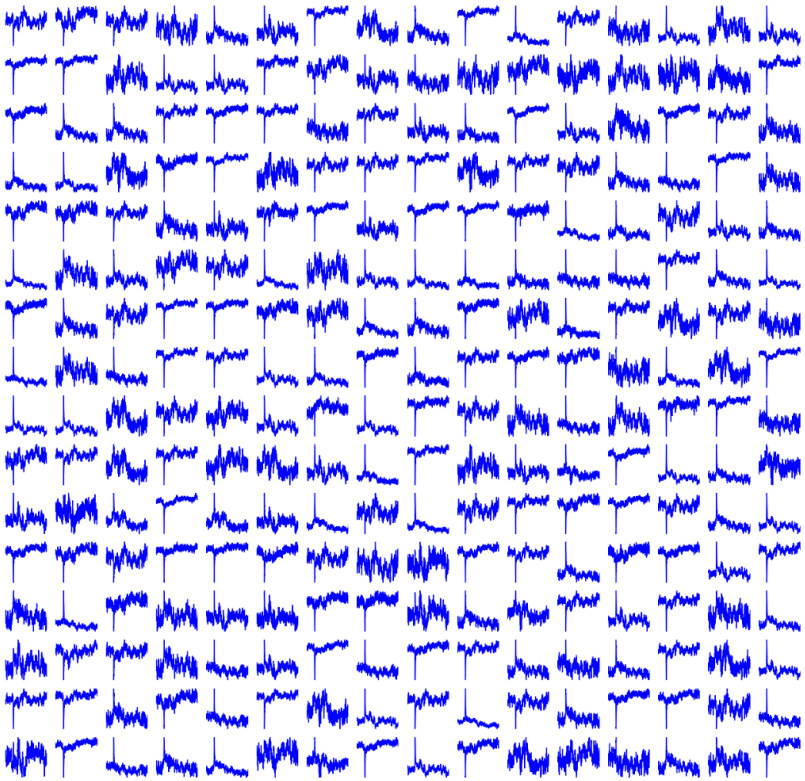


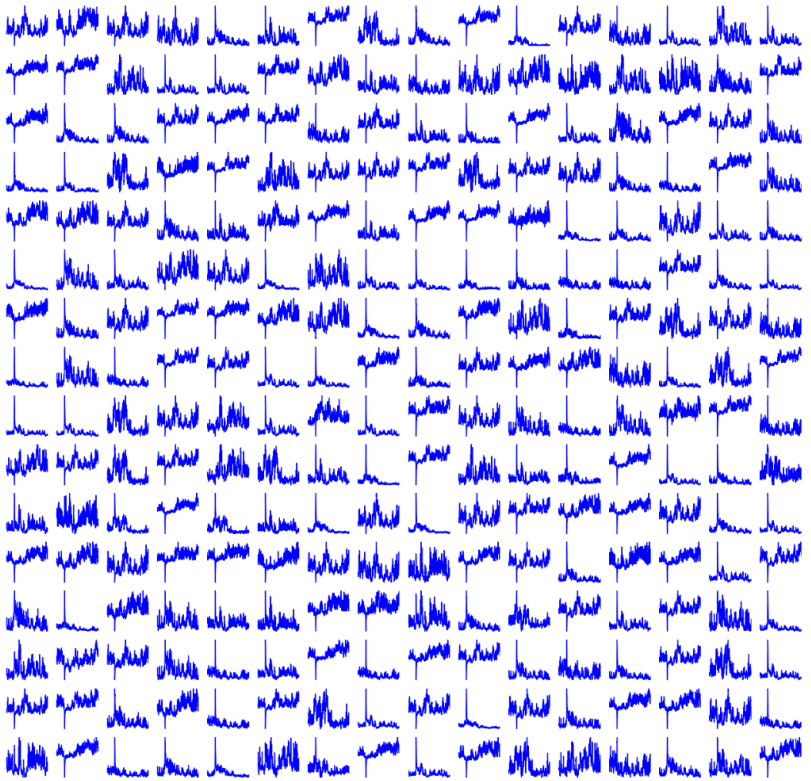


Example: First Digit Touch

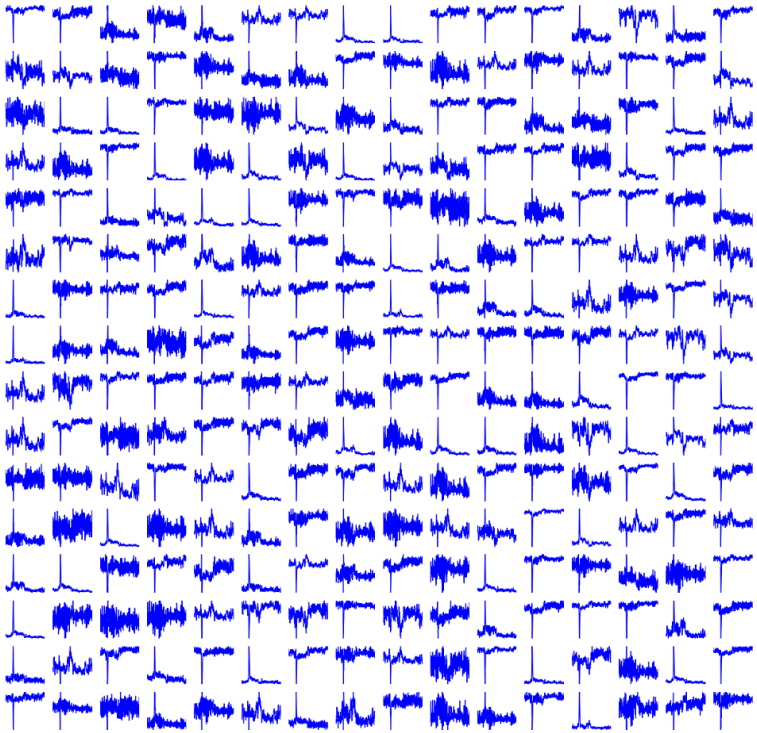


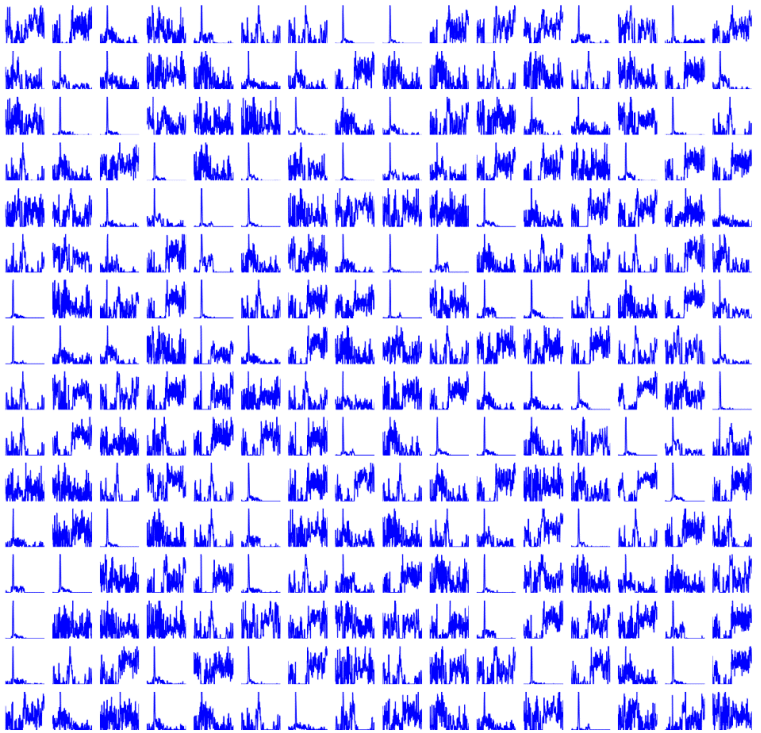
Example: First Digit Touch



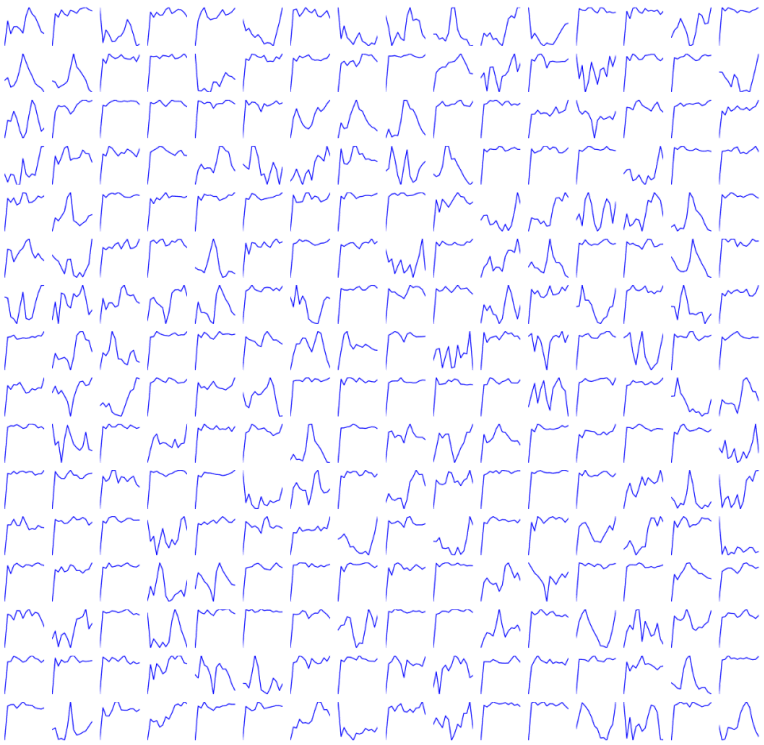


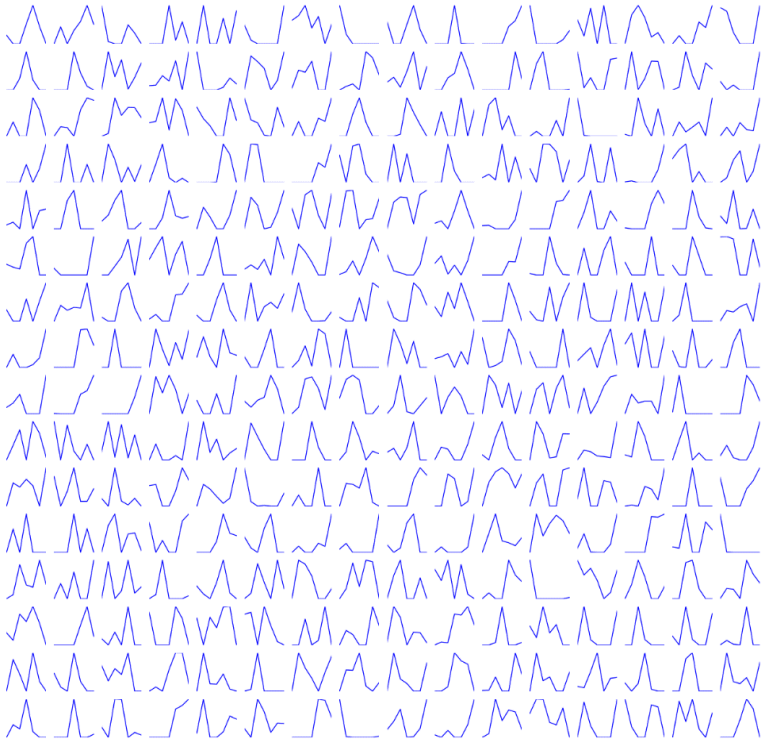
Example: First Digit Touch



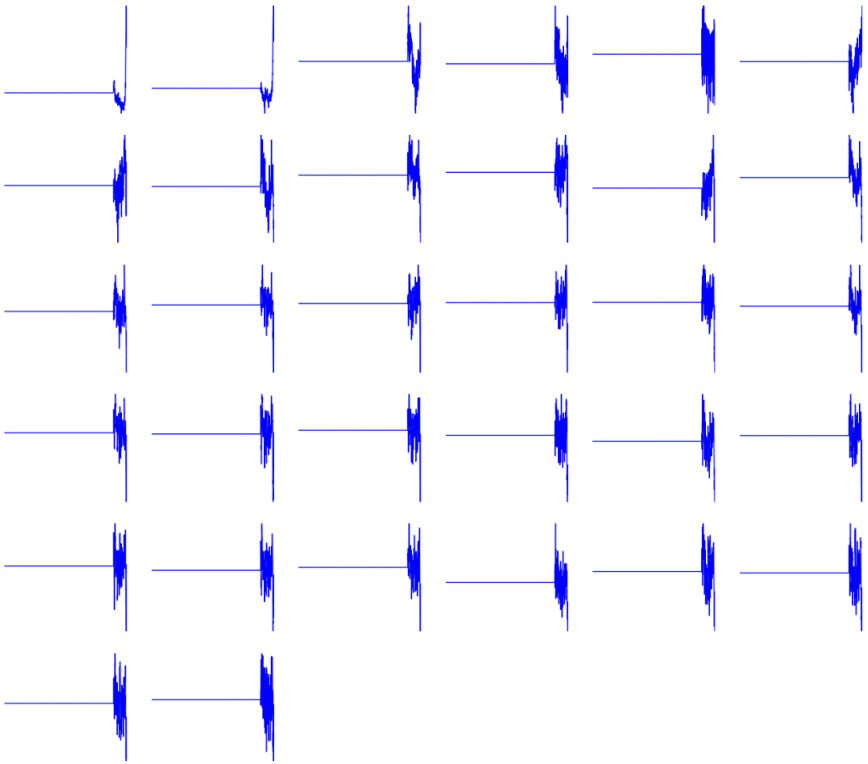


Example: First Digit Touch

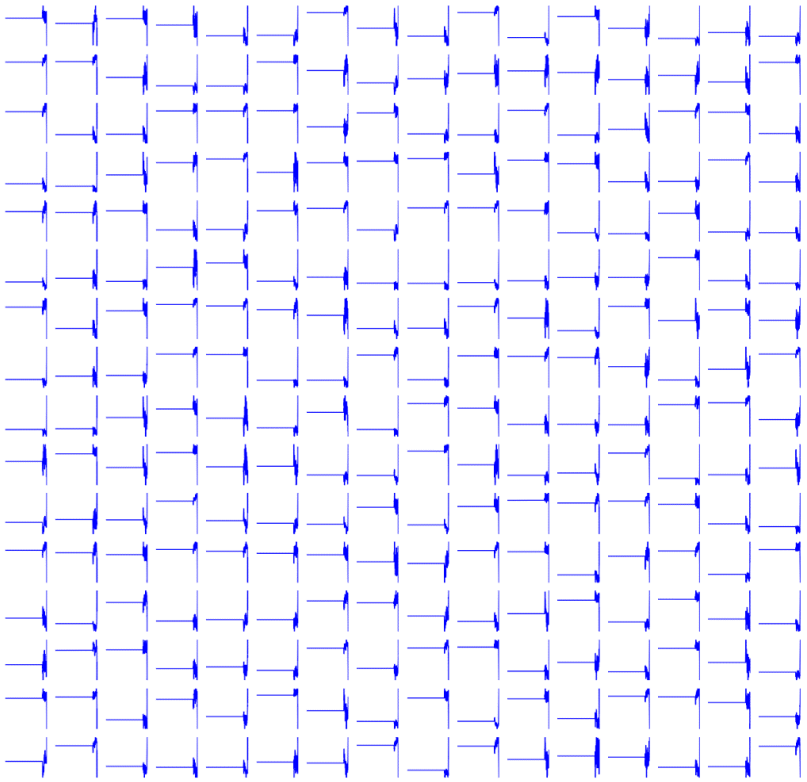


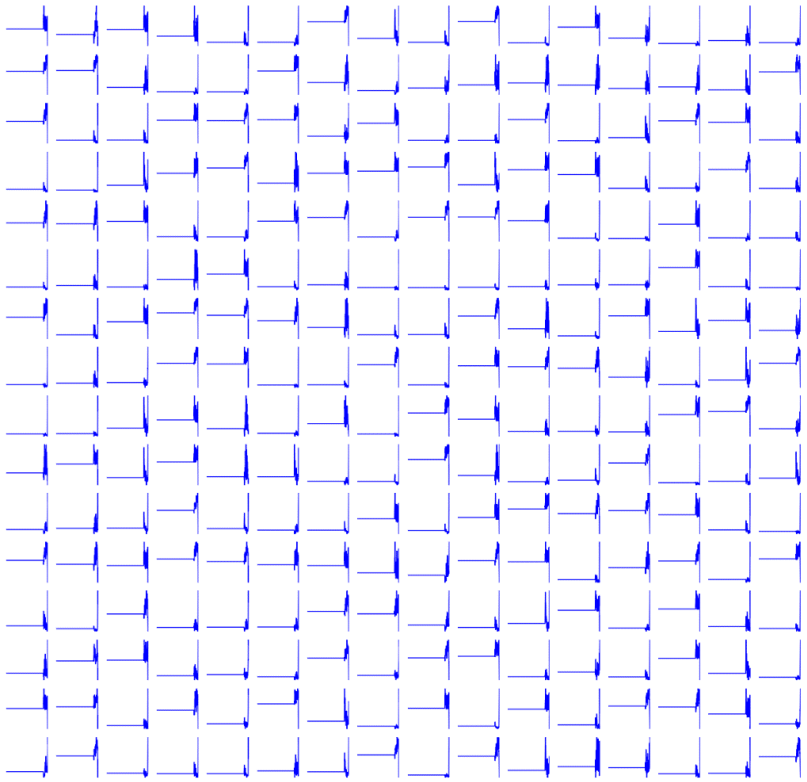


Example: Replace

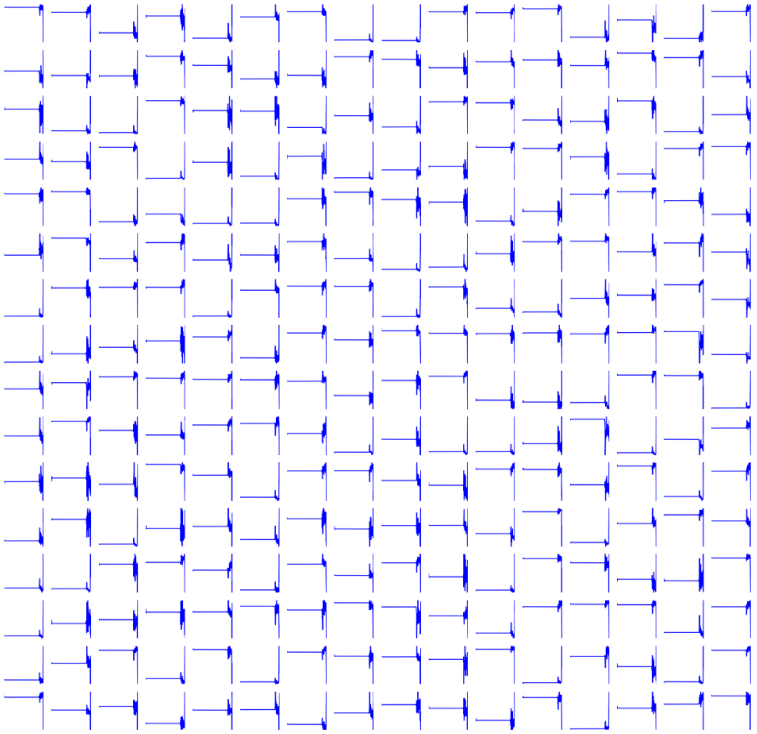


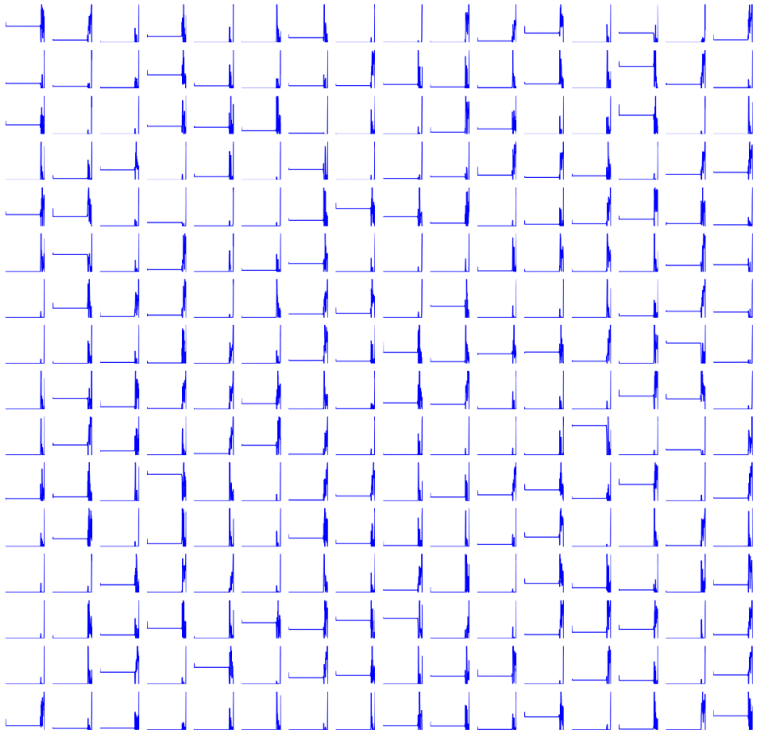
Example: Replace



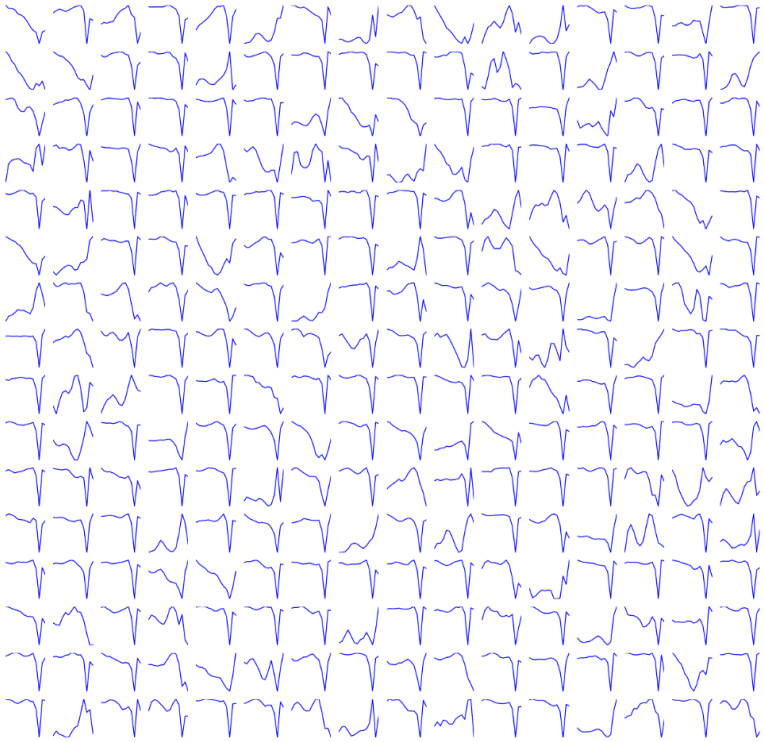


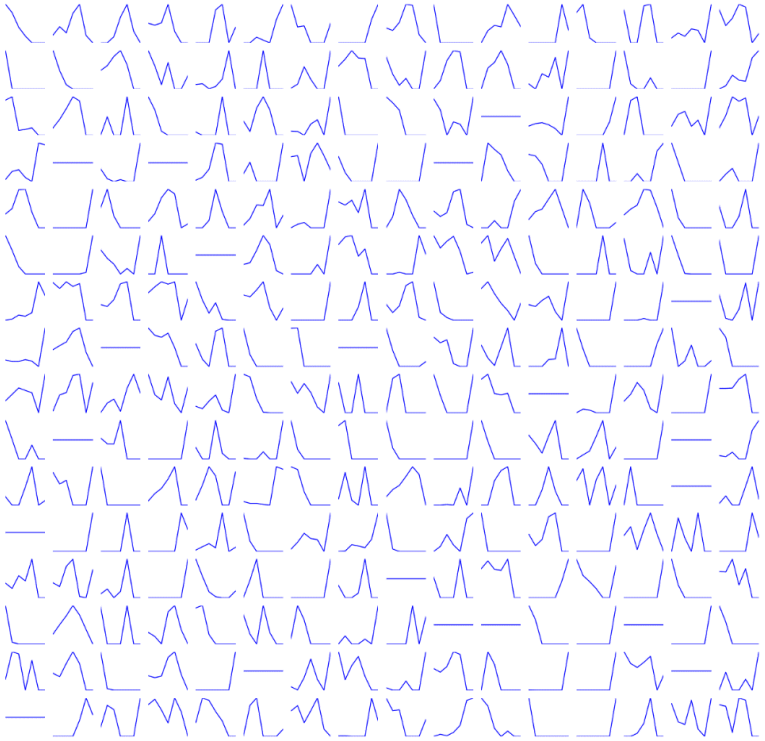
Example: Replace

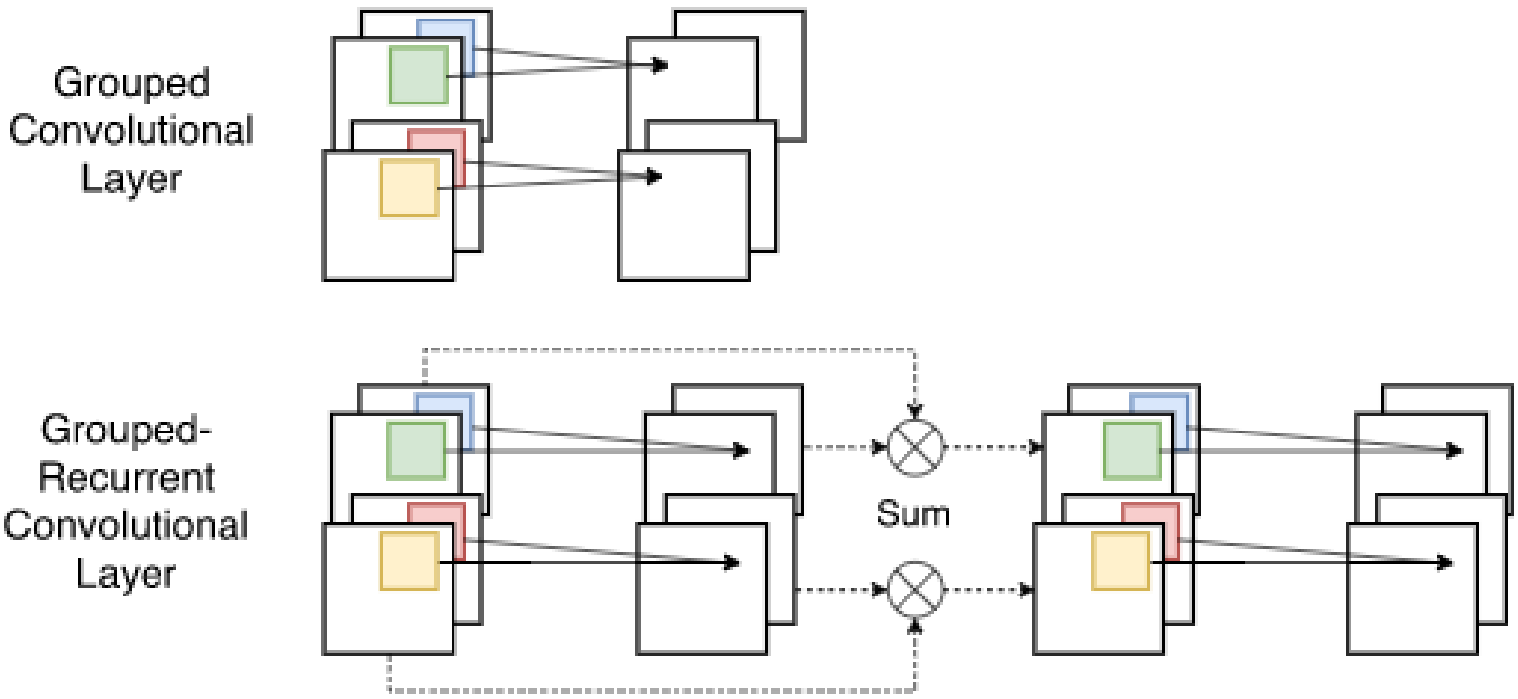




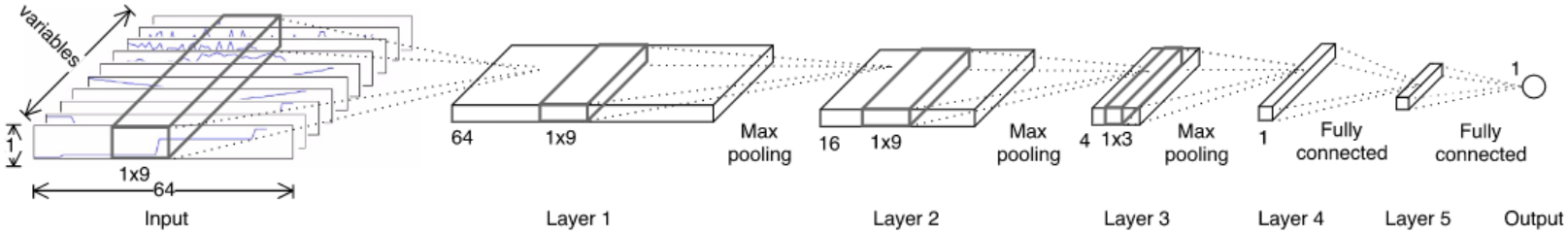
Example: Replace



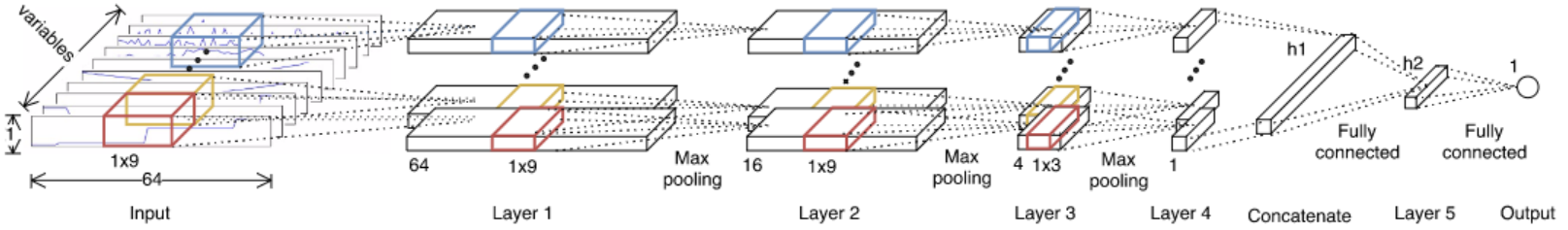




Grouped CNN/ Grouped RCNN
(Yi, Ju and Choi, 2017)



(a) Convolutional Neural Network (CNN).

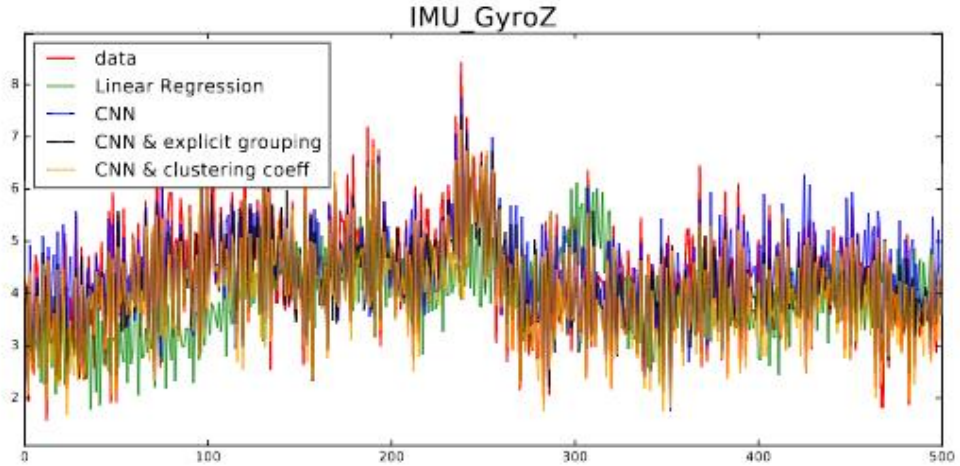


(b) CNN with grouped convolutional layers.

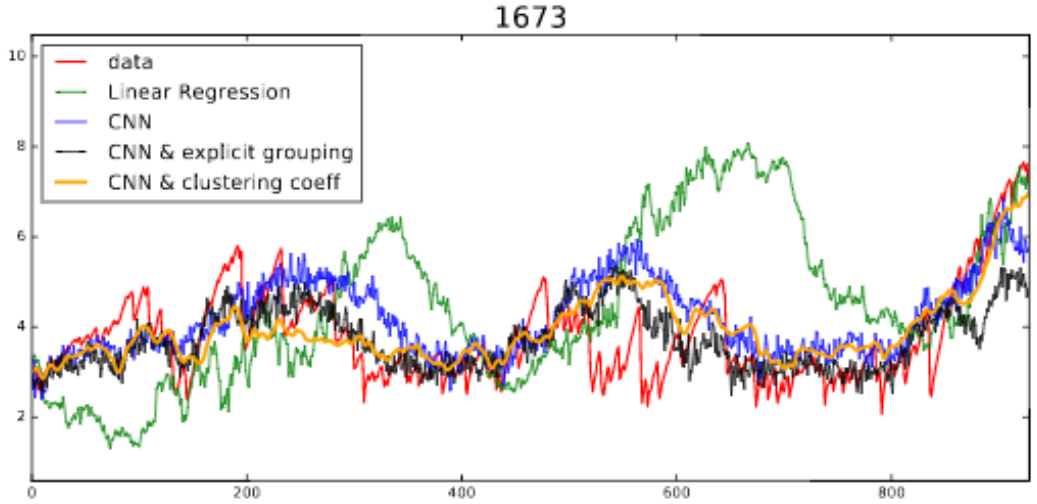
Grouped CNN/ Grouped RCNN (Yi, Ju and Choi, 2017)

(a) quadcopter used for the test.

(b) drone's flight path.



Collected from 148 sensors of 12,654 time steps



Collected from 88 sites for 28 years

Grouped CNN/ Grouped RCNN (Yi, Ju and Choi, 2017)

Model	Water	Drone
Linear Regression	1.298	0.690
Ridge Regression	1.298	0.731
LSTM	0.812	0.615
CNN	0.999	0.464
RCNN[1]	0.985	0.465
Group CNN	0.861	0.479
Group CNN (soft)	0.783	0.460
Group RCNN[9]	**0.775**	**0.438**

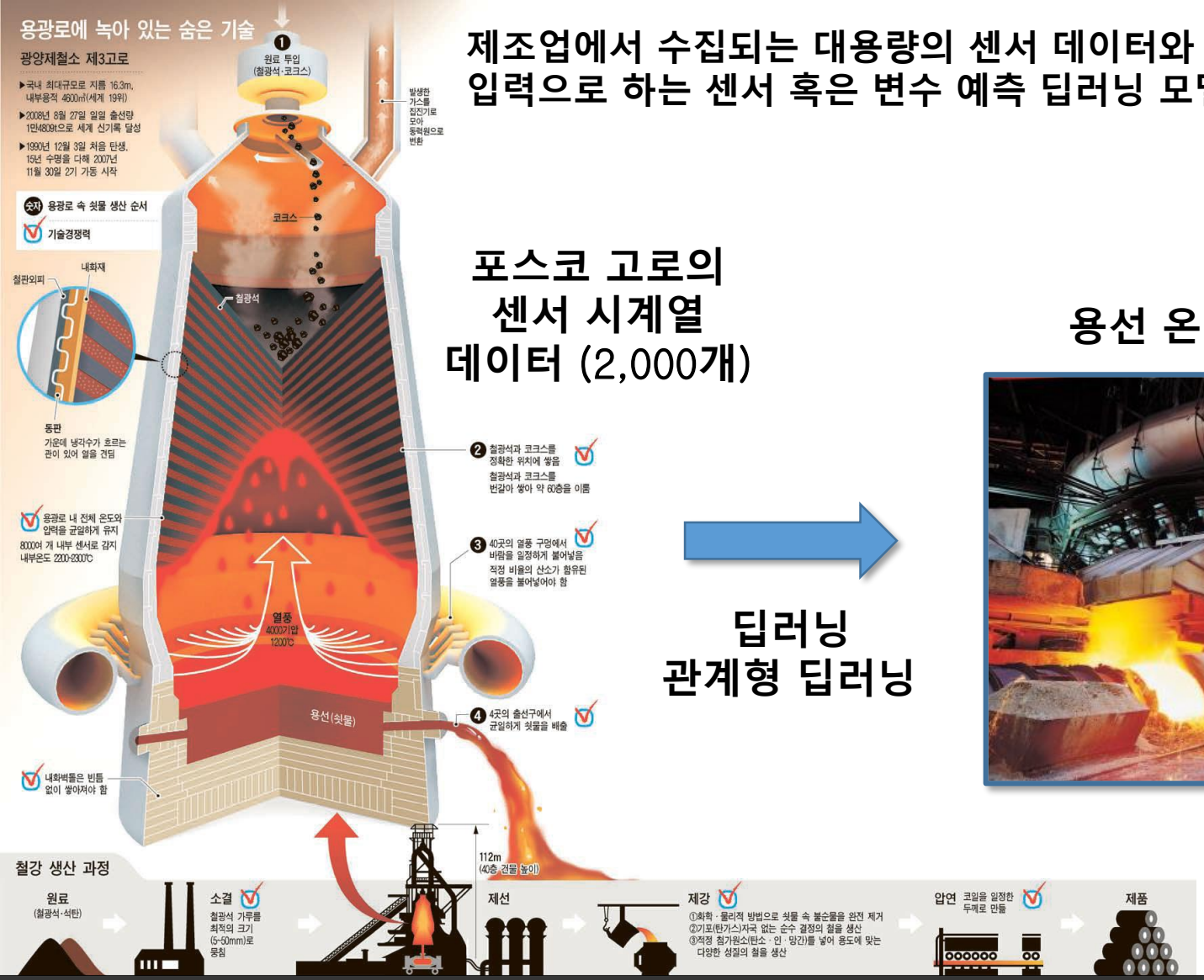
Dataset: US Groundwater
(Yi, Ju and Choi, 2017)

400 Time Series Data

Deep Learning

Temperature prediction

Complex Hybrid System - Manufacturing



제조업에서 수집되는 대용량의 센서 데이터와 파생 변수를 입력으로 하는 센서 혹은 변수 예측 딥러닝 모델

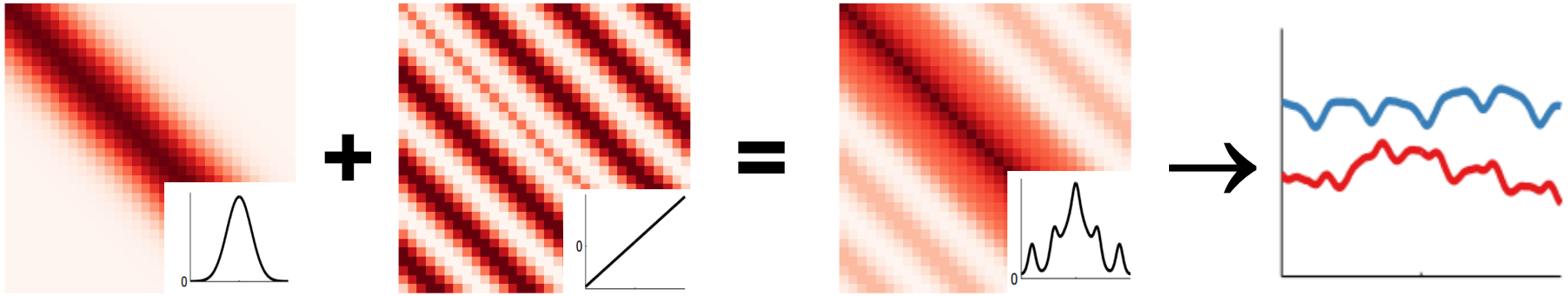
포스코 고로의 센서 시계열 데이터 (2,000개)

딥러닝
관계형 딥러닝

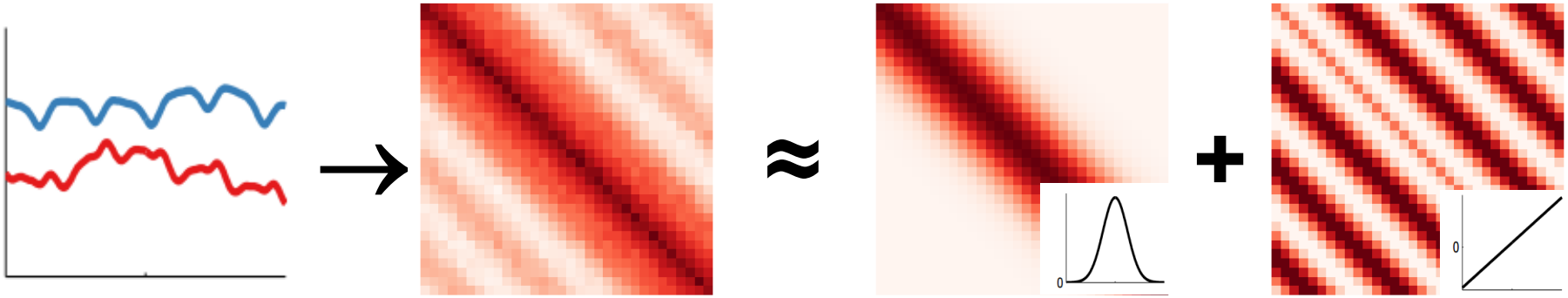
용선 온도 변화 예측

인공지능 기술을 활용한 제조 경쟁력 향상
군집 딥러닝(Group CNN, UNIST 2017)

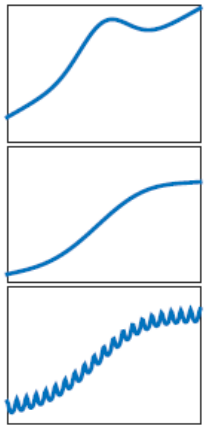
Kernel Composition: Generate Data from Models



Covariance Decomposition: Learn Explainable Models from Data



Kernel Composition & Covariance Decomposition

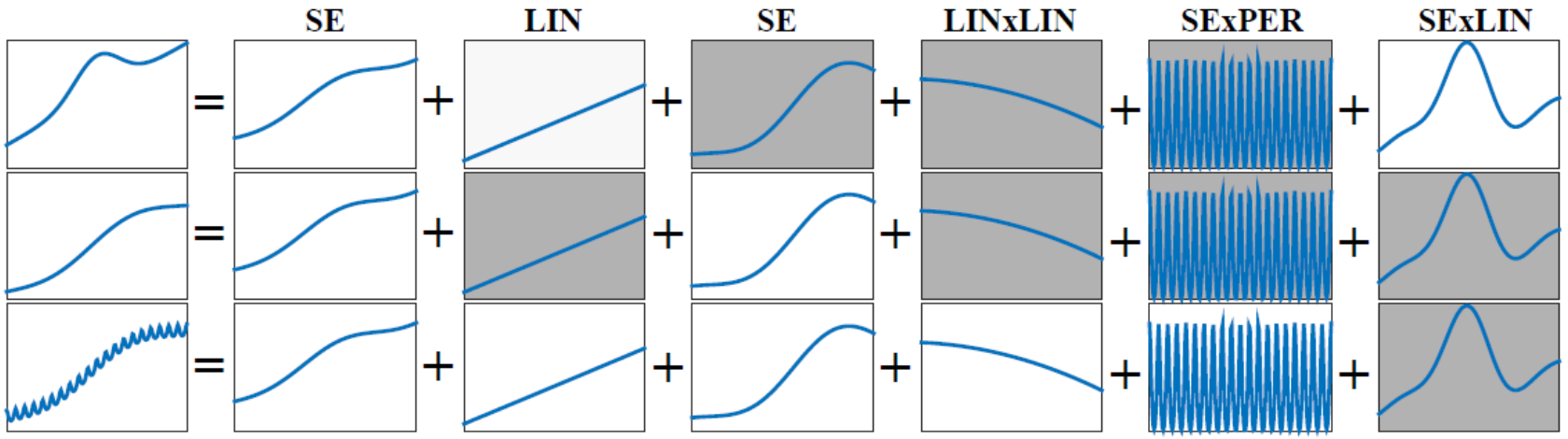


Challenges: Selective Kernel Search

Q: Can we selectively search over time series?

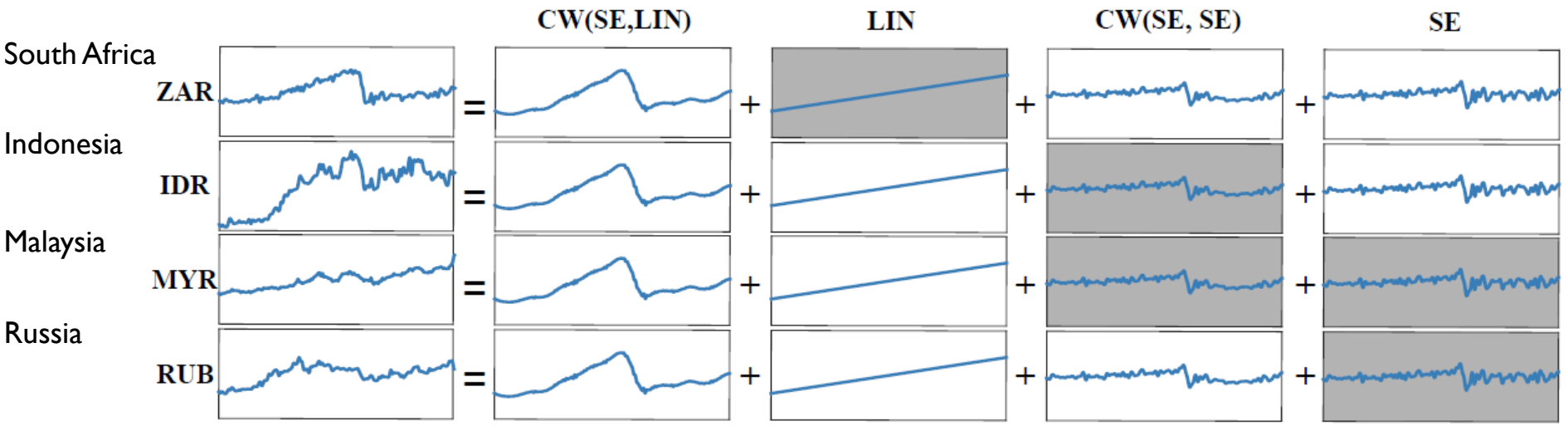
Indian Buffet Processes (IBP) + Gaussian Processes

(Nonparametric Clustering) **(Nonparametric Regression)**



Explainable Nonparametric Clustering and Regression

[Tong; Choi. Arxiv. 2017]



South African Rand and Indonesian Rupiah and Malaysian Ringgit and Russian Rouble share the following properties

→ This component is a smooth function with a typical lengthscale of 6.4 days. This component applies until Sep. 15th 2015 and from Sep. 17th 2015 onwards.

Indonesian Rupiah and Malaysian Ringgit and Russian Rouble share the following properties

→ This component is linearly increasing.

Explainable Regression with Nonparametric Clustering

[Tong; Choi. Arxiv. 2017]

	3 stocks	6 stocks	9 stocks	2 houses	4 houses	6 houses	4 currencies
Spike and Slab	1.80	2.81	10.26	13.95	8.59	11.33	151.37
GPRN	0.49	3.76	6.30	12.96	11.07	8.84	117.05
ABCD	0.40	3.69	8.35	6.58	5.84	7.96	330.00
RABCD	0.38	**1.22**	4.85	2.75	2.22	3.10	210.56
SM	**0.35**	1.43	**4.14**	5.72	3.13	4.26	158.17
All-in-one Search	**0.35**	1.84	6.35	5.07	4.87	7.30	**59.40**
Selective Search	**0.36**	1.47	4.82	**2.15**	**2.08**	**2.28**	81.98

IBP + GP methods

Explainable Regression with Nonparametric Clustering

[Tong; Choi. Arxiv. 2017]

Adobe beats Street 3Q forecasts

Associated Press September 20, 2017

SAN JOSE, Calif. (AP) _ Adobe Systems Inc. (ADBE) on Tuesday reported fiscal third-quarter profit of $419.6 million.

The San Jose, California-based company said it had profit of 84 cents per share. Earnings, adjusted for one-time gains and costs, were $1.10 per share.

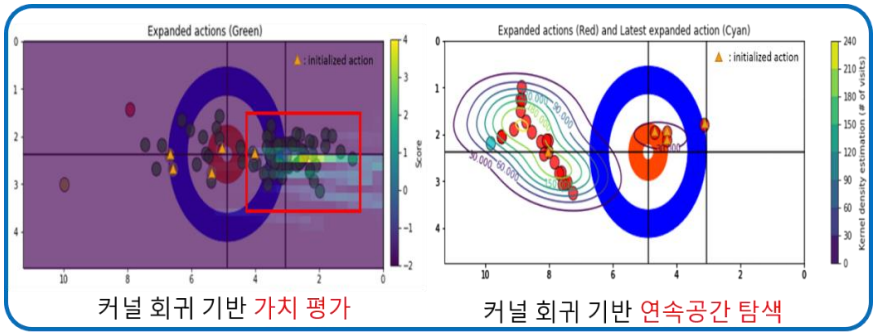
…

Adobe shares have climbed 52 percent since the beginning of the year. In the final minutes of trading on Tuesday, shares hit $156.61, an increase of 57 percent in the last 12 months.

…

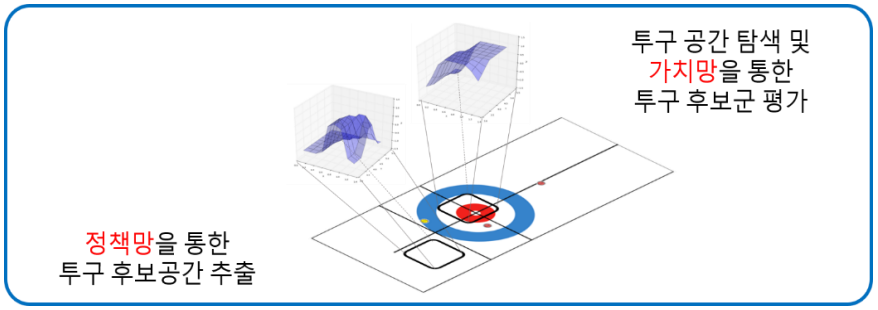
In coming months, Adobe shares are expected to increase since Adobe is one of selected Silicon-based SW companies showing linearly increase trends.

This story was generated by Automated Insights and Relational Automatic Statistician.

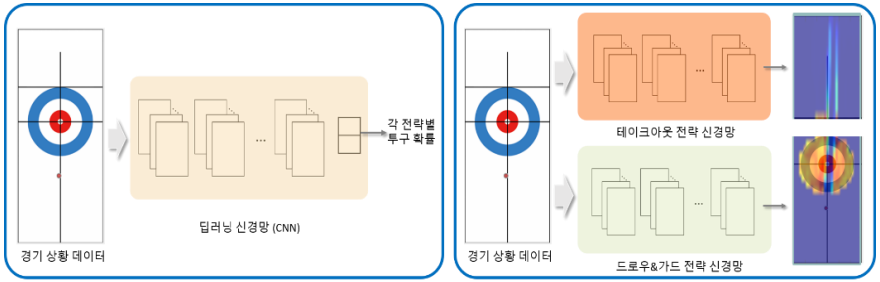


커널 회귀 기반 가치 평가 커널 회귀 기반 연속공간 탐색

커널 회귀 기반 몬테카를로 트리 탐색



정책망과 가치망을 통한 전략 추천

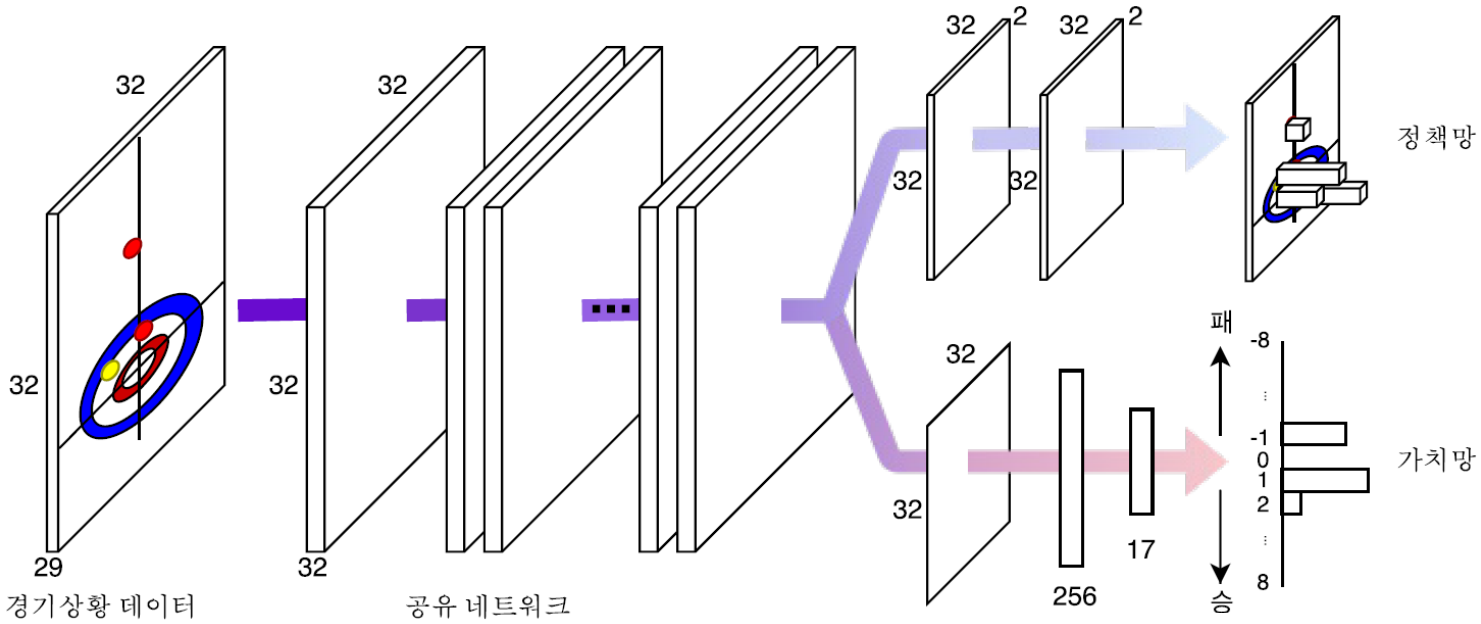


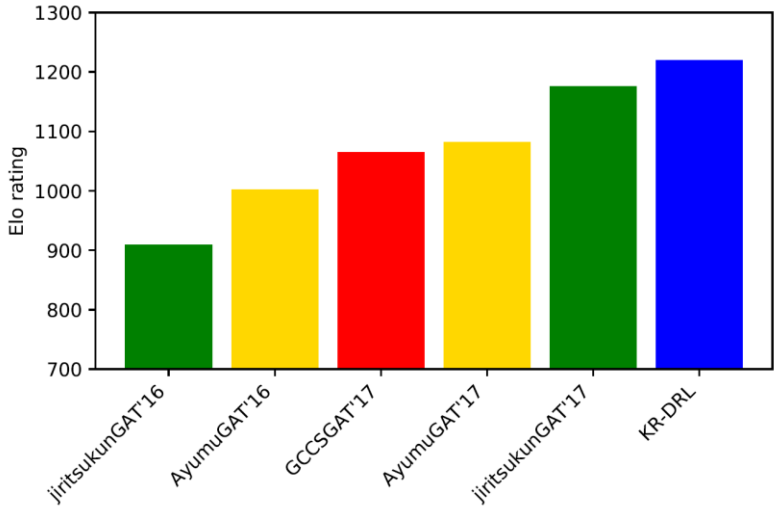
경기 상황별 전략 예측 신경망

			1st	2nd	3rd	4th		
1	Parthia	益子直(東北大) / Sunao Mashiko(Tohoku Univ.)	6 10	5 9/h	5 2/h	11 3/h	3-1	2nd
2	CurlGo_R	Hong-Bok Lee and Tae-San Eom(Korea Univ.)	5 9	9 4/h	2 1	6 5/h	2-2	5th
3	Kim Byungdo Bento	Human	8 8	3 7/h	7 10/h	7 1	1-3	7th
4	Kumhau	松井亮平(電通大) / Ryohei Matsui(UEC)	1 7	4 2	3 5/h	3 10/h	0-4	10th
5	Tetsuro Tanaka and Takeshi Ito	Human	6 6	4 10/h	14 4	5 2	1-3	7th
6	CSRL	Kyowoon Lee(UNIST)	15 5/h	13 8	11 9	4 7/h	3-1	2nd
7	CSVP	Jaesik Choi(UNIST)	17 4/h	8 3	8 8/h	8 6	4-0	1st
8	CurlGo_D	Sang-Hun Lee(Korea Univ.)	14 3/h	6 6/h	6 7	5 9/h	1-3	7th
9	じりつくん / Jiritsukun	加藤修(北大) / Shu Kato(Hokkaido Univ.)	7 2/h	7 1	5 6/h	6 8	3-1	2nd
10	Eom Tae San	Human	5 1/h	6 5	6 3	13 4	2-2	5th

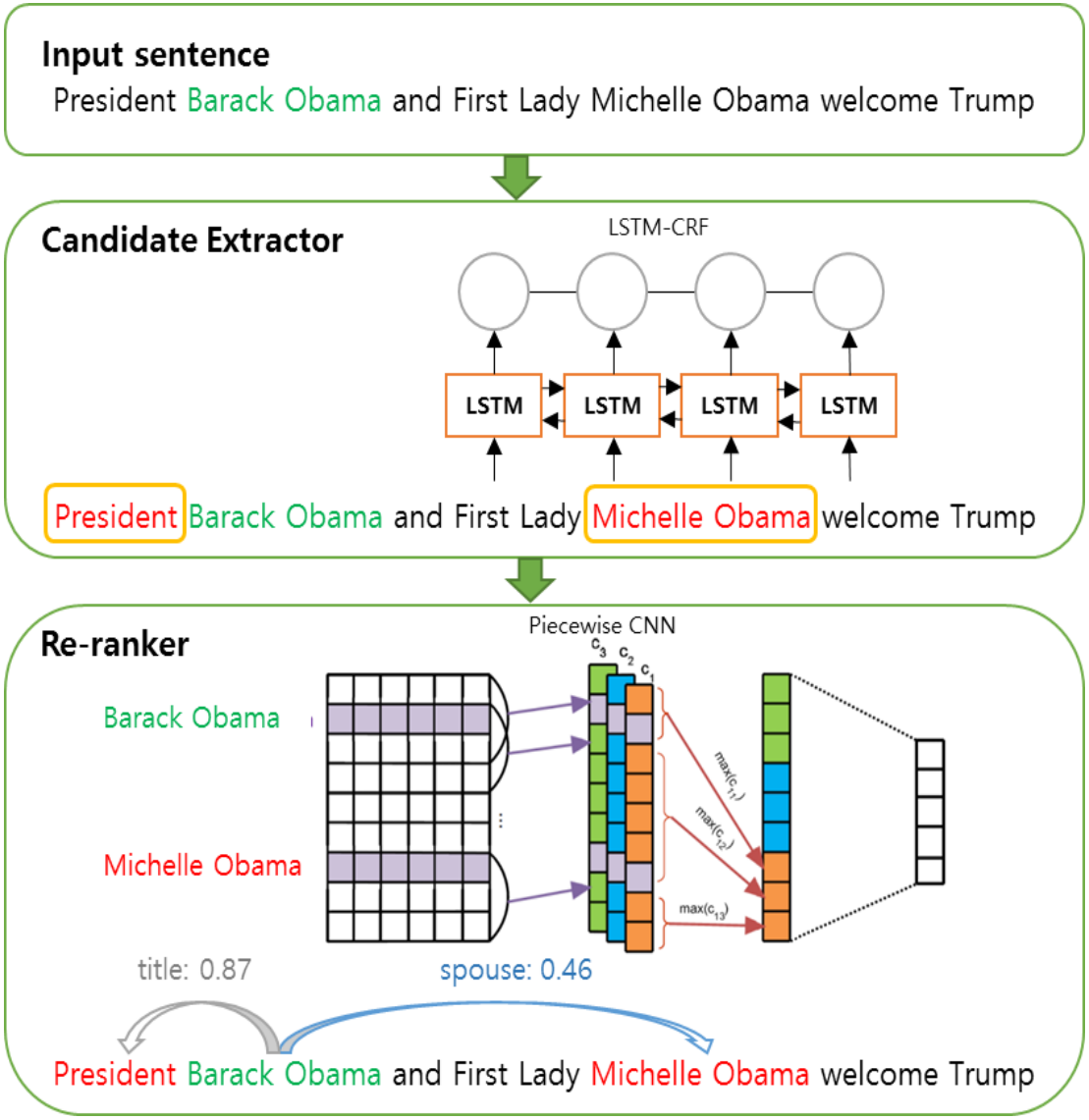
GPW 2017 (Game programing workshop)
Digital curling 부분 우승

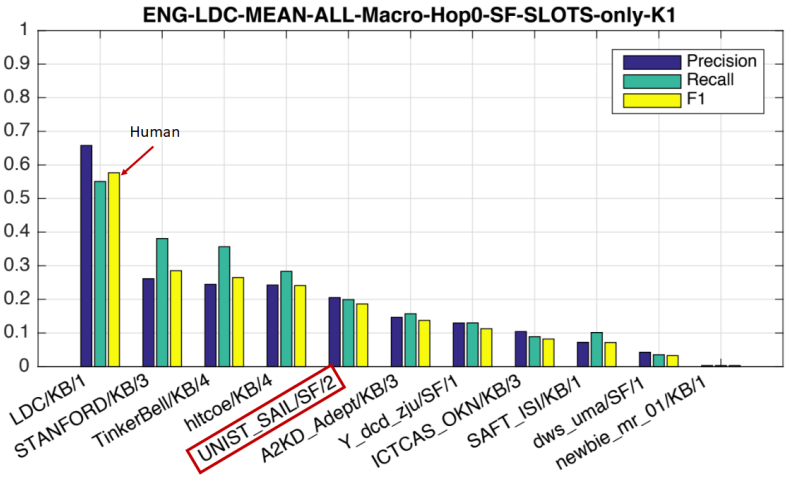
딥러닝 기반 연속 공간 상의 컬링 전략 추천 (UNIST 2017)





딥러닝 기반 연속 공간 상의 컬링 전략 추천 (UNIST 2017)





- **개체 (Entity) 사이의 관계 유추**
  - LSTM-CRF 기반 정답 후보 추출 모듈
  - Piecewise-CNN 기반 재순위화 모듈
- **정답 후보 추출 모델과 재순위화 모델 간 end2end 결합 학습으로 성능 향상**
- **TAC 2017 경진대회에서 4위**

딥러닝 기반 Slot-filling 시스템 (UNIST 2017)

Thank you

jaesik@unist.ac.kr