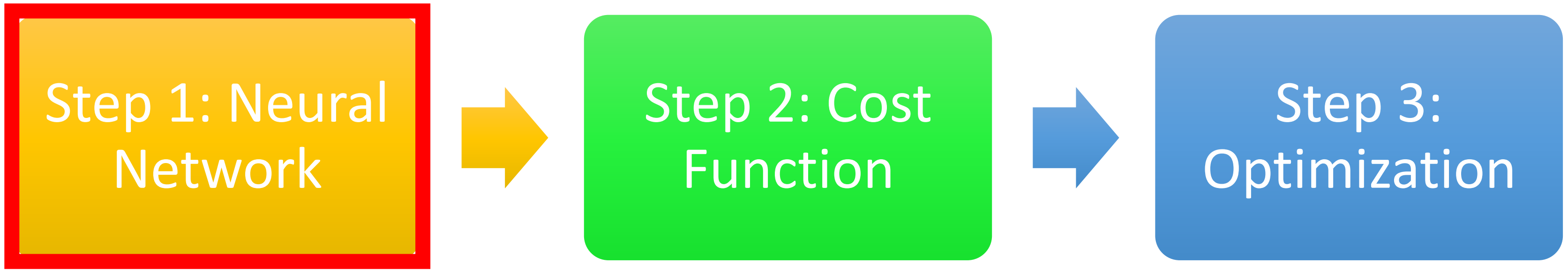
Network Structure

Hung-yi Lee

李宏毅

Three Steps for Deep Learning



Step 1. A neural network is a function composed of simple functions (neurons)

- ➢ Usually we design the network structure, and let machine find parameters from data

Step 2. Cost function evaluates how good a set of parameters is

- ➢ We design the cost function based on the task

Step 3. Find the best function set (e.g. gradient descent)

Outline

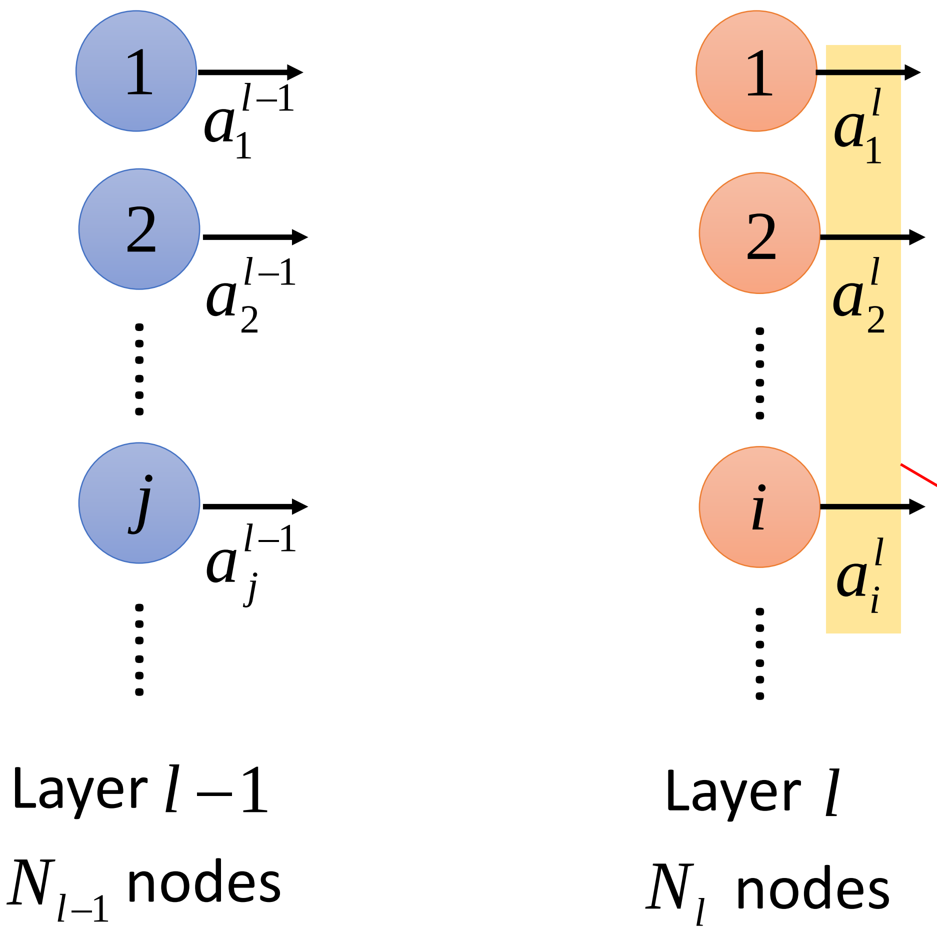
- Basic structure (3/03)
  - Fully Connected Layer
  - Recurrent Structure
  - Convolutional/Pooling Layer
- Special Structure (3/17)
  - Spatial Transformation Layer
  - Highway Network / Grid LSTM
  - Recursive Structure
  - Batch Normalization
  - Sequence-to-sequence / Attention (3/24)

Prerequisite

- Brief Introduction of Deep Learning
  - https://youtu.be/Dr-WRlEFefw?list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49
- Convolutional Neural Network
  - https://youtu.be/FrKWiRv254g?list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49
- Recurrent Neural Network (Part I)
  - https://youtu.be/xCGidAeyS4M?list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49
- Recurrent Neural Network (Part II)
  - https://www.youtube.com/watch?v=rTqmWlnwz_0&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=25

Basic Structure:
Fully Connected Layer

Fully Connected Layer



Output of a neuron:

$$a_i^l$$

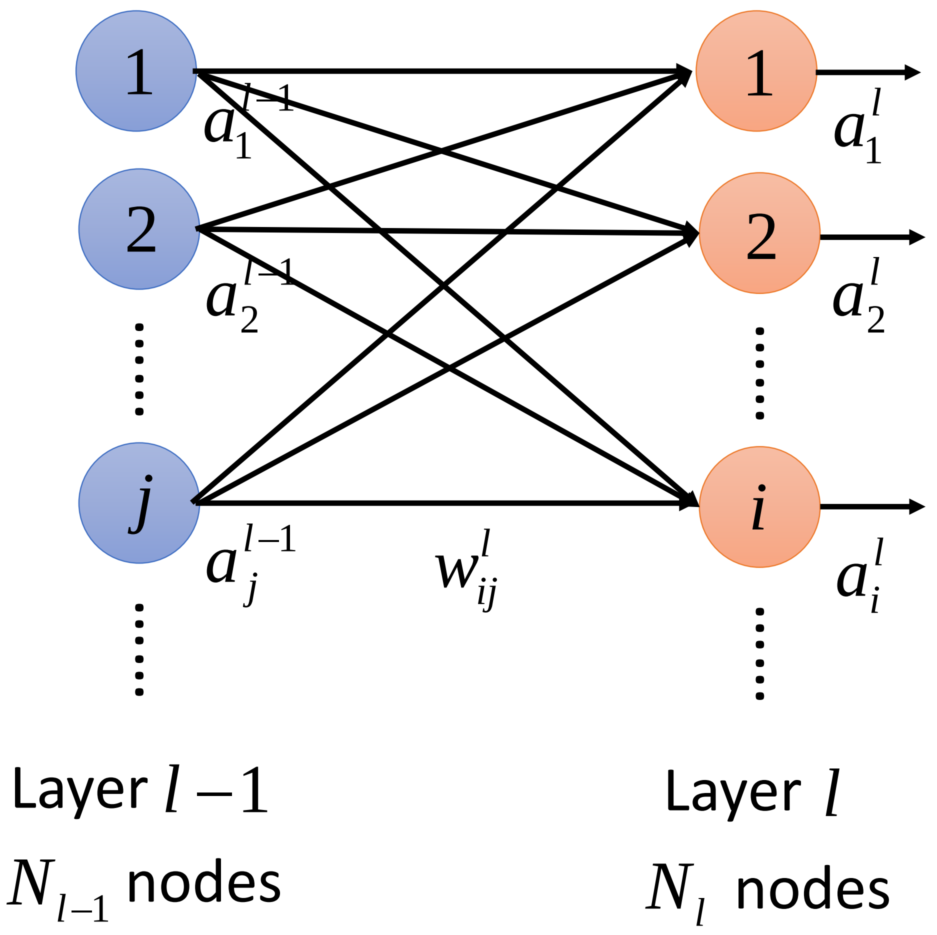
l → Layer l

i → Neuron i

Output of one layer:

a^l : a vector

Fully Connected Layer

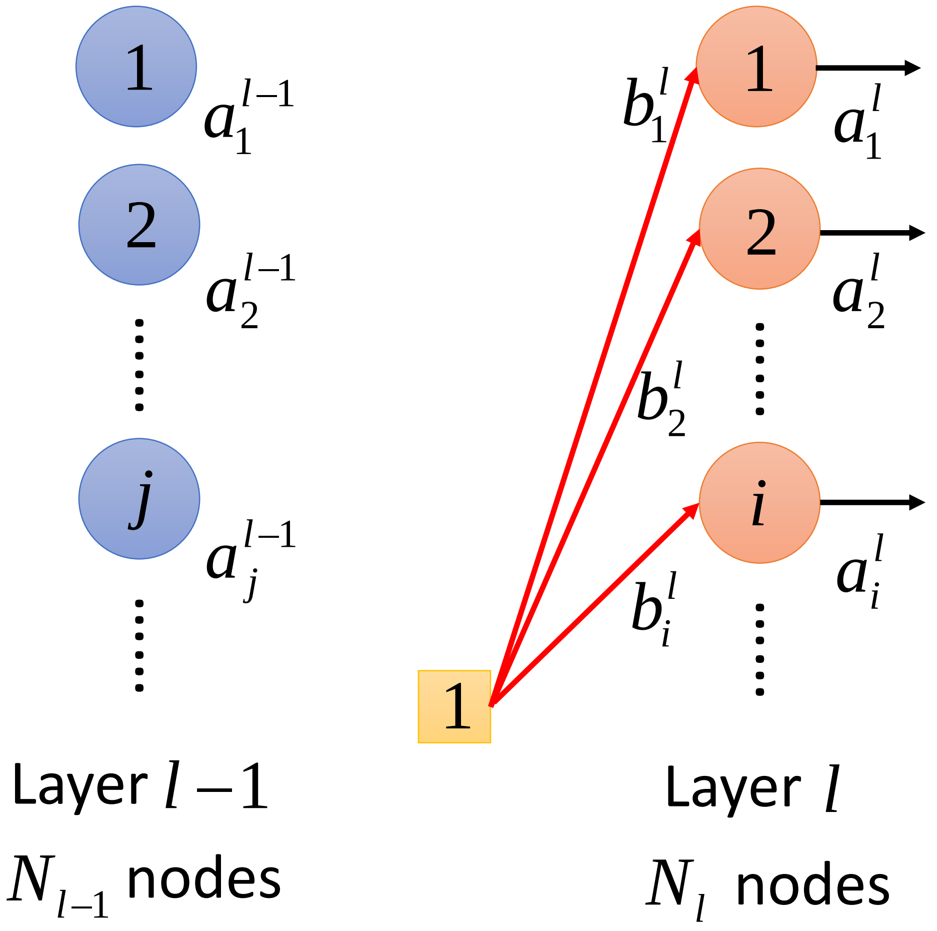


w_{ij}^l → Layer $l-1$ to Layer l

↓ from neuron j (Layer $l-1$) to neuron i (Layer l)

$$
\begin{array}{cc}
\overbrace{\hphantom{w_{11}^l \quad w_{12}^l \quad \cdots}}^{\mathrm{N}_{l-1}} & \\
W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix} & \left.\vphantom{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}\right\} \mathrm{N}_l
\end{array}
$$

Fully Connected Layer

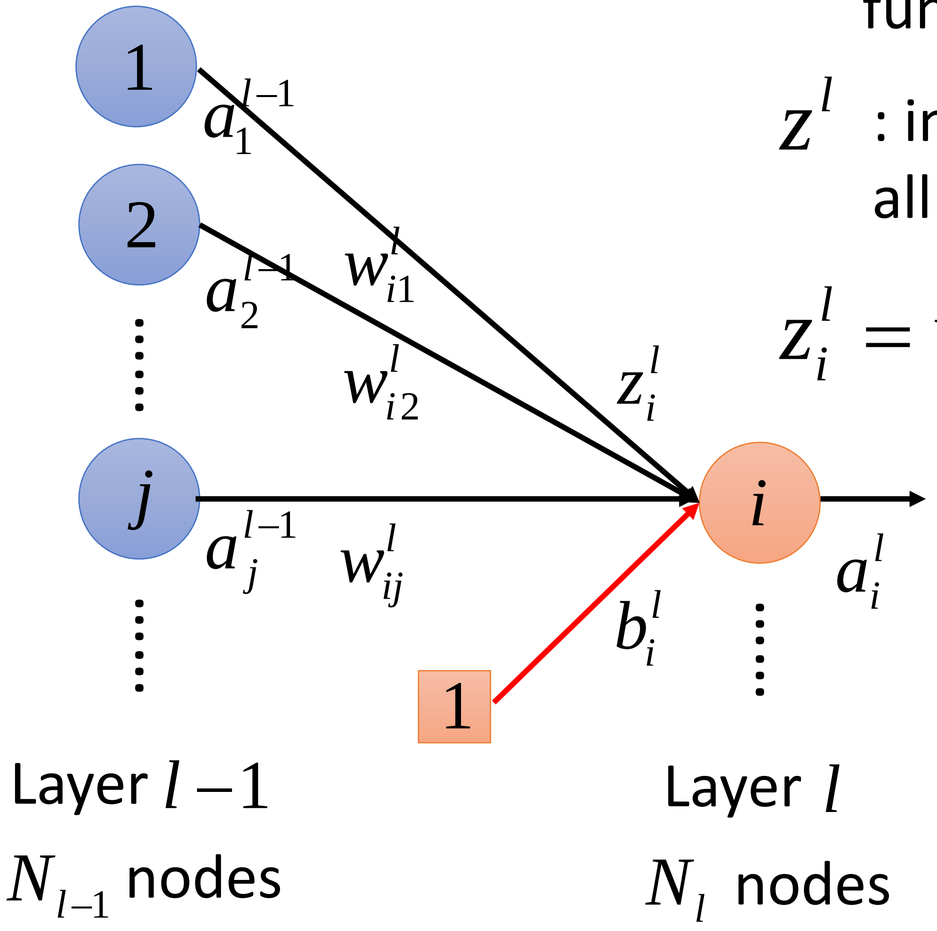


b_i^l : bias for neuron i at layer l

$$b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

bias for all neurons in layer l

Fully Connected Layer



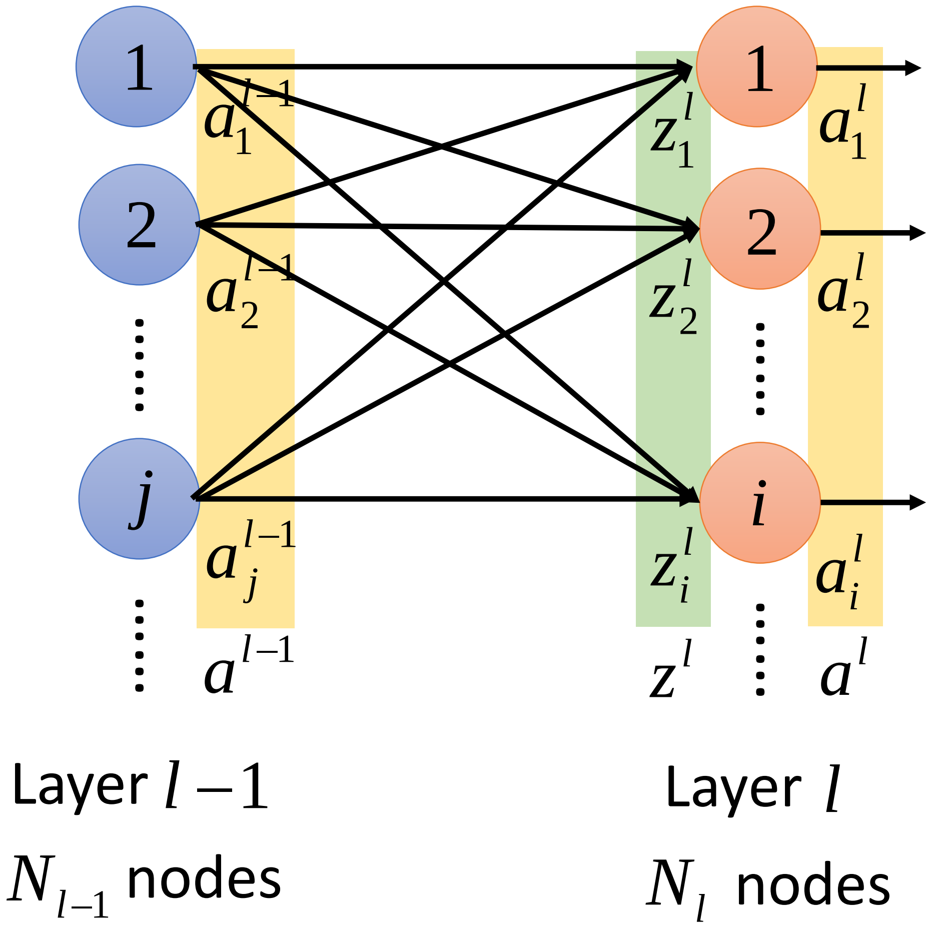
z_i^l : input of the activation function for neuron i at layer l

z^l : input of the activation function all the neurons in layer l

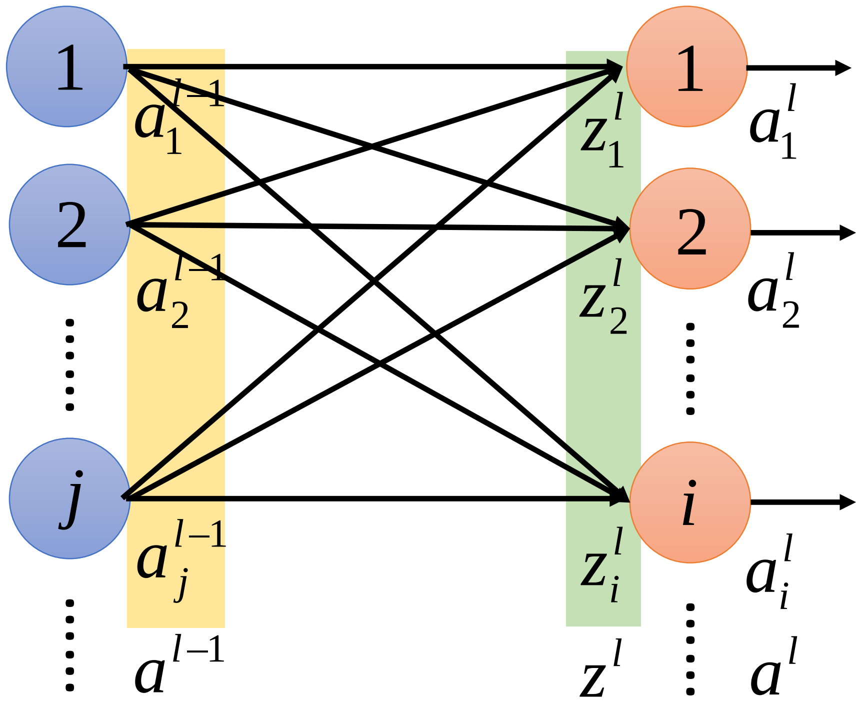
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

Relations between Layer Outputs



Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

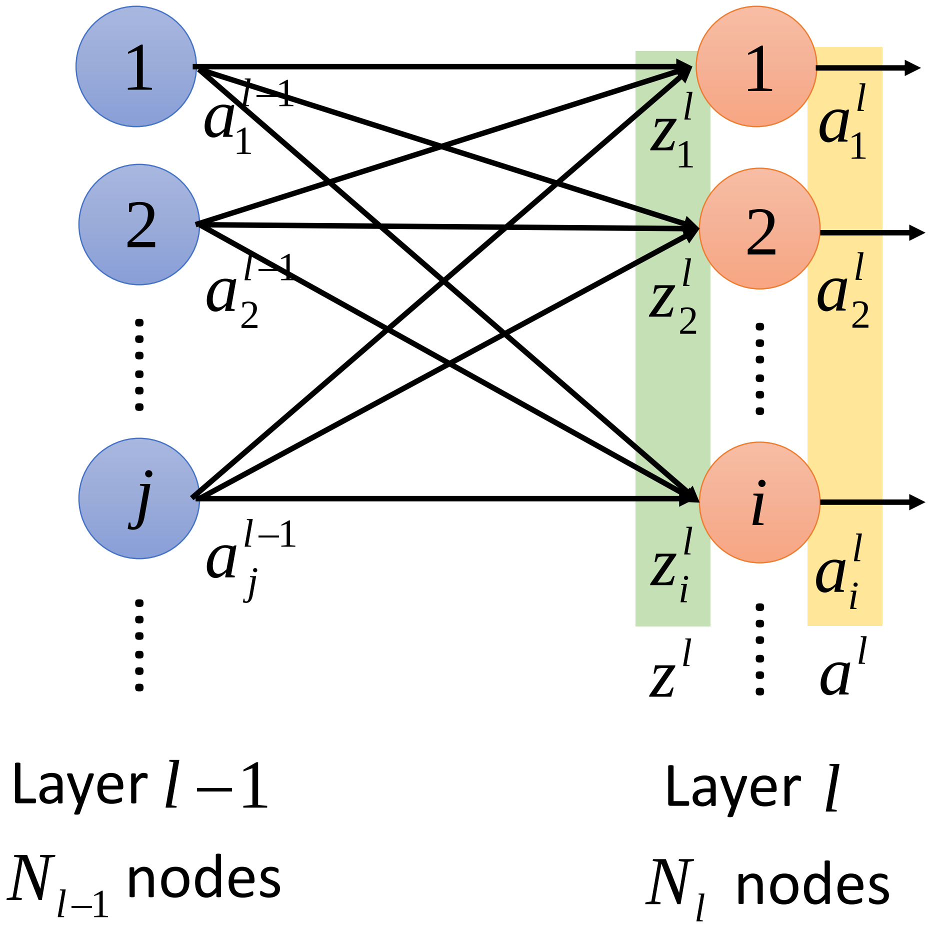
N_l nodes

$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$
$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$
$$\vdots$$
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$
$$\vdots$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs

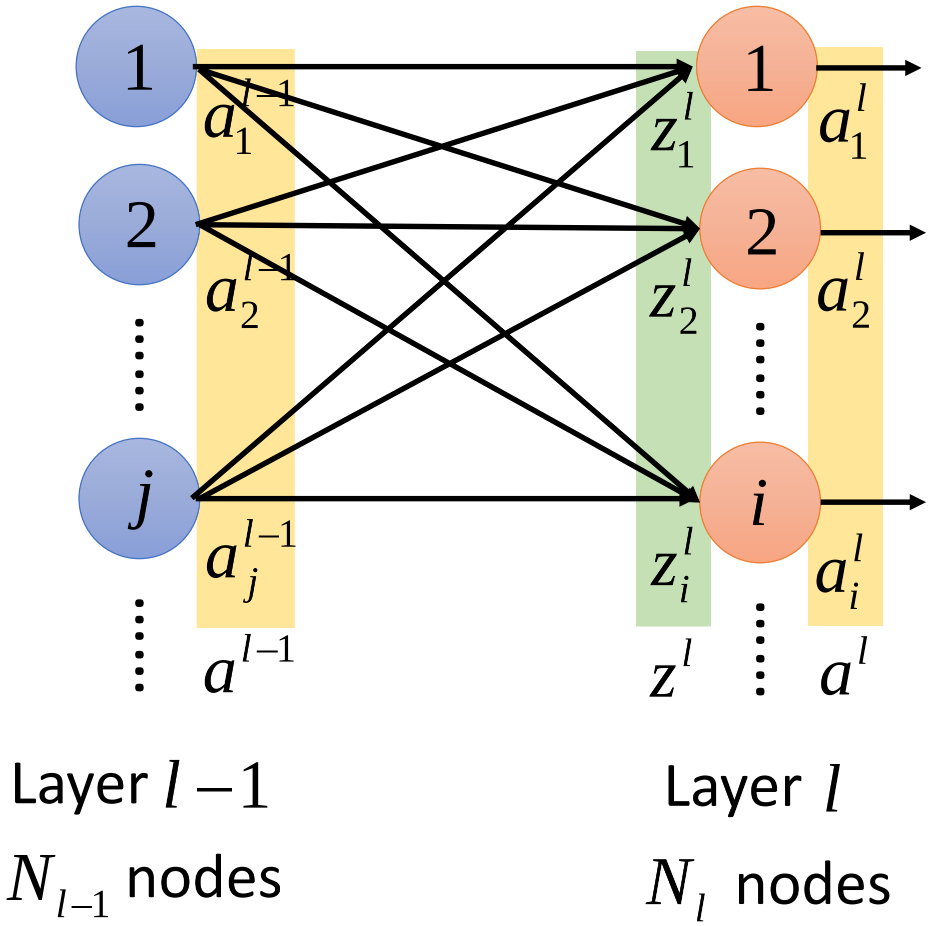


$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs



$$z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

Basic Structure: Recurrent Structure

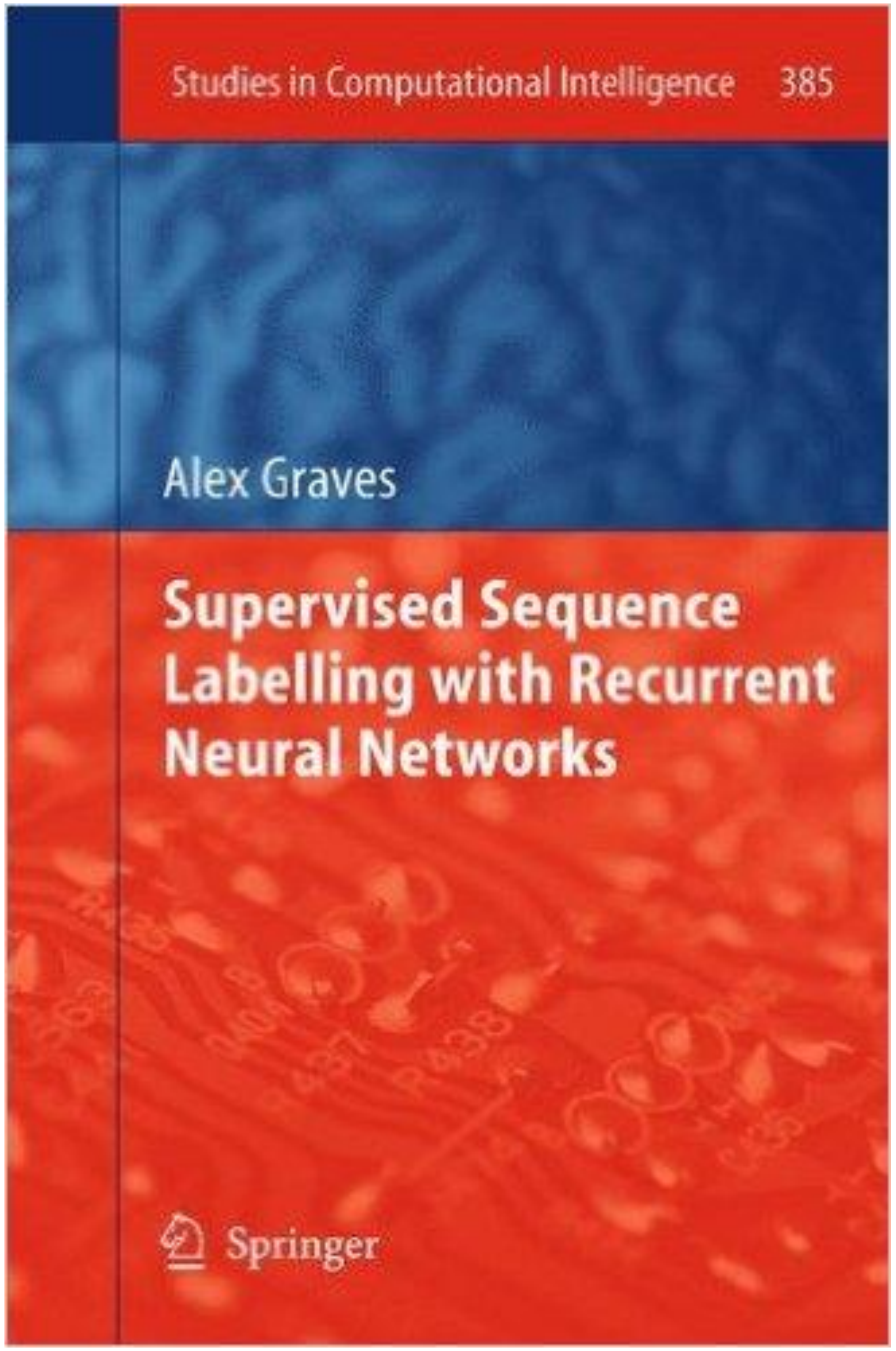
Simplify the network
by using the same function again and again

Reference

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, "LSTM: A Search Space Odyssey," in *IEEE Transactions on Neural Networks and Learning Systems*, 2016

Rafal Józefowicz, Wojciech Zaremba, Ilya Sutskever, “An Empirical Exploration of Recurrent Network Architectures,” in ICML, 2015

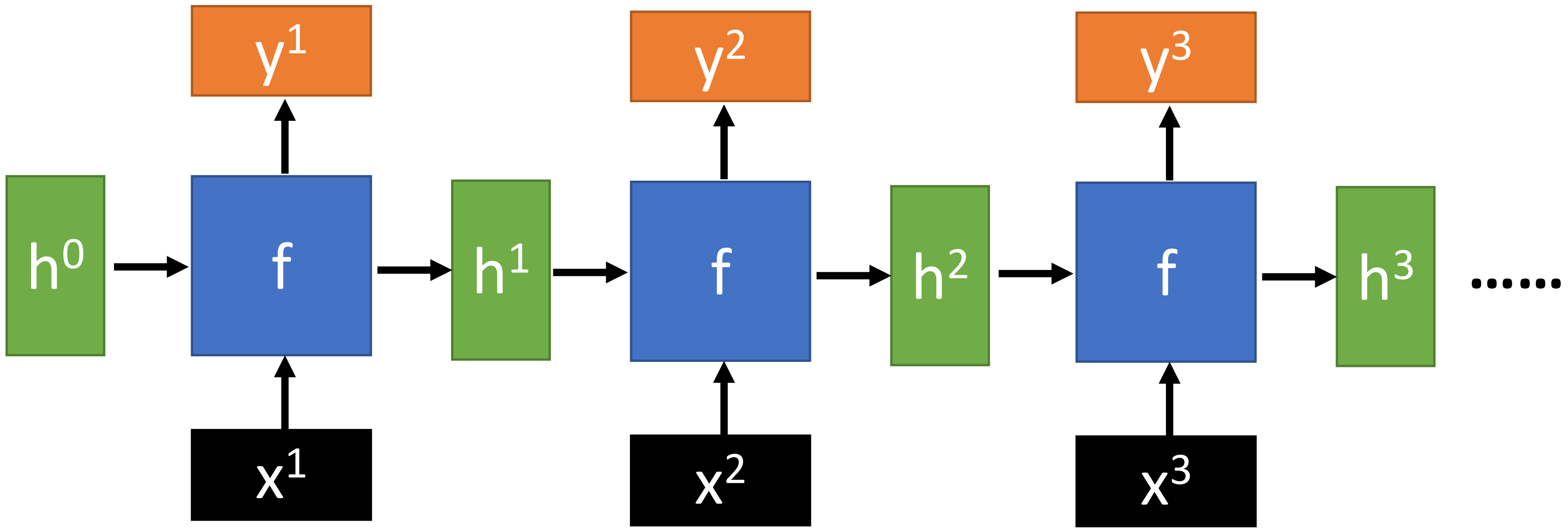
https://www.cs.toronto.edu/~graves/preprint.pdf


Studies in Computational Intelligence 385
Alex Graves
Supervised Sequence Labelling with Recurrent Neural Networks
Springer

Recurrent Neural Network

- Given function f: $h', y = f(h, x)$

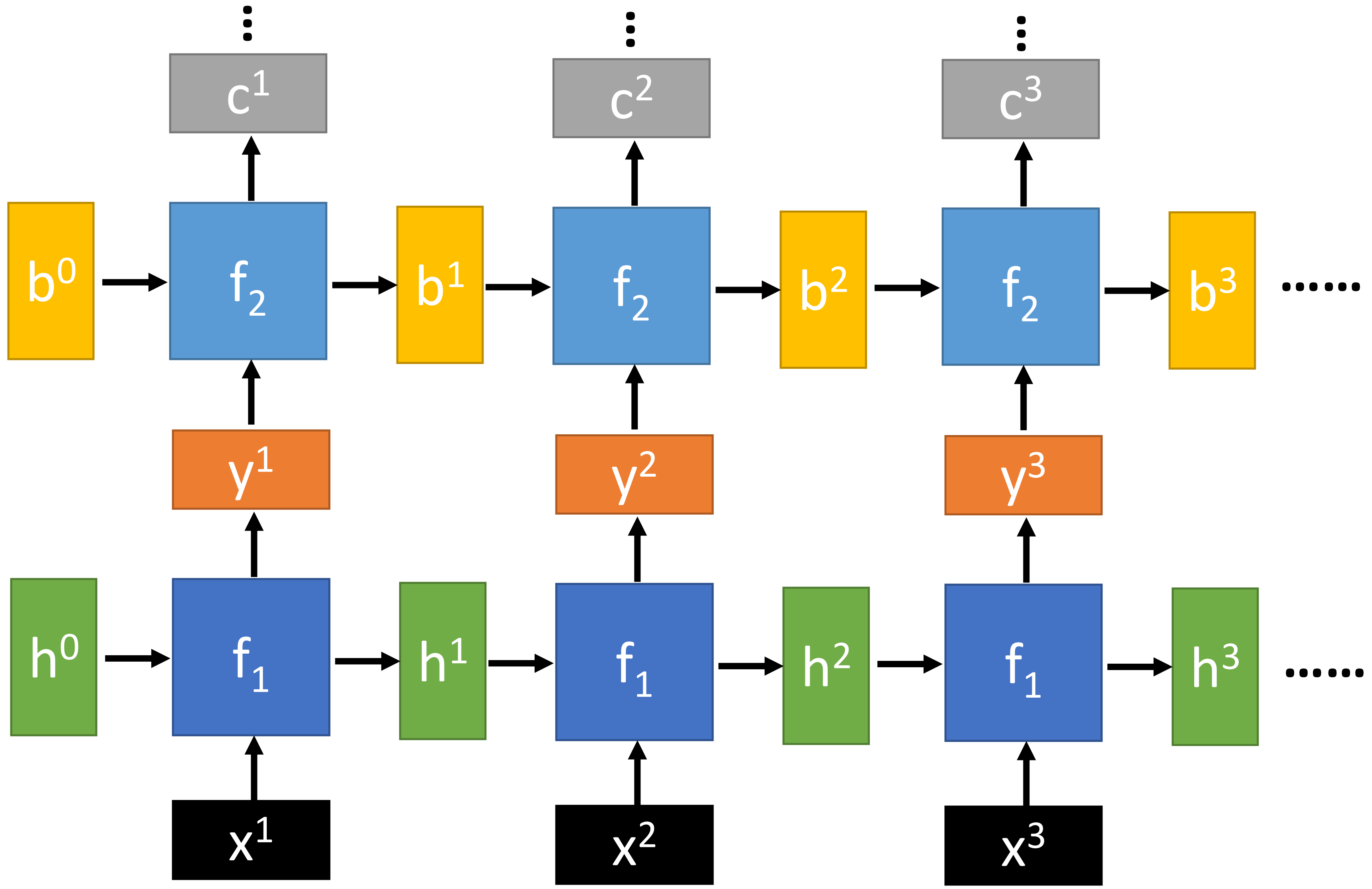
h and h' are vectors with the same dimension



No matter how long the input/output sequence is, we only need one function f

Deep RNN

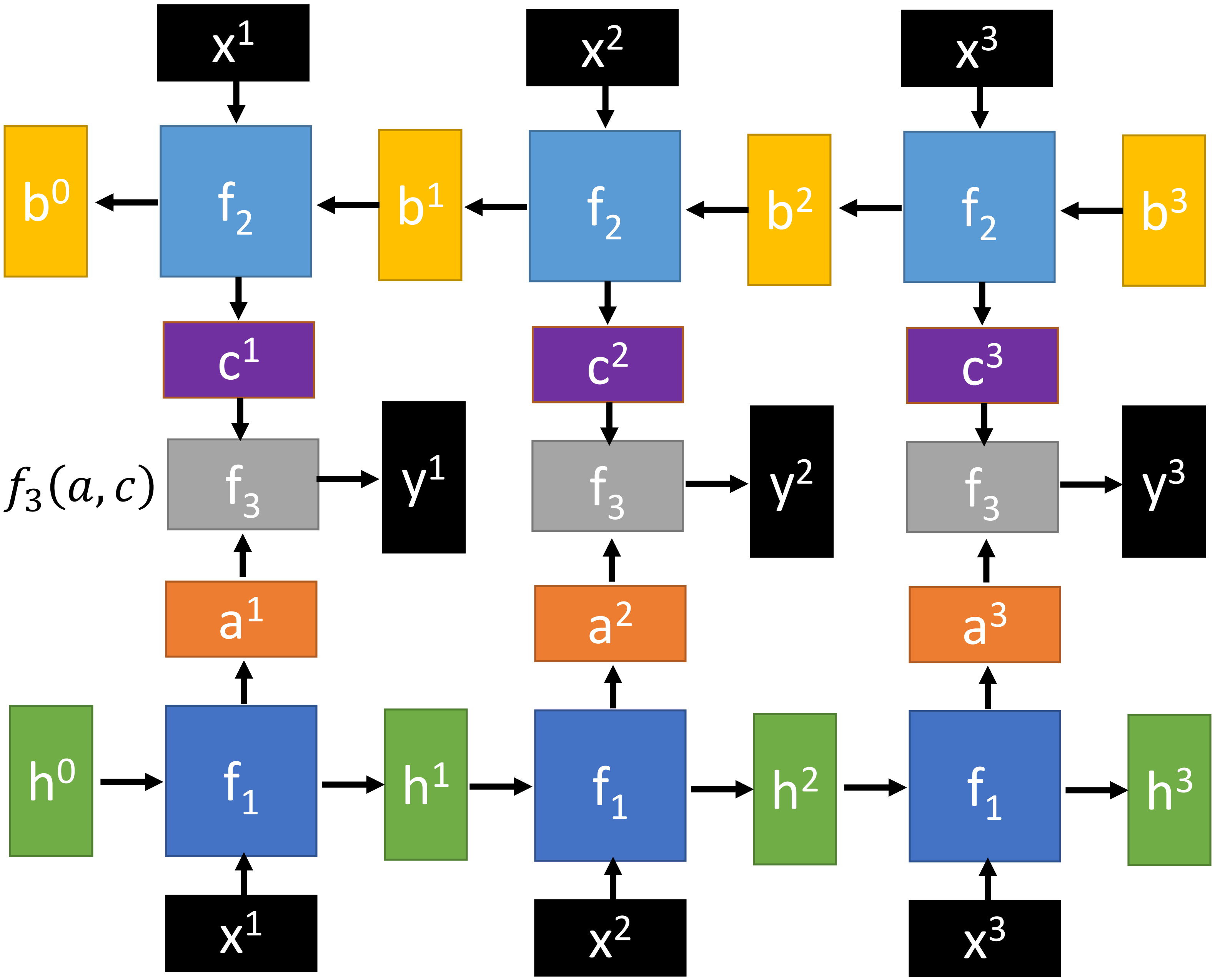
$h', y = f_1(h, x) \quad b', c = f_2(b, y) \quad \cdots$



Bidirectional RNN

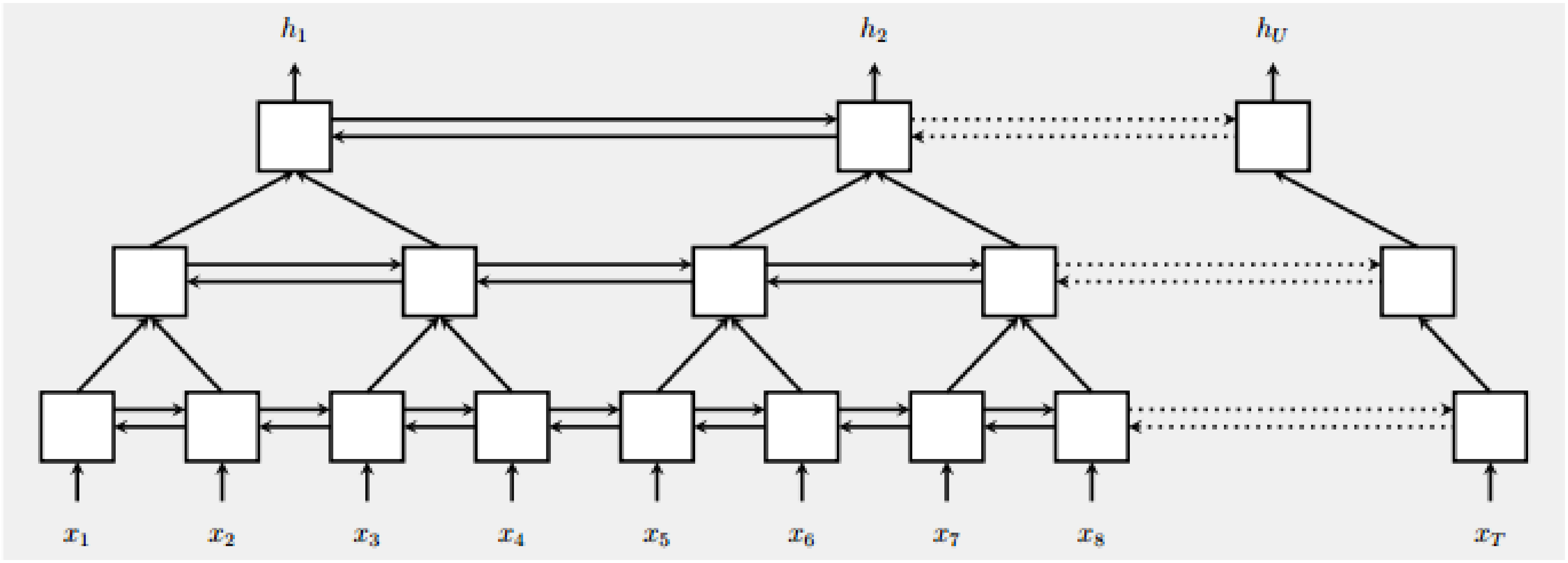
$h', a = f_1(h, x) \quad b', c = f_2(b, x)$

$y = f_3(a, c)$



Pyramidal RNN

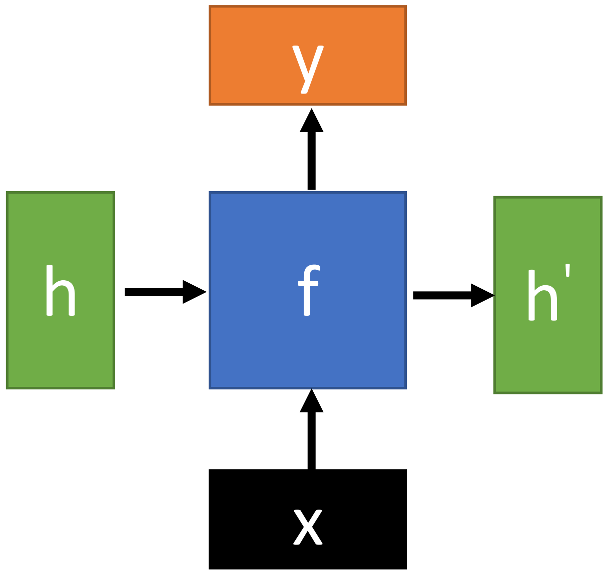
- Reducing the number of time steps



W. Chan, N. Jaitly, Q. Le and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," ICASSP, 2016

Naïve RNN

- Given function f: $h', y = f(h, x)$



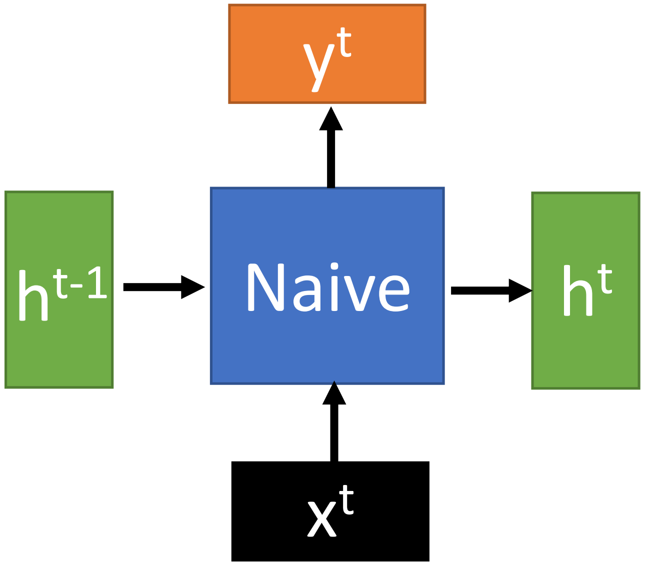
$$h' = \sigma(W^h h + W^i x)$$

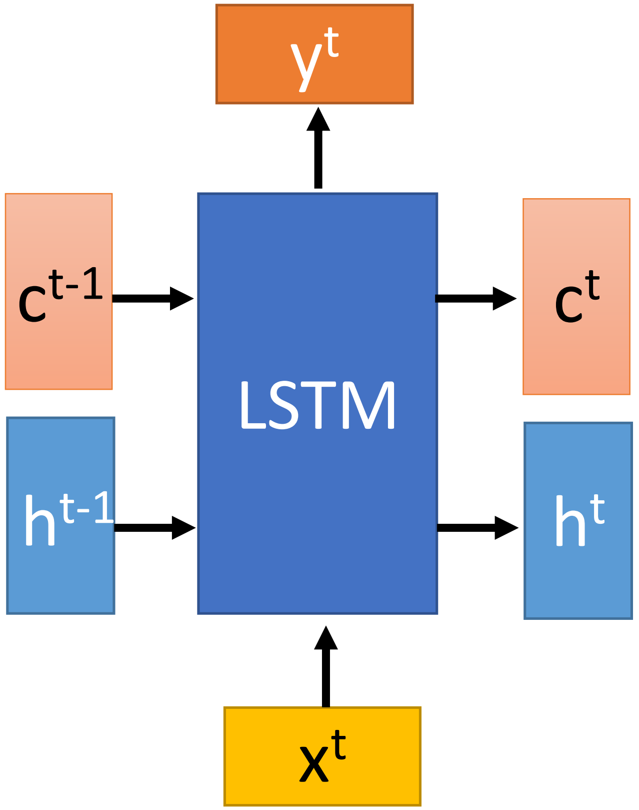
$$y = \sigma(W^o h')$$

softmax

Ignore bias here

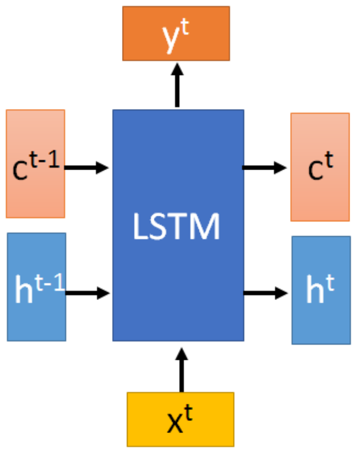
LSTM





c change slowly ➡ c^t is c^{t-1} added by something

h change faster ➡ h^t and h^{t-1} can be very different



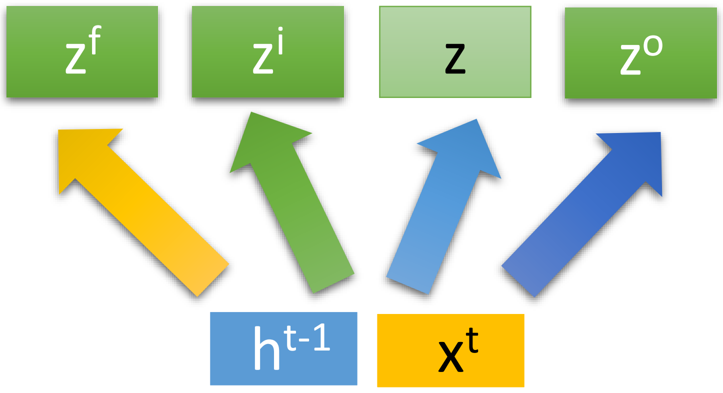
c^{t-1}

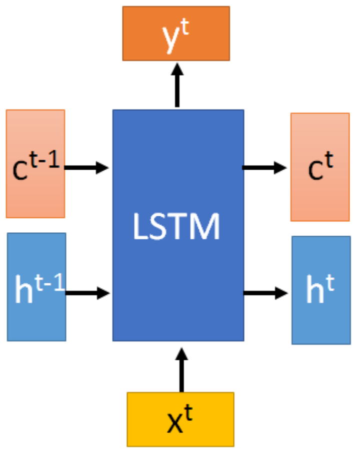
$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

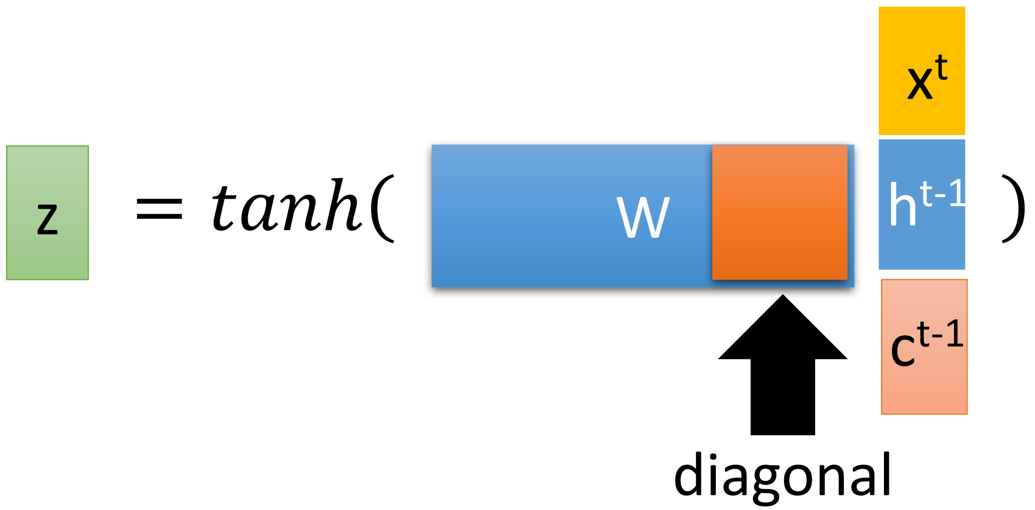
$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

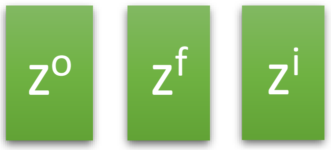
$$z^f = \sigma\left(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

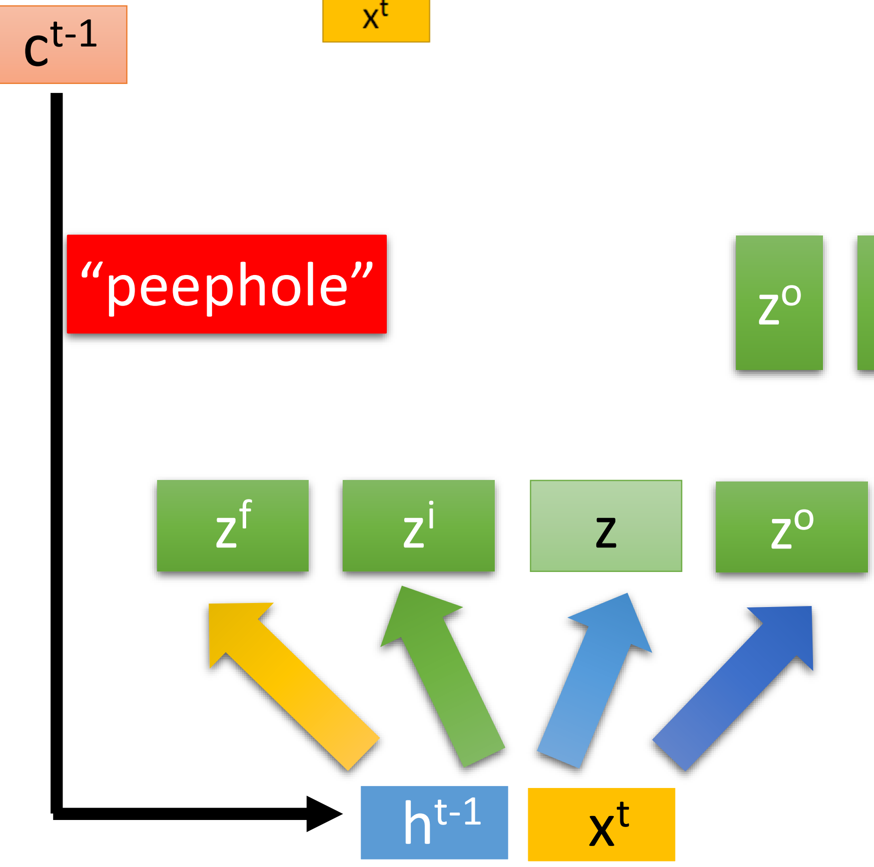


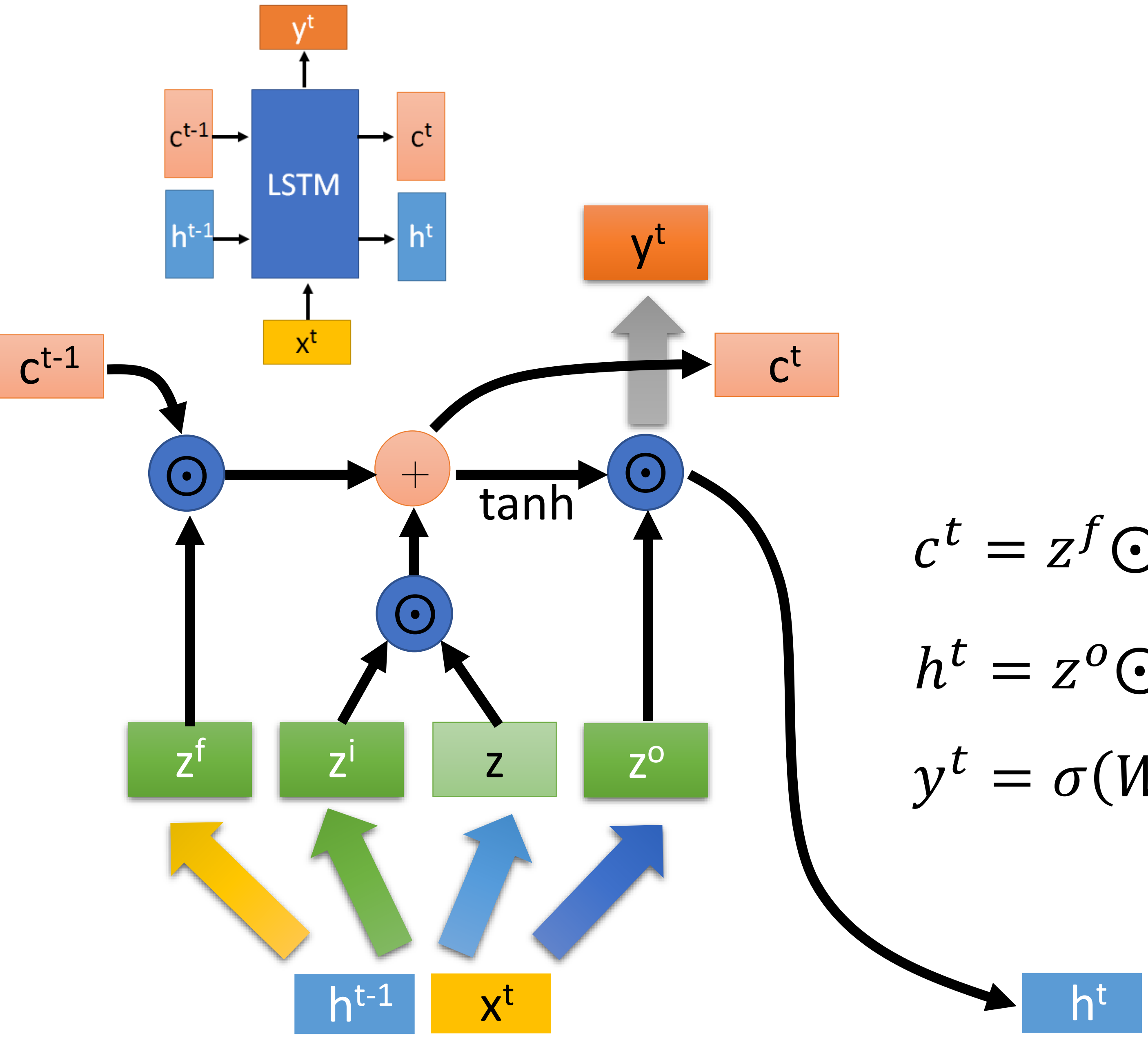






obtained by the same way



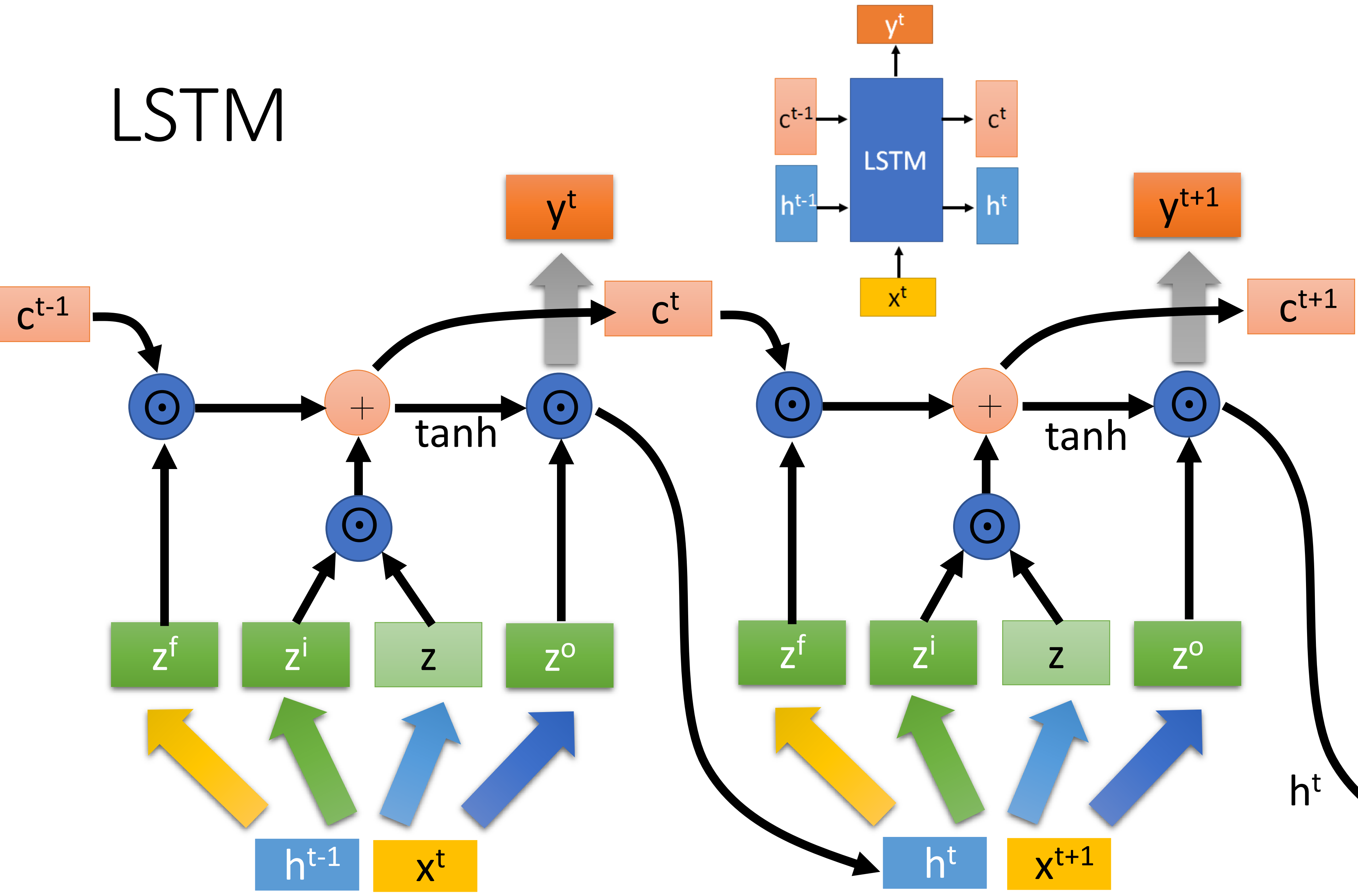


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$

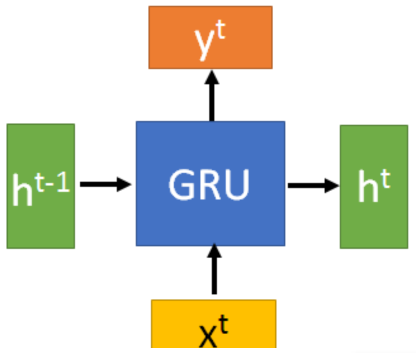
LSTM

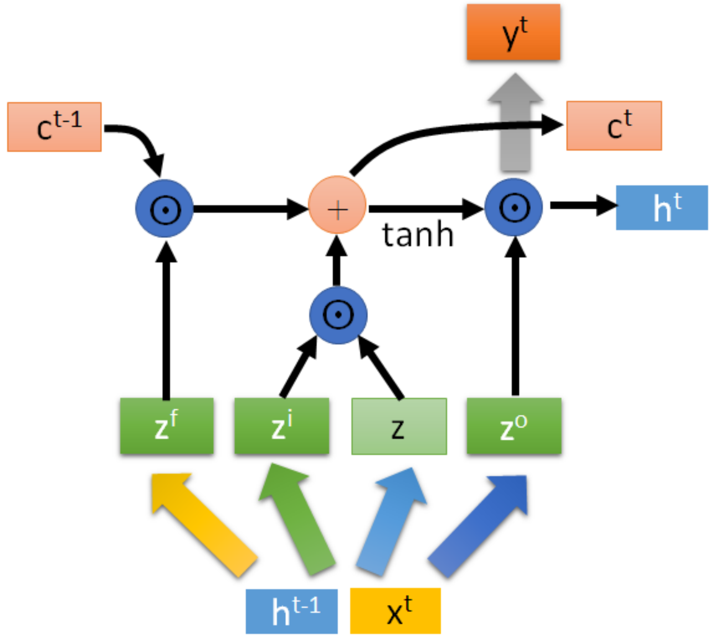


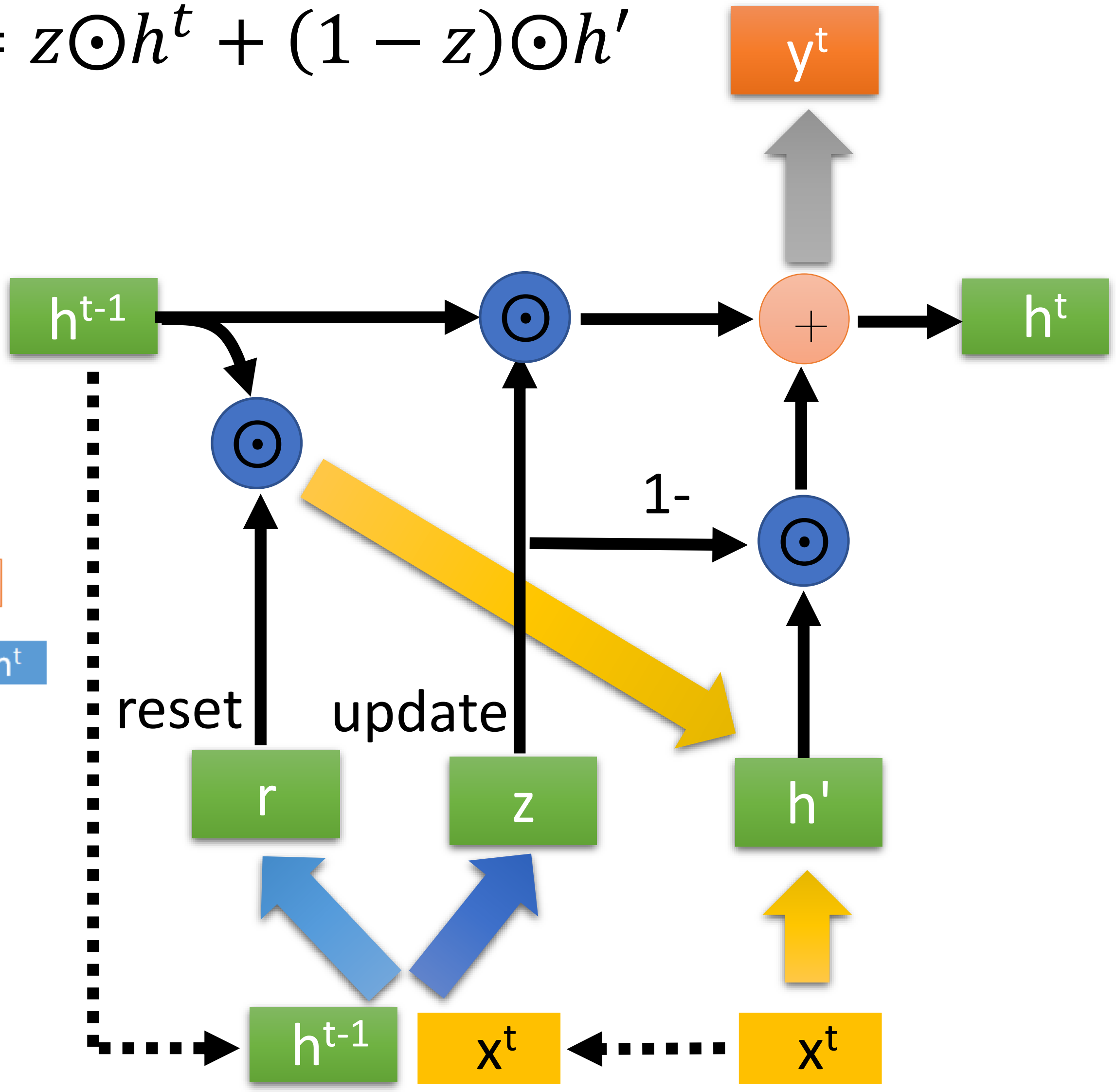
y^t
c^{t-1}
LSTM
c^t
h^{t-1}
h^t
x^t
y^t
y^{t+1}
c^{t-1}
c^t
c^{t+1}
tanh
tanh
z^f
z^i
z
z^o
z^f
z^i
z
z^o
h^{t-1}
x^t
h^t
x^{t+1}
h^t

GRU

$$h^t = z \odot h^t + (1 - z) \odot h'$$

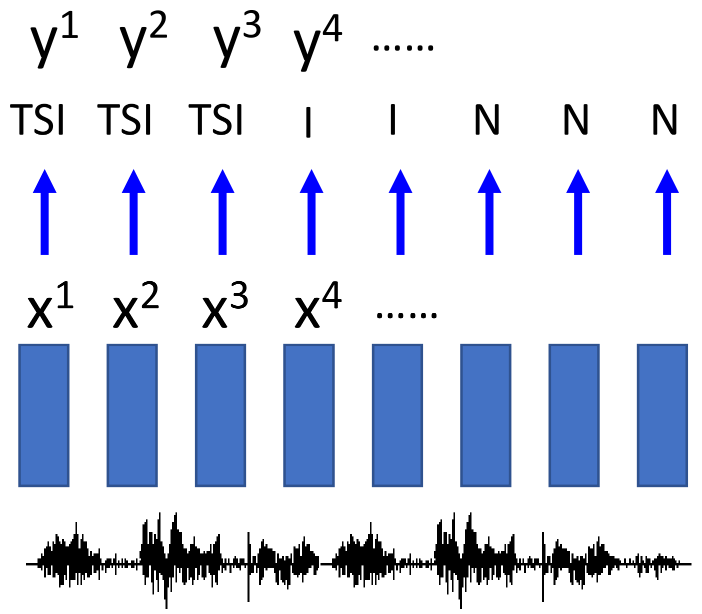




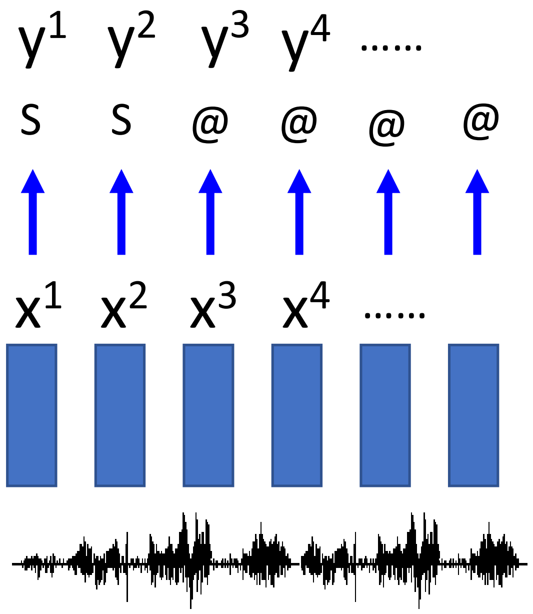


Example Task

- (Simplified) Speech Recognition: Frame classification on TIMIT



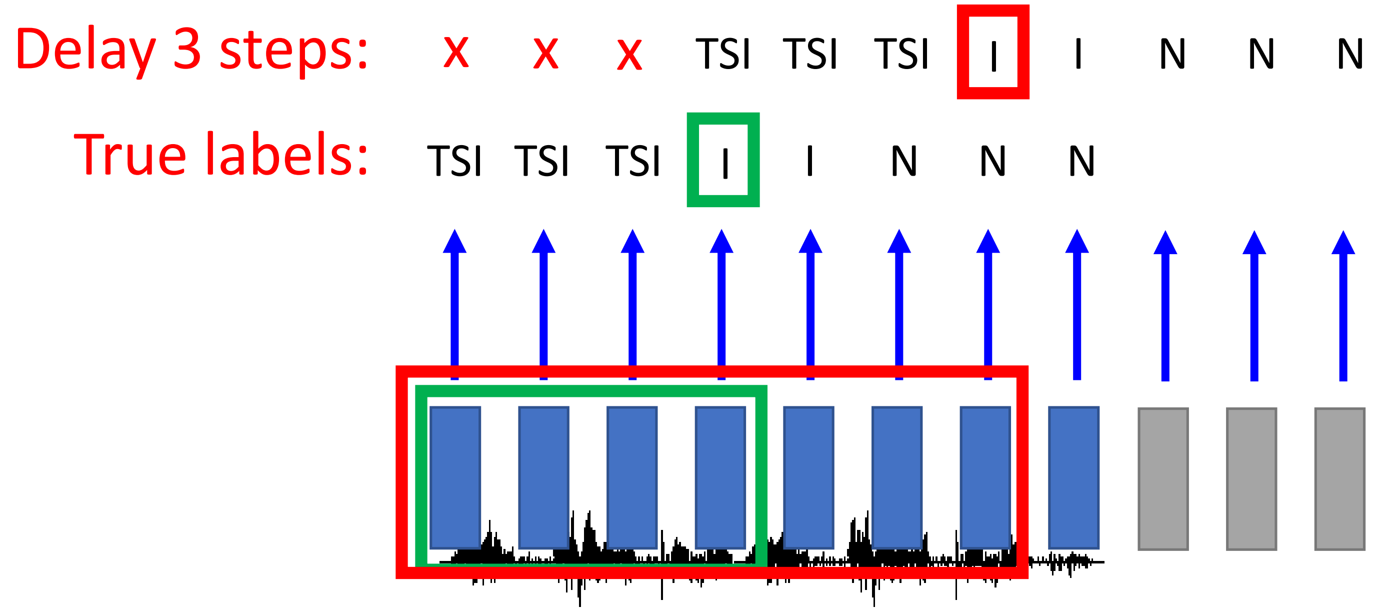
Utterance 1

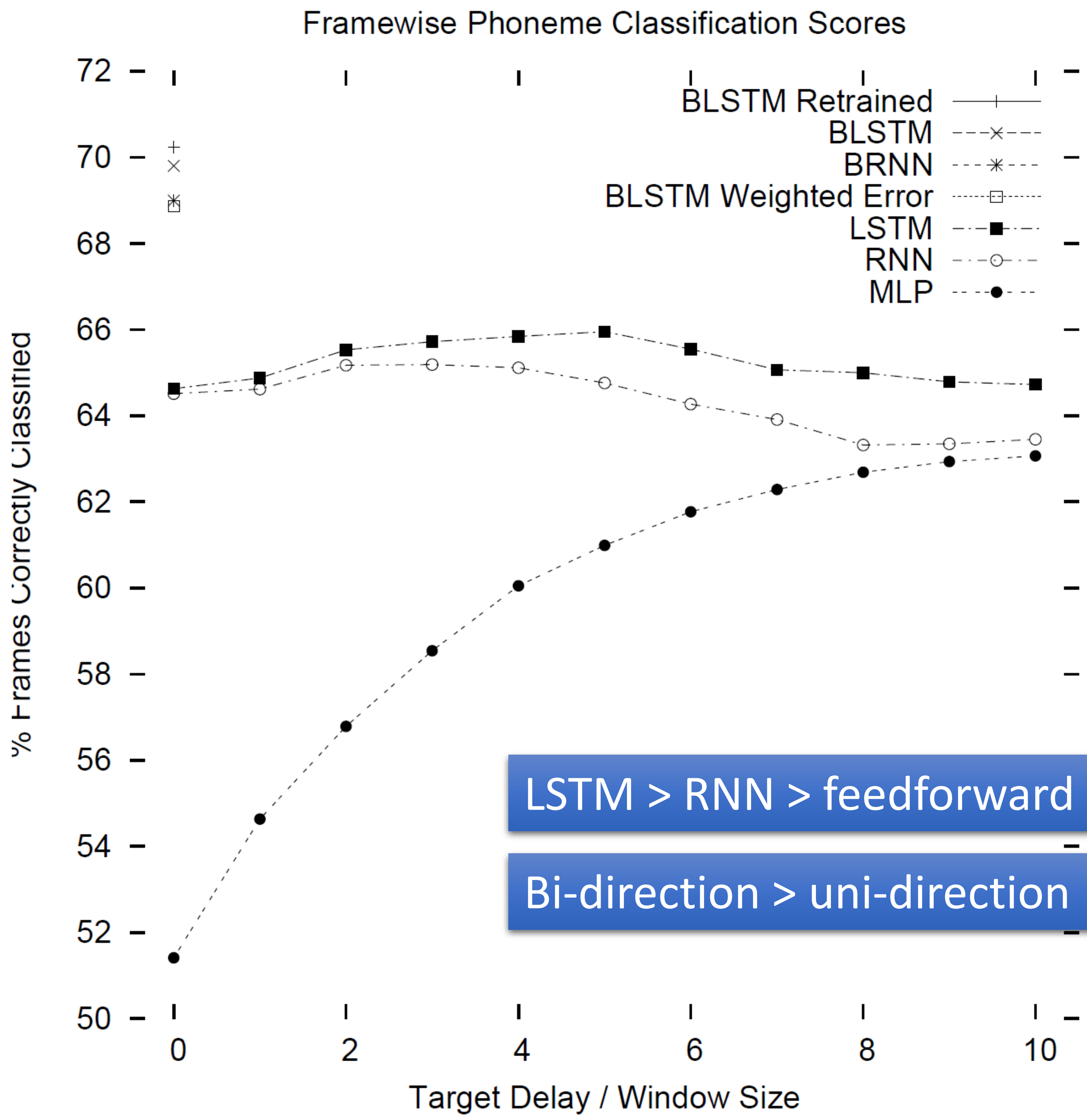


Utterance 2

Target Delay

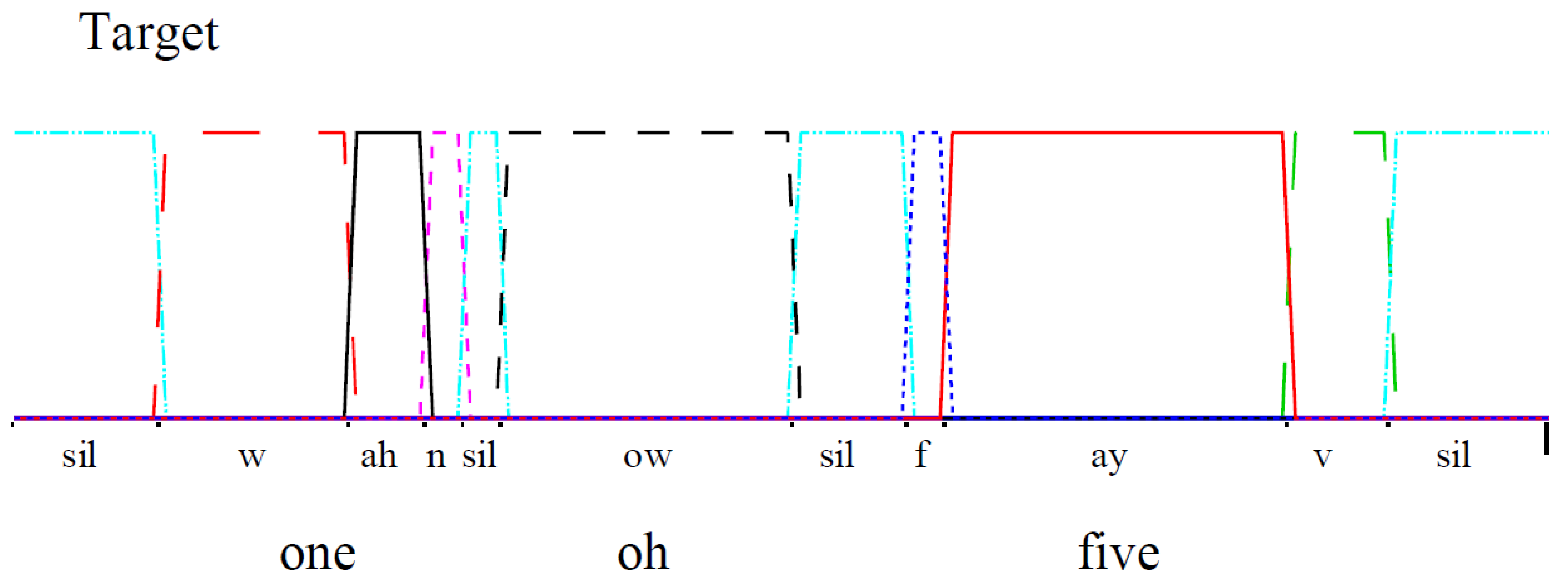
- Only for unidirectional RNN

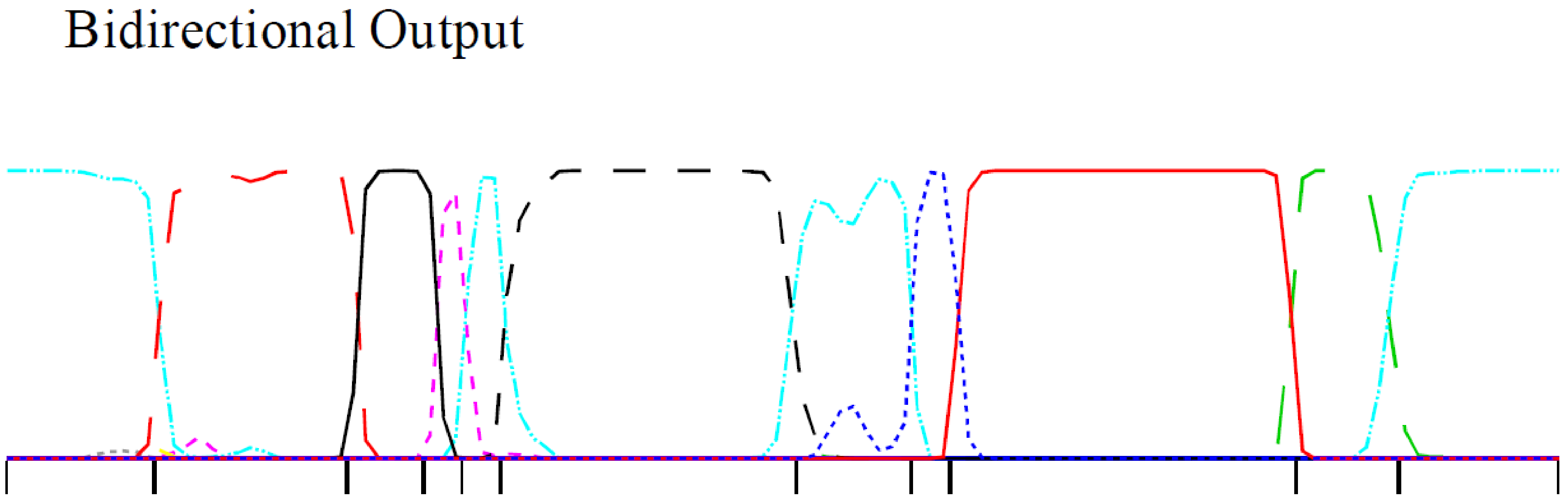


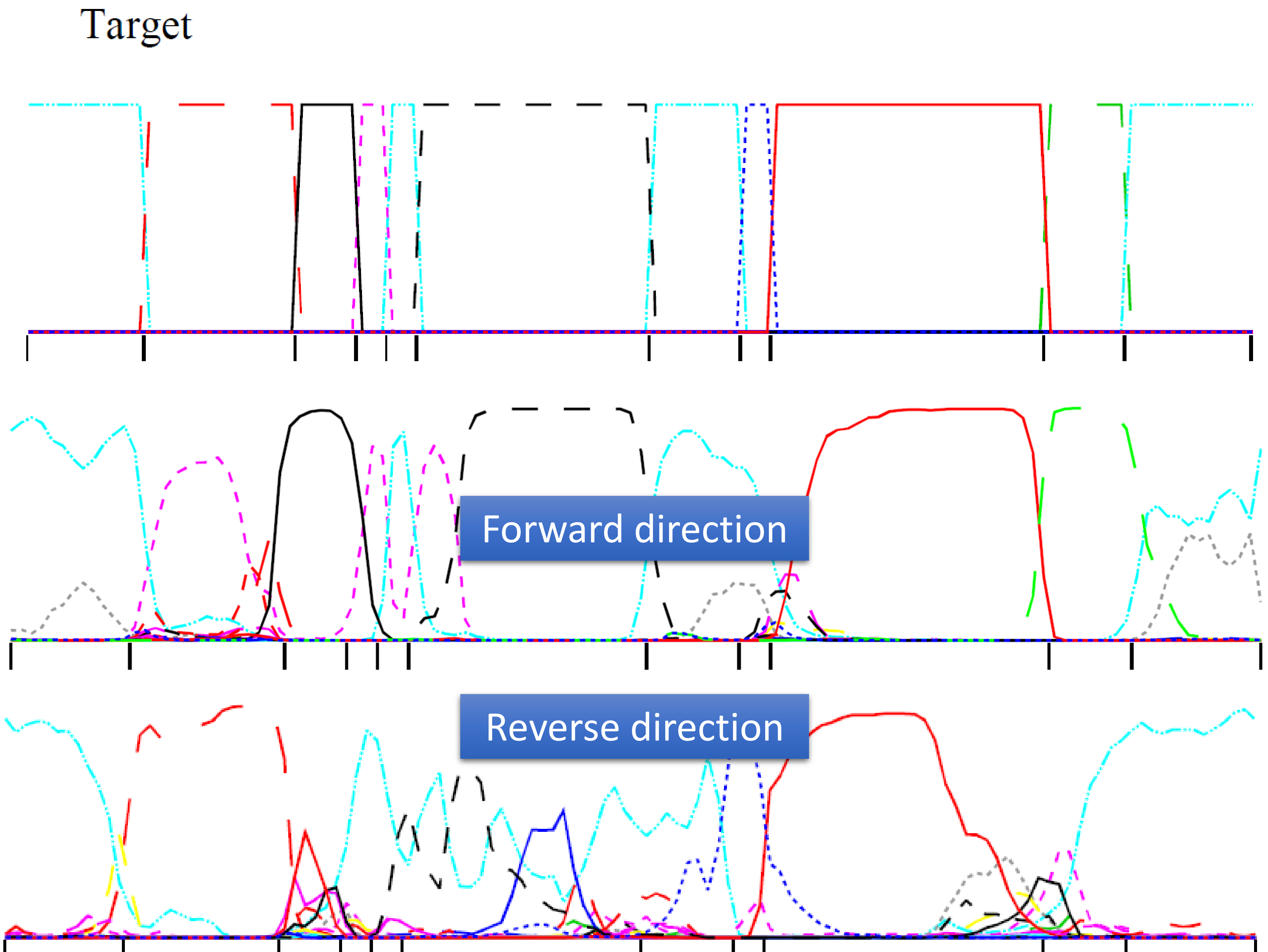


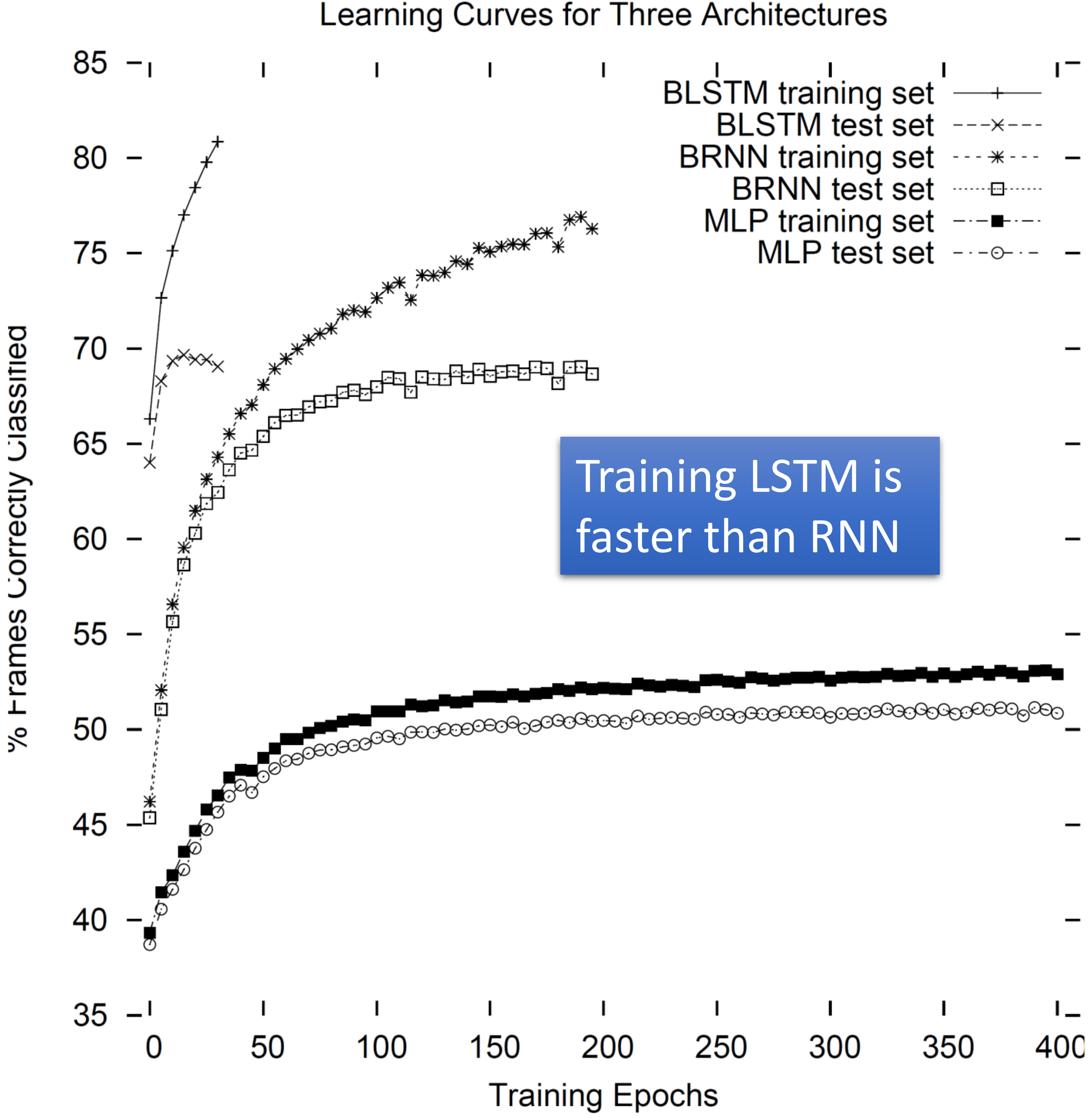
LSTM > RNN > feedforward

Bi-direction > uni-direction

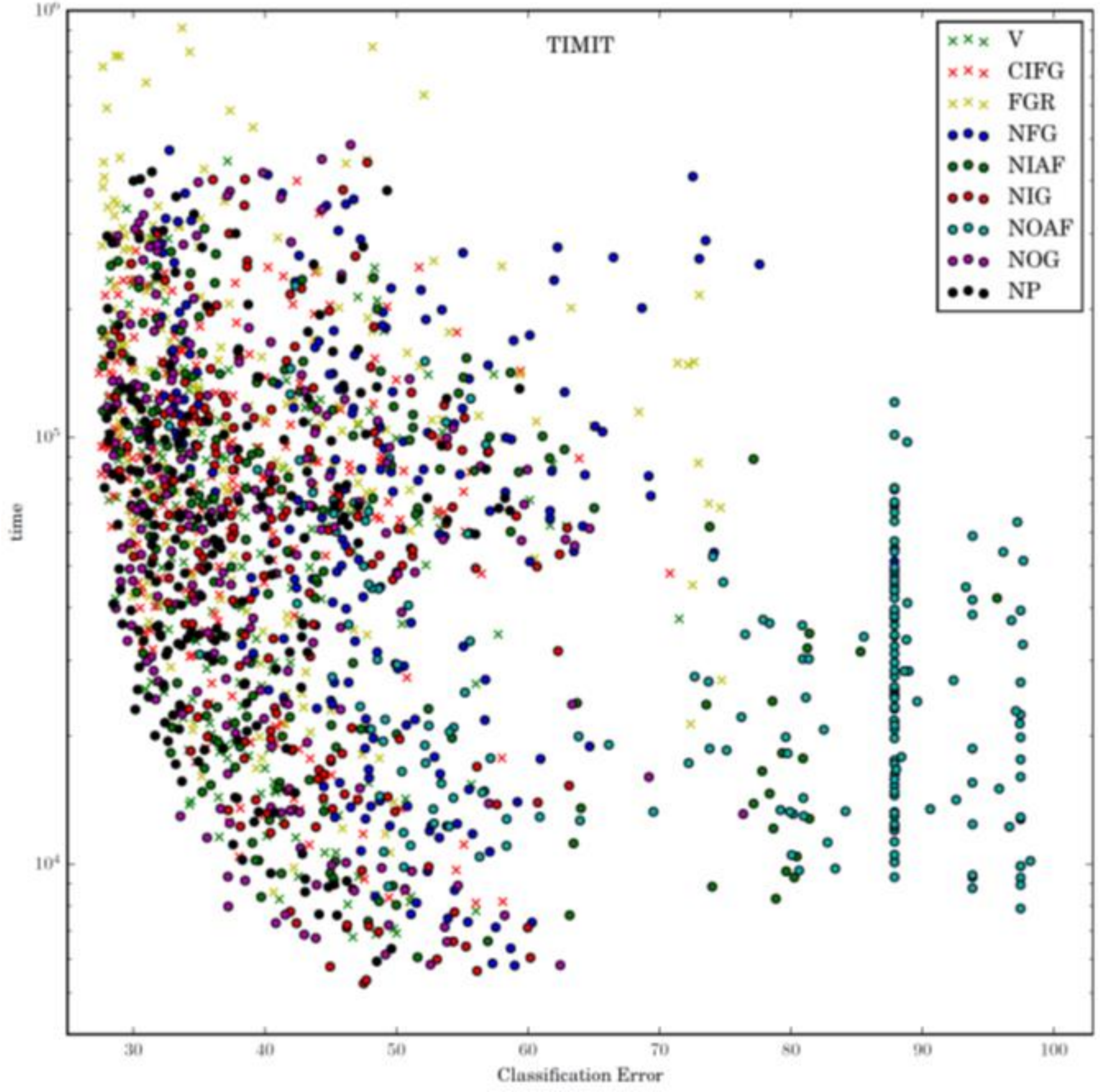

Target
sil
w
ah
n
sil
ow
sil
f
ay
v
sil
one
oh
five


Bidirectional Output


Target
Forward direction
Reverse direction

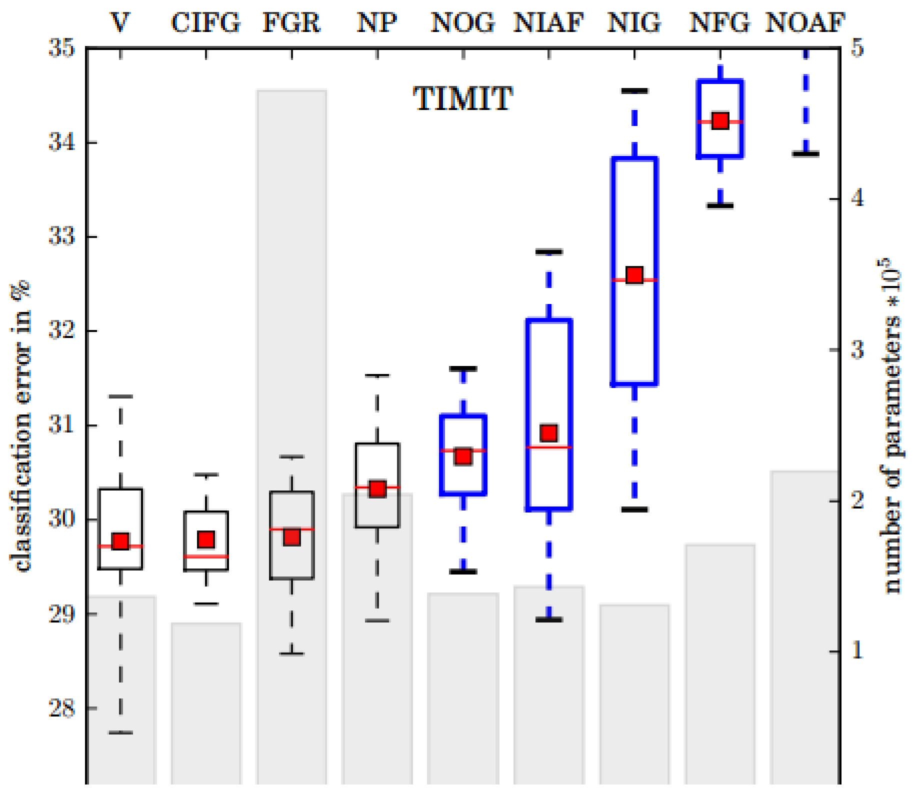


Training LSTM is faster than RNN


TIMIT
V
CIFG
FGR
NFG
NIAF
NIG
NOAF
NOG
NP
time
10^6
10^5
10^4
Classification Error
30
40
50
60
70
80
90
100

LSTM: A Search Space Odyssey

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)



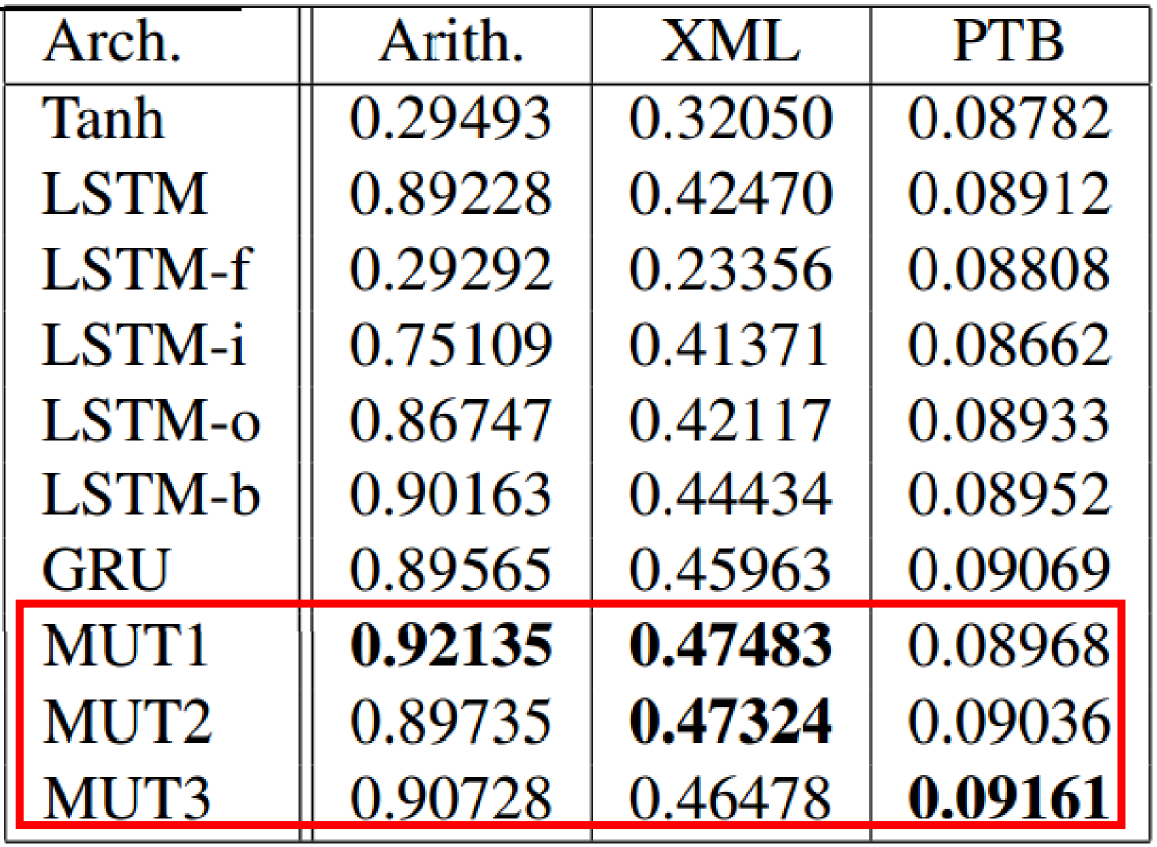
Standard LSTM works well

Simply LSTM: coupling input and forget gate, removing peephole

Forget gate is critical for performance

Output gate activation function is critical

An Empirical Exploration of Recurrent Network Architectures



Arch.	Arith.	XML	PTB
Tanh	0.29493	0.32050	0.08782
LSTM	0.89228	0.42470	0.08912
LSTM-f	0.29292	0.23356	0.08808
LSTM-i	0.75109	0.41371	0.08662
LSTM-o	0.86747	0.42117	0.08933
LSTM-b	0.90163	0.44434	0.08952
GRU	0.89565	0.45963	0.09069
MUT1	**0.92135**	**0.47483**	0.08968
MUT2	0.89735	**0.47324**	0.09036
MUT3	0.90728	0.46478	**0.09161**

LSTM-f/i/o: removing forget/input/output gates

LSTM-b: large bias

Importance: forget > input > output

Large bias for forget gate is helpful

An Empirical Exploration of Recurrent Network Architectures

MUT1:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

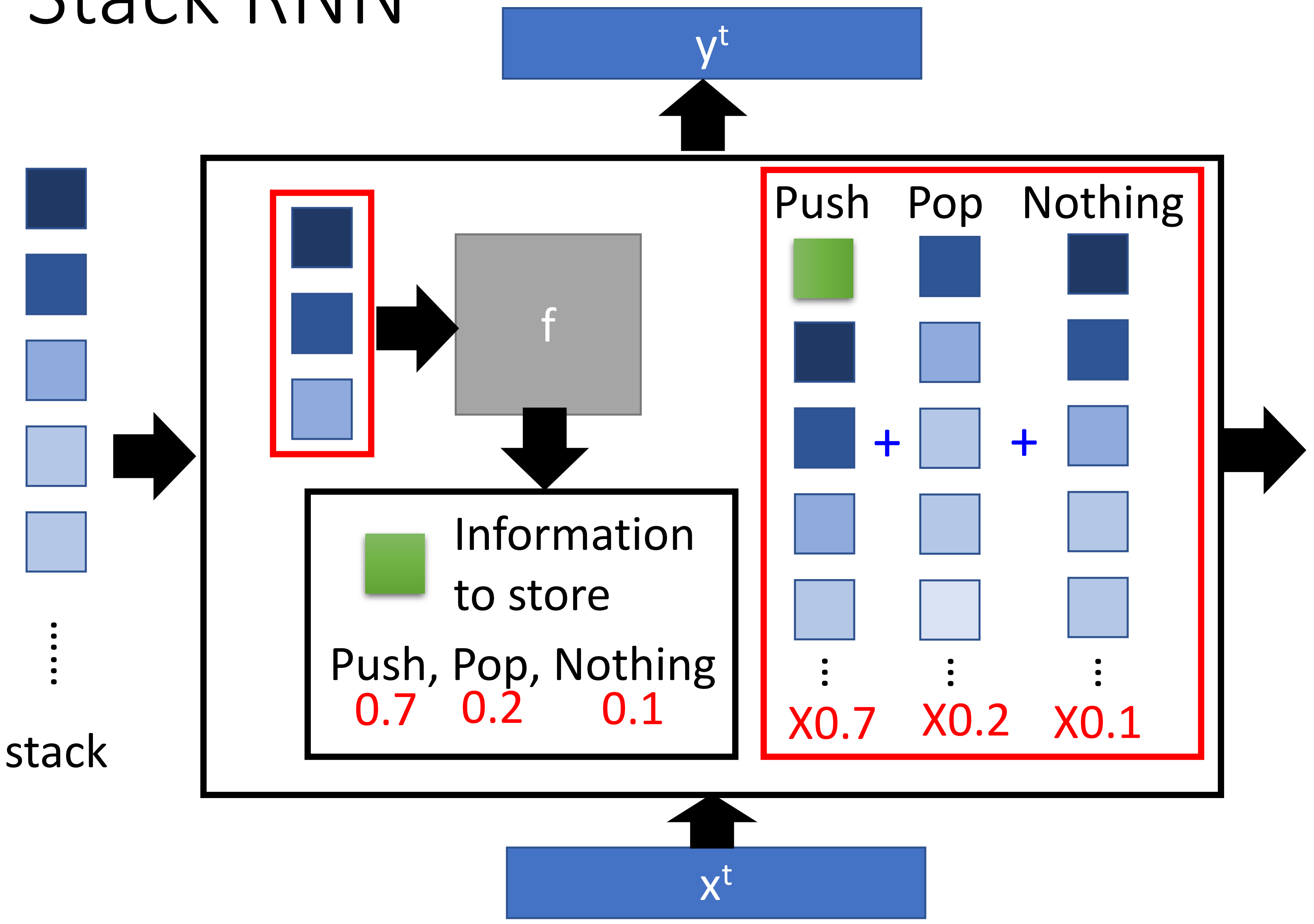
$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} \tanh(h_t) + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Armand Joulin, Tomas Mikolov, Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets, arXiv Pre-Print, 2015

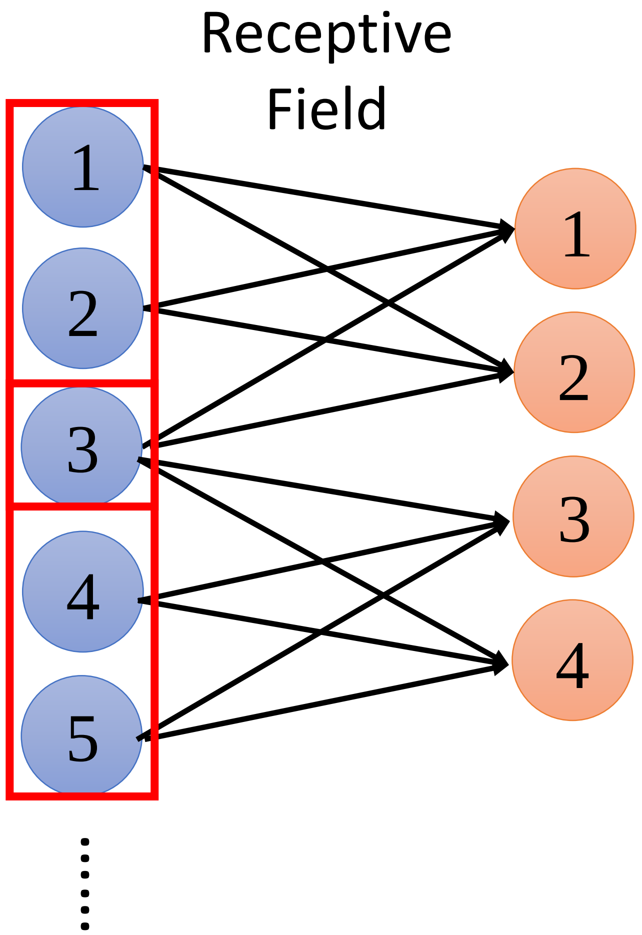
Stack RNN



Basic Structure: Convolutional / Pooing Layer

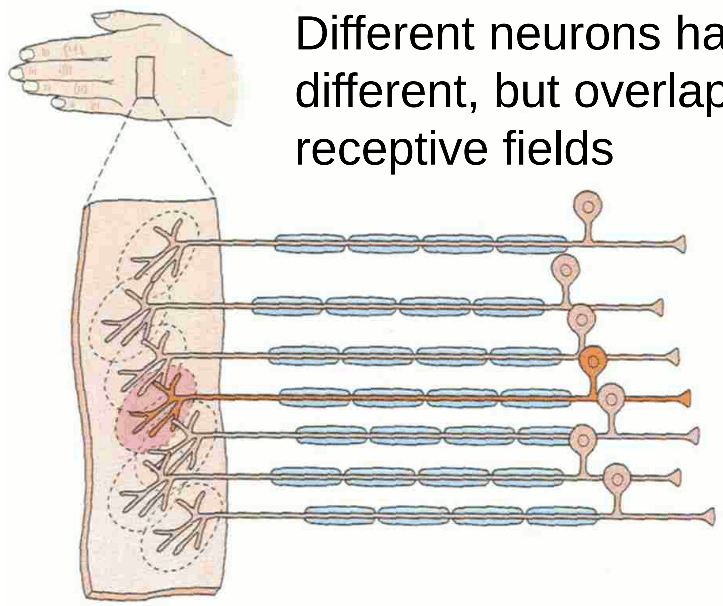
Simplify the neural network
(based on prior knowledge of the task)

Convolutional Layer



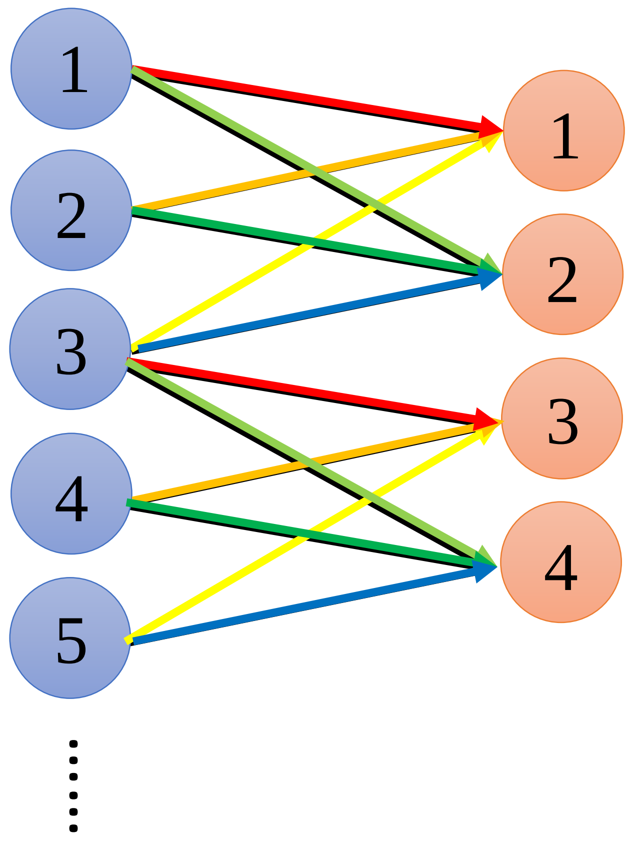
__Sparse Connectivity__

Each neural only connects to part of the output of the previous layer



Different neurons have different, but overlapping, receptive fields

Convolutional Layer



Sparse Connectivity

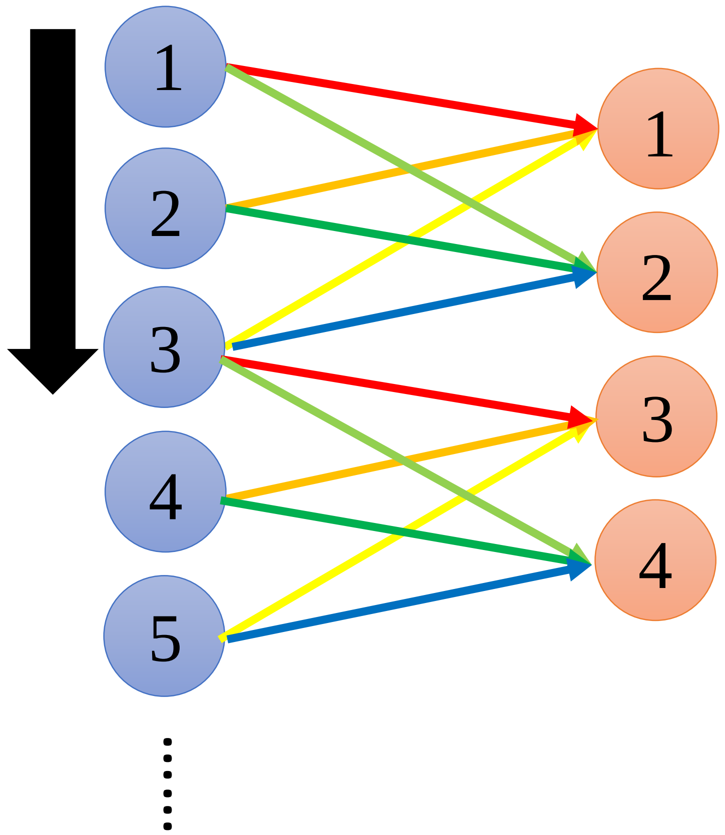
Each neural only connects to part of the output of the previous layer

Parameter Sharing

The neurons with different receptive fields can use the same set of parameters.

Less parameters then fully connected layer

Convolutional Layer



Considering neuron 1 and 3 as "filter 1" (kernel 1)

filter (kernel) size: size of the receptive field of a neuron

Stride = 2

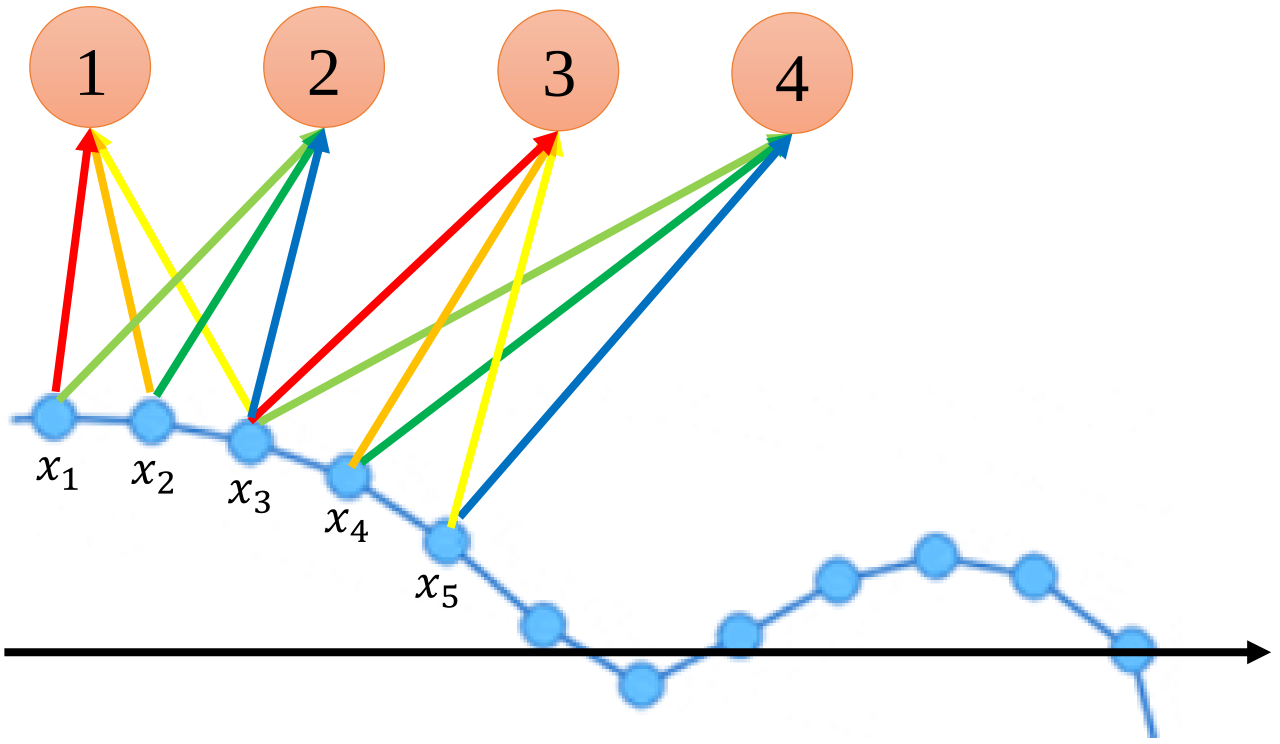
Considering neuron 2 and 4 as "filter 2" (kernel 2)

Kernel size, no. of filter, stride are all designed by the developers.

Example – 1D Signal + Single Channel

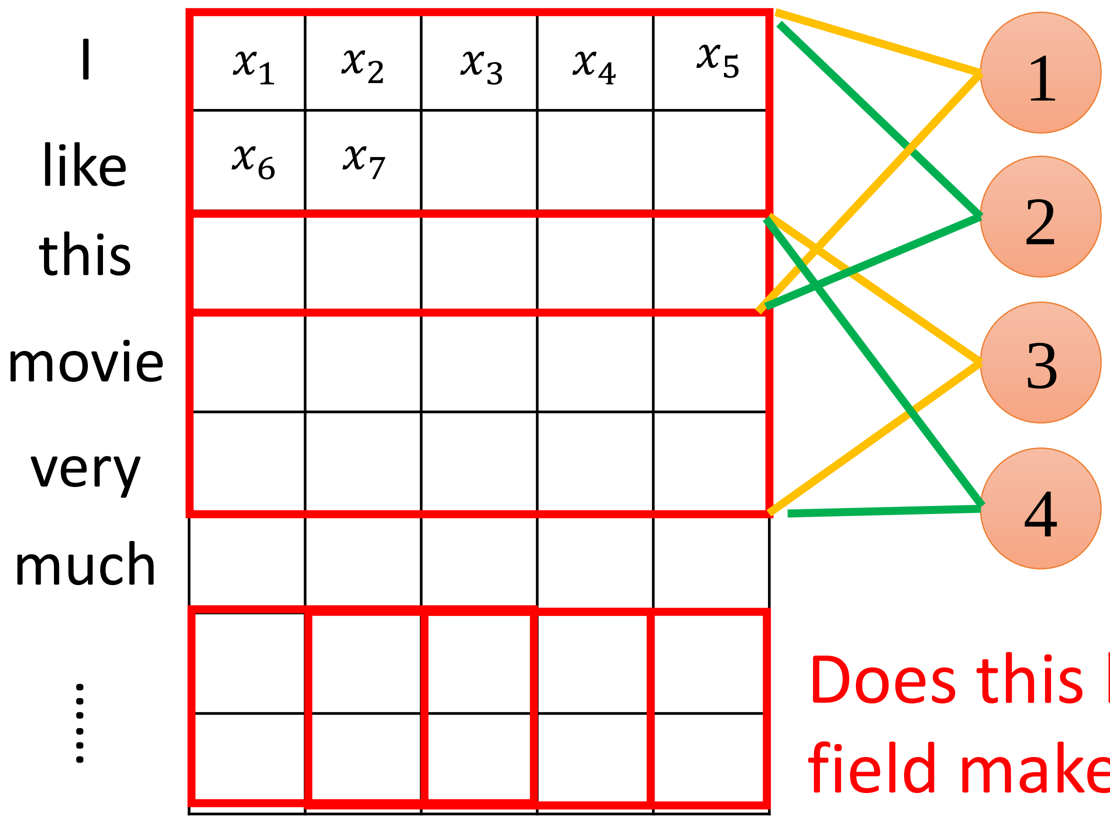
Classification, Predict the future …

Audio Signal,
Stock Value …



Example –
1D Signal + Multiple Channel

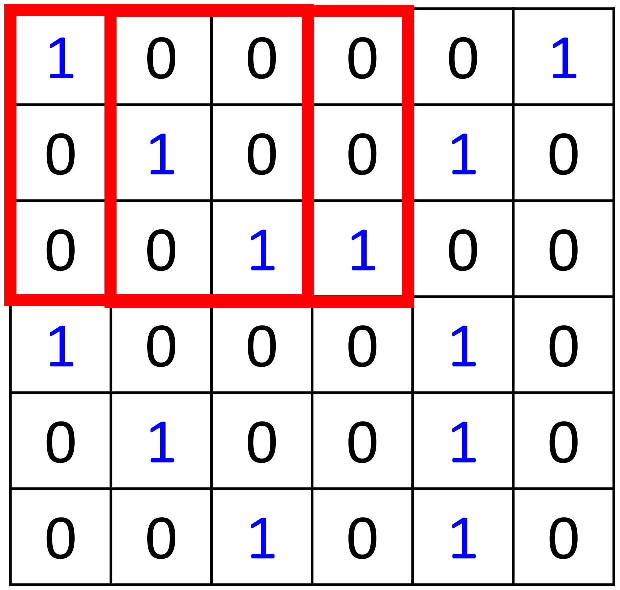
A document: each word is a vector



Does this kind of receptive field make sense?

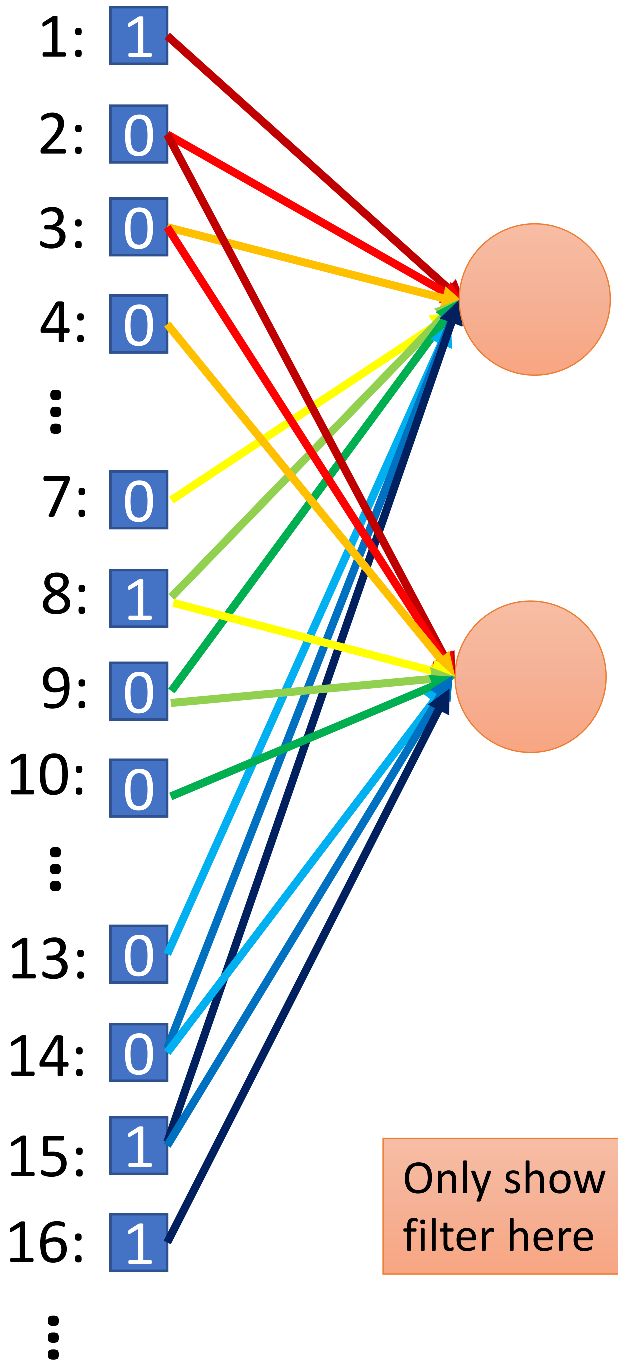
Example –
2D Signal + Single Channel

Size of Receptive field is 3x3, Stride is 1



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

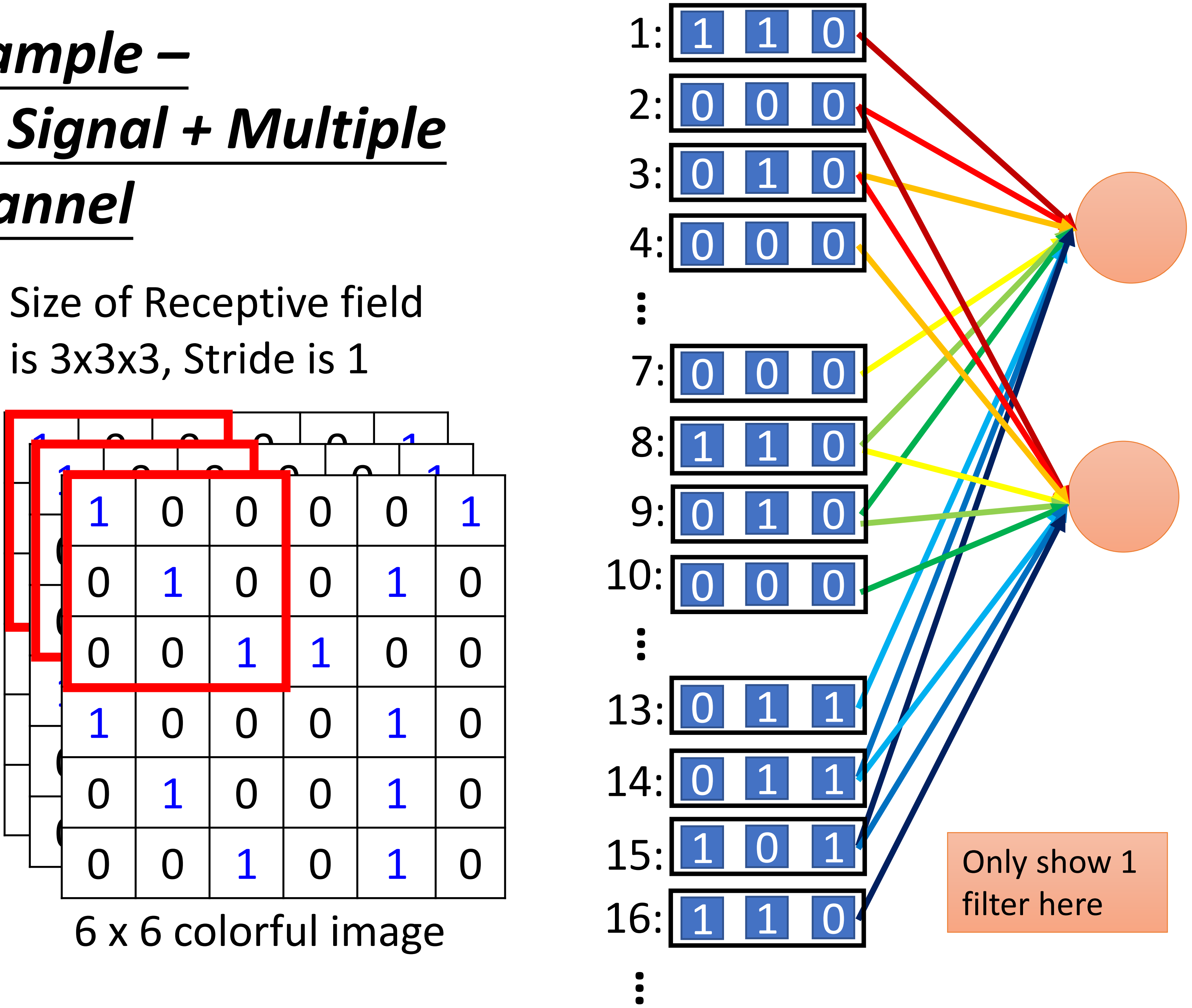
6 x 6 black & white picture image



Only show 1 filter here

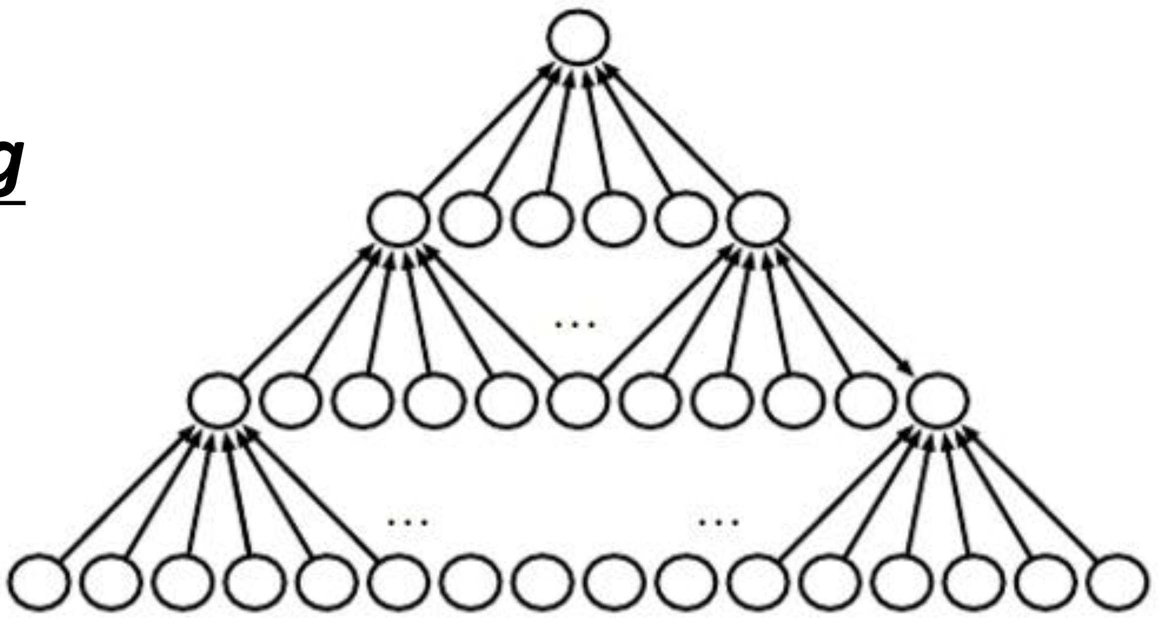
Example – 2D Signal + Multiple Channel

Size of Receptive field is 3x3x3, Stride is 1

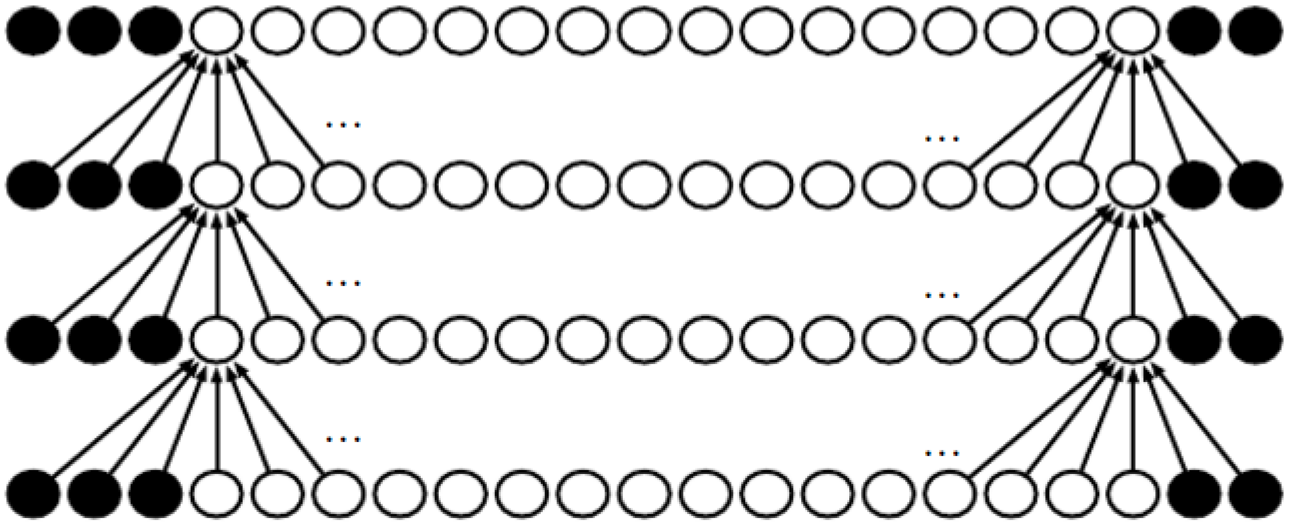


6 x 6 colorful image

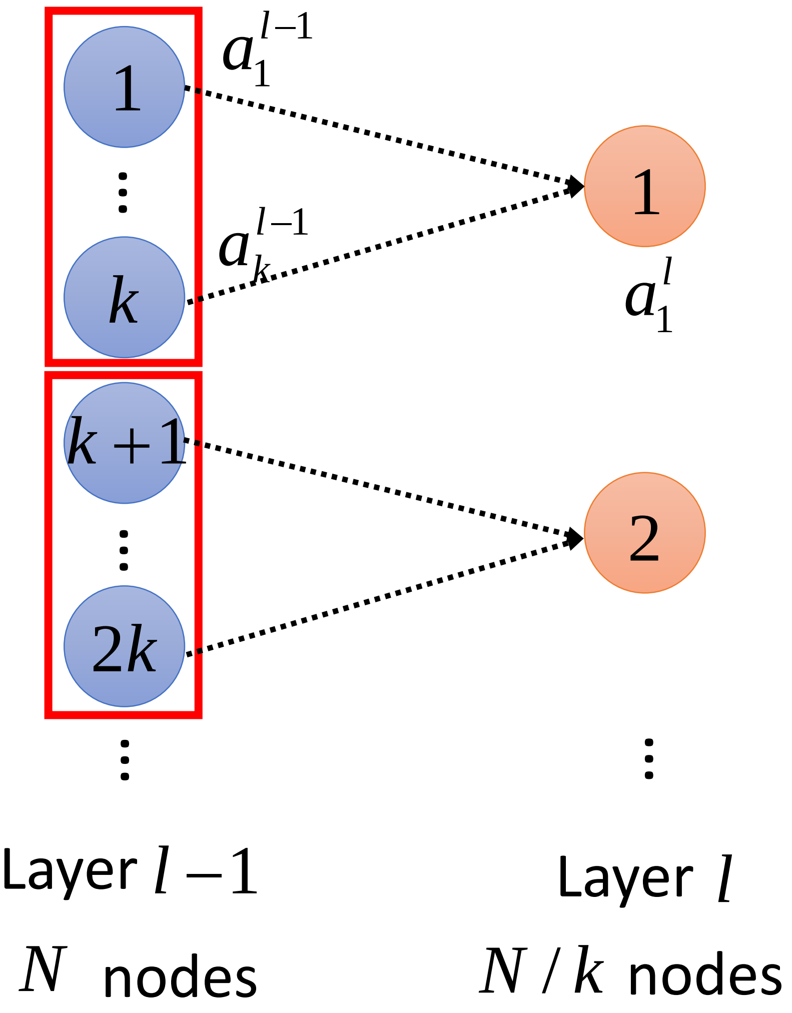
Without Zero Padding



Zero Padding



Pooling Layer



k outputs in layer $l-1$ are grouped together

Each output in layer l "summarizes" k inputs

Average Pooling:

$$a_1^l = \frac{1}{k}\sum_{j=1}^{k} a_j^{l-1}$$

Max Pooling:

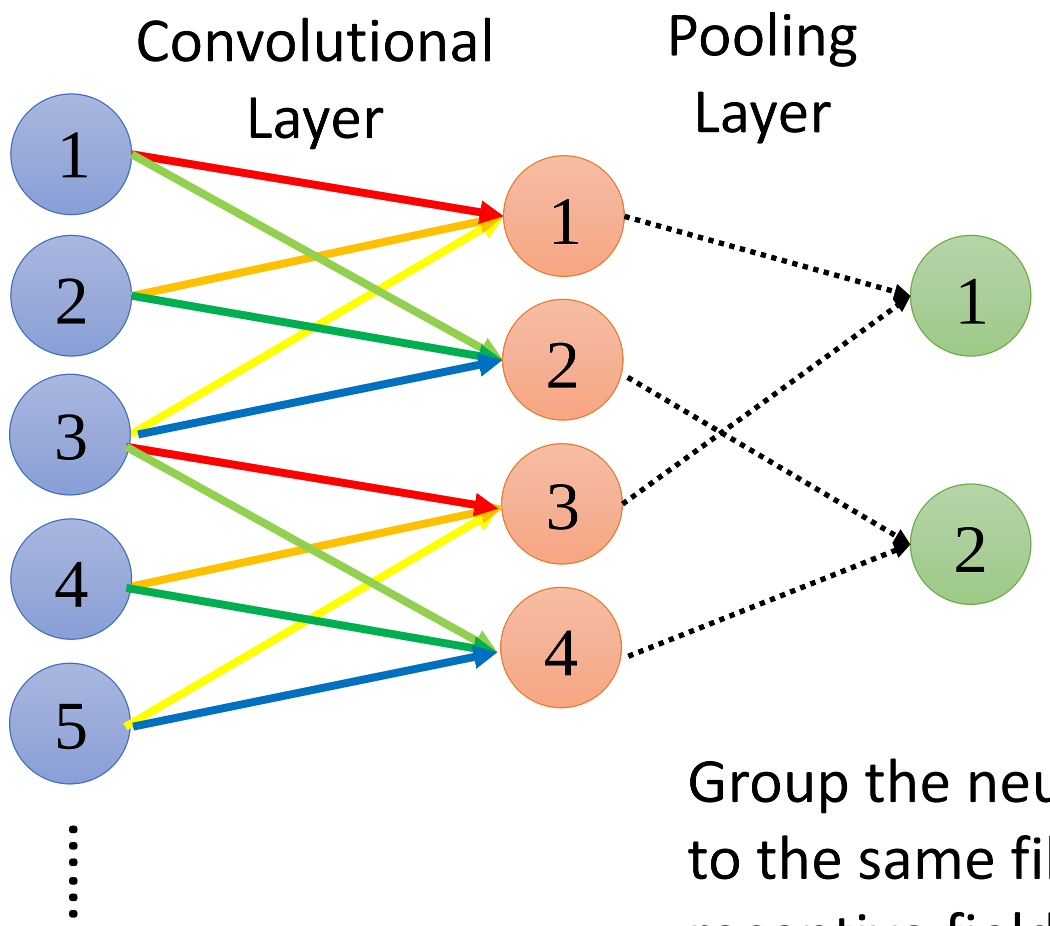
$$a_1^l = \max\left(a_1^{l-1}, a_2^{l-1}, \cdots, a_k^{l-1}\right)$$

L2 Pooling:

$$a_1^l = \frac{1}{k}\sqrt{\sum_{j=1}^{k}\left(a_j^{l-1}\right)^2}$$

Pooling Layer

Which outputs should be grouped together?

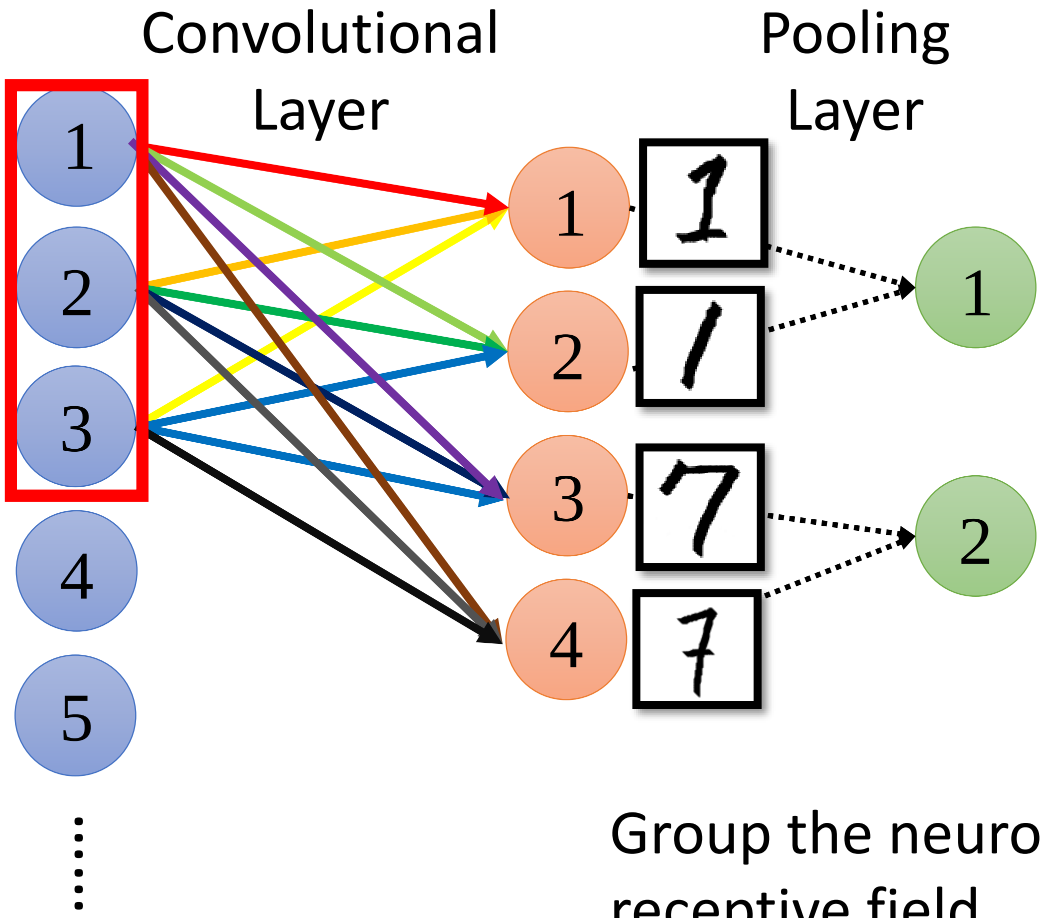


Subsampling

Group the neurons corresponding to the same filter with nearby receptive fields

Pooling Layer

Which outputs should be grouped together?



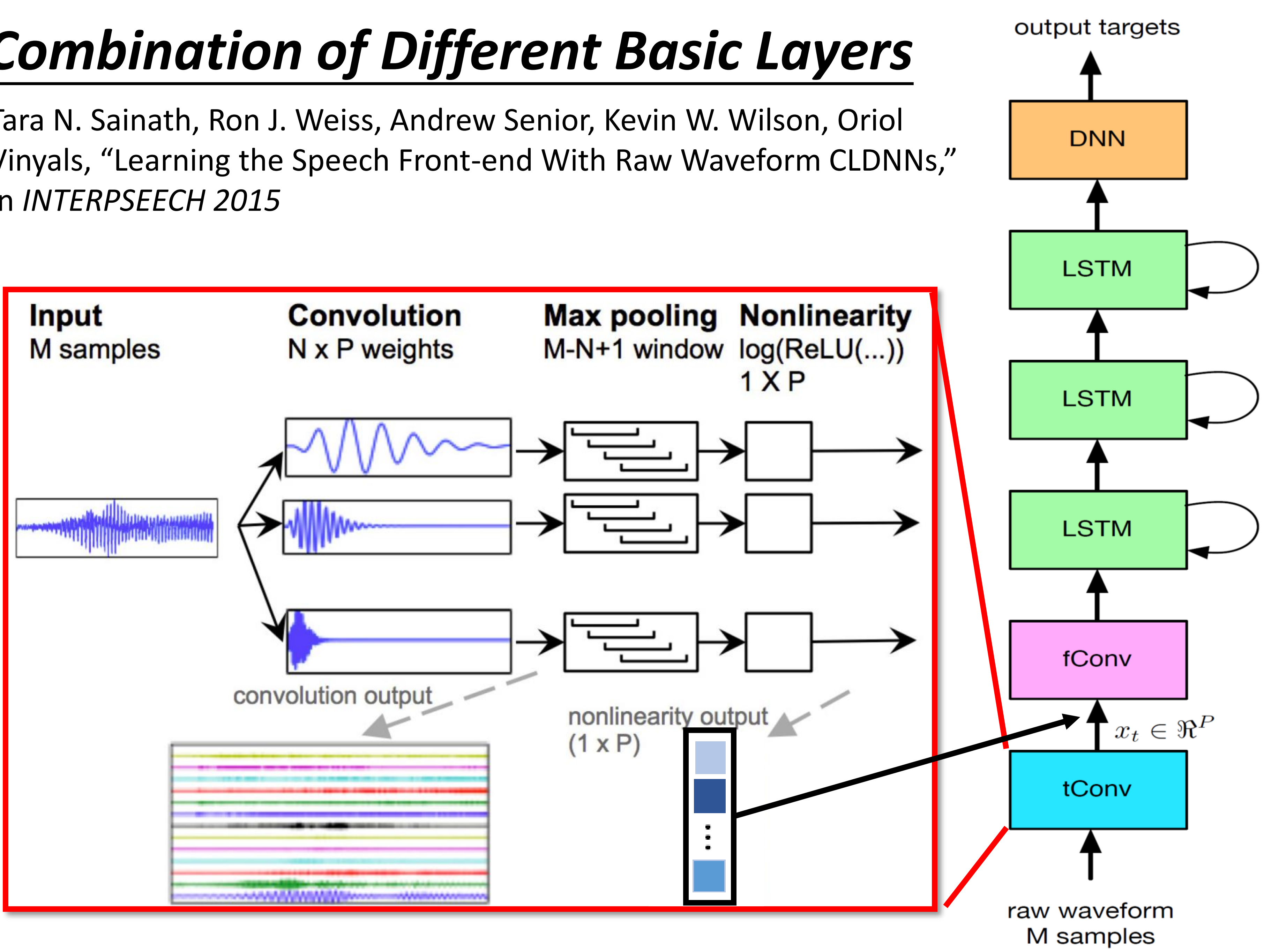
Maxout Network

How can you know whether the neurons detect the same pattern?

Group the neurons with the same receptive field

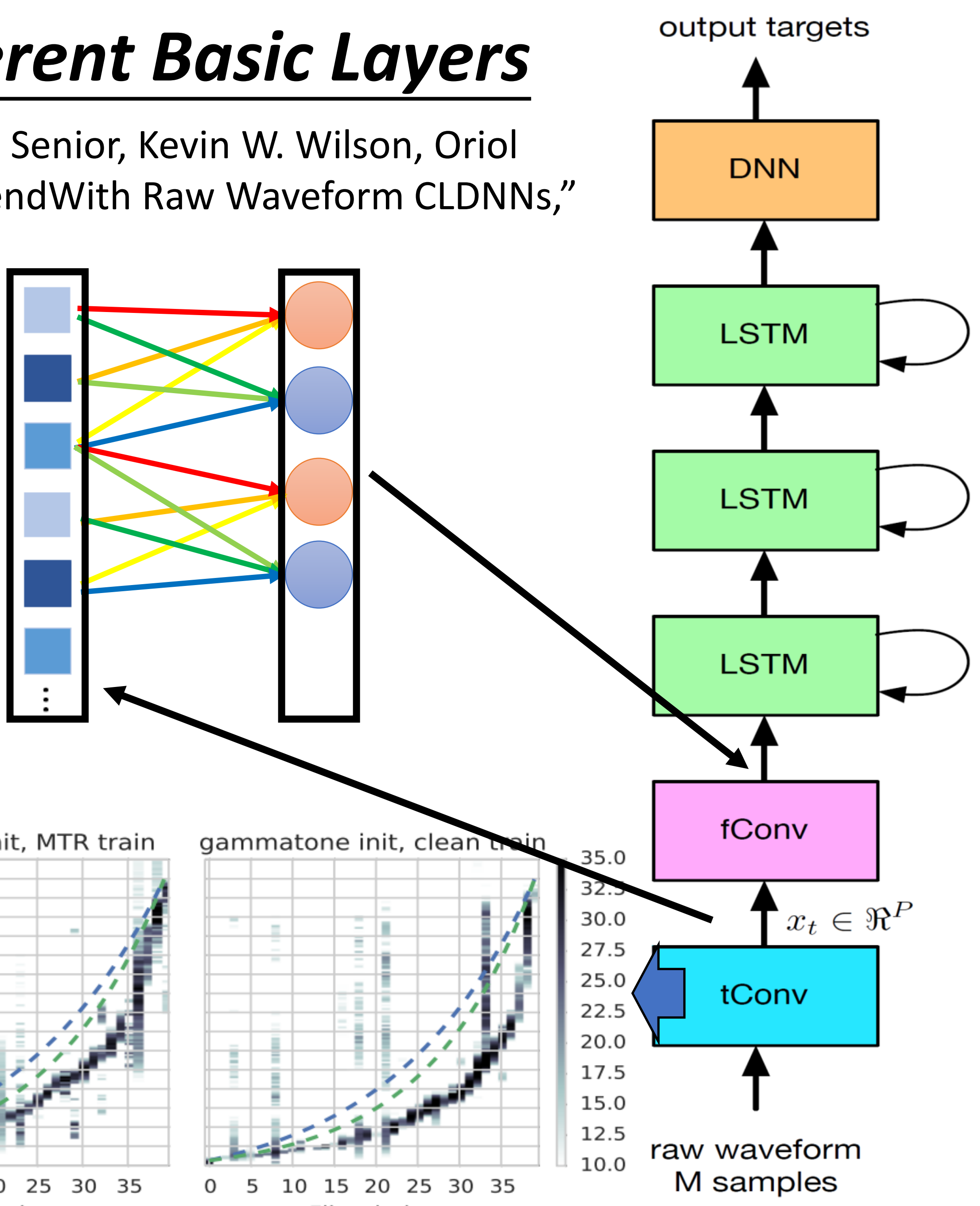
Combination of Different Basic Layers

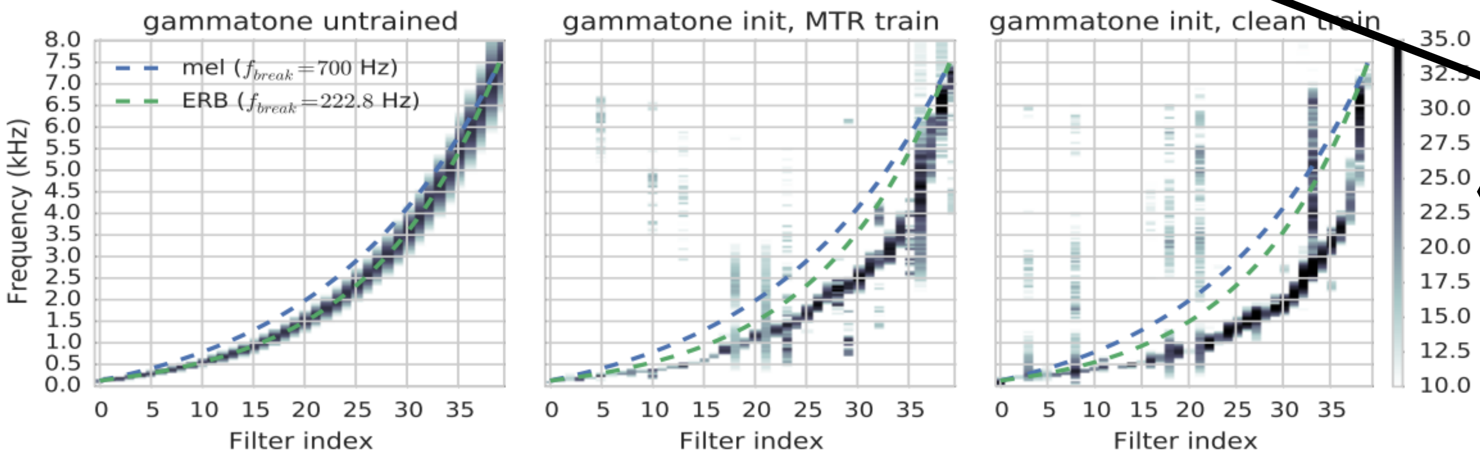
Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, Oriol Vinyals, “Learning the Speech Front-end With Raw Waveform CLDNNs,” In *INTERPSEECH 2015*



Combination of Different Basic Layers

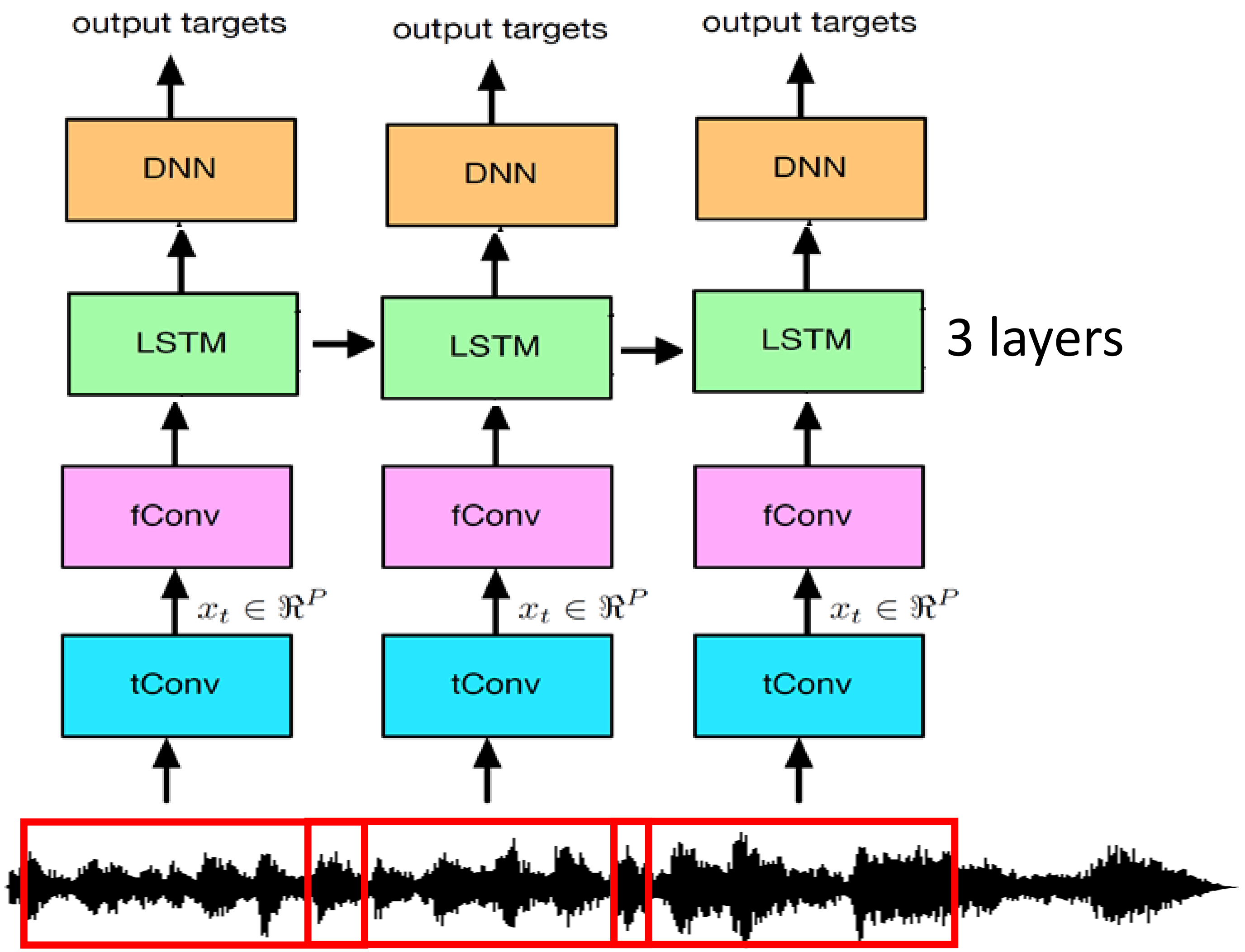
Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, Oriol Vinyals, "Learning the Speech Front-endWith Raw Waveform CLDNNs," In *INTERPSEECH 2015*

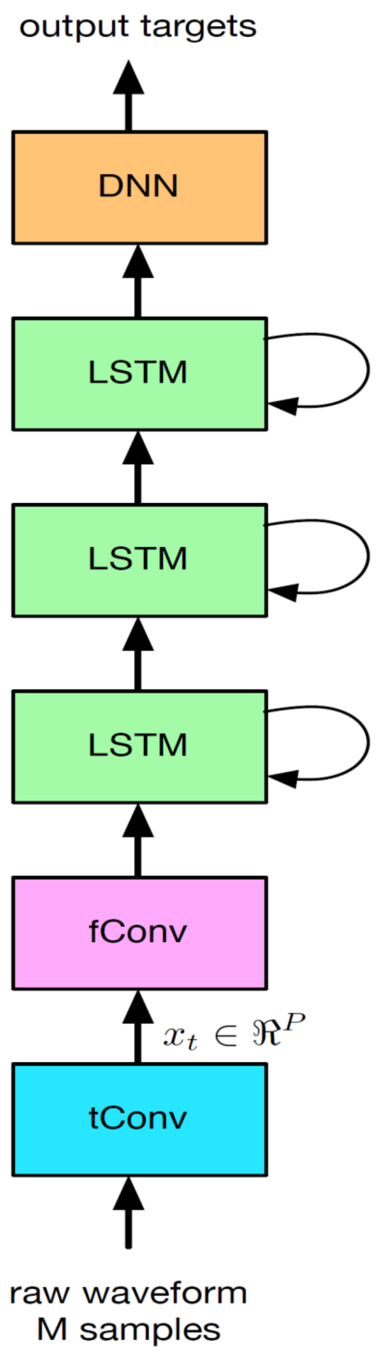




Combination of Different Basic Layers

Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, Oriol Vinyals, "Learning the Speech Front-endWith Raw Waveform CLDNNs," In *INTERPSEECH 2015*





Next Time

- 3/10: TAs will teach TensorFlow
  - TensorFlow for regression
  - TensorFlow for word vector
    - word vector: https://www.youtube.com/watch?v=X7PH3NuYW0Q
  - TensorFlow for CNN
- If you want to learn Theano
  - http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html
  - http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20RNN.ecm.mp4/index.html