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

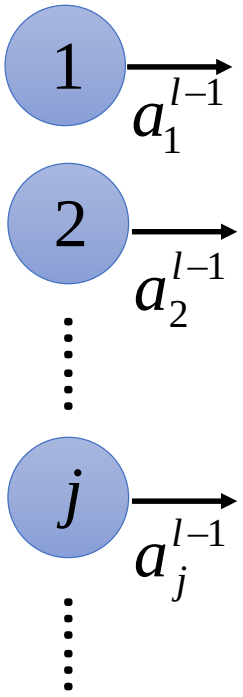
- Basic structure (3/03)
  - Fully Connected Layer
  - Recurrent Structure
  - Convolutional/Pooling Layer
- Special Structure (3/17)
  - Spatial Transformation Layer
  - Highway Network / Grid LSTM
  - Recursive Structure
  - Batch Normalization
  - Sequence-to-sequence / Attention (3/24)

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

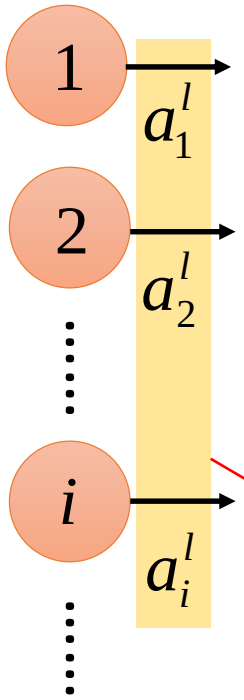
Basic Structure:
Fully Connected Layer

Fully Connected Layer



Layer $l-1$

N_{l-1} nodes



Layer l

N_l nodes

Output of a neuron:

a_i^l — Layer l; Neuron i

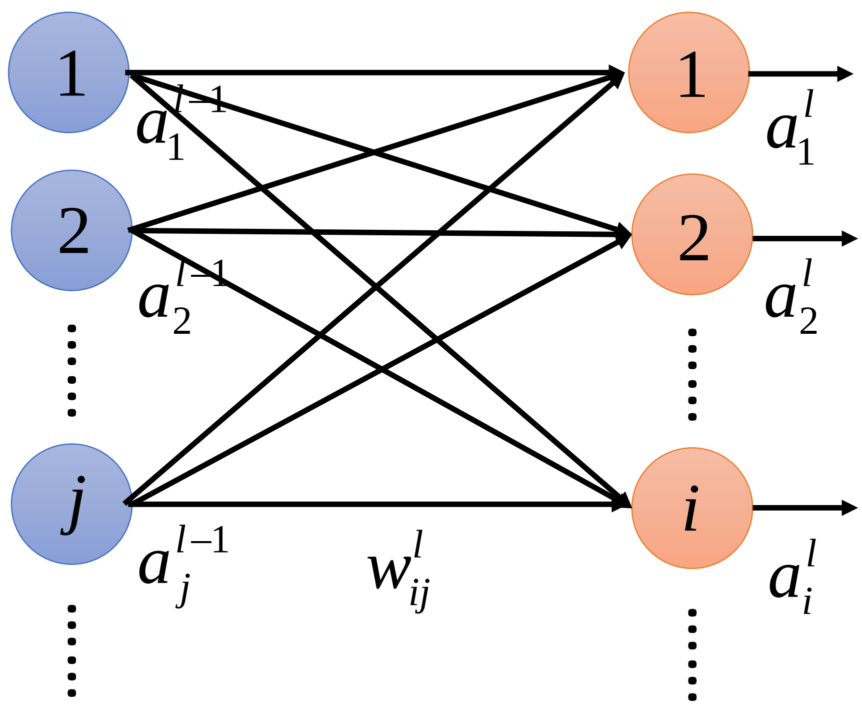
Output of one layer:

a^l : a vector

a(l)i代表第l层第i个神经元地输出
a（l）代表第l层地输出

W(l)ij表示第L-1层地第j个神经元到L层第i个神经元地输出。（注意j在后面）

Fully Connected Layer



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

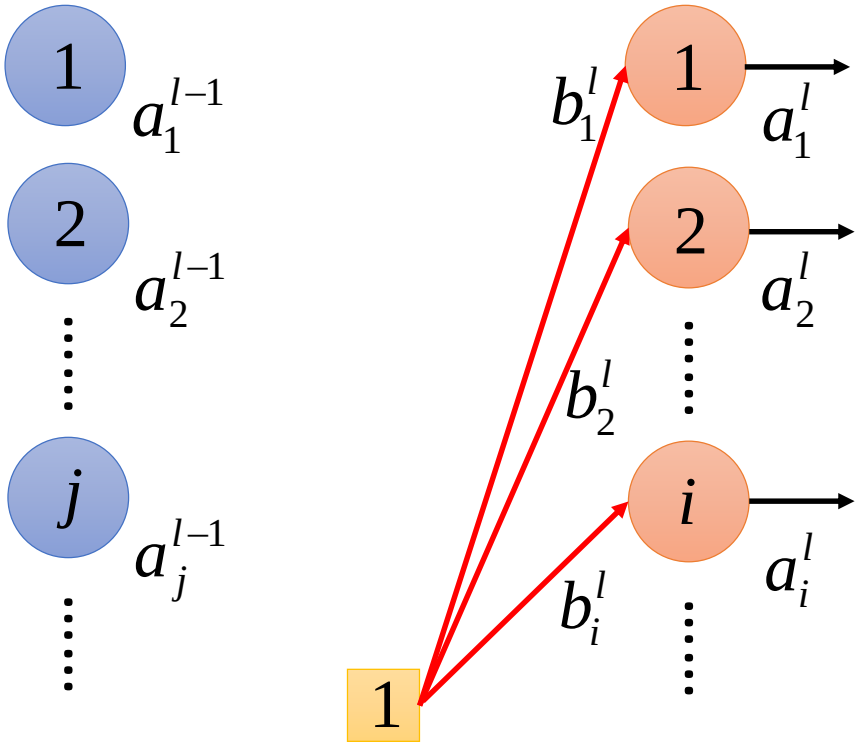
w_{ij}^l → Layer $l-1$ to Layer l

w_{ij}^l → from neuron j (Layer $l-1$) to neuron i (Layer l)

$$W^l = \underbrace{\begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix}}_{N_{l-1}} \Bigg\} N_l$$

把l层第i个神经元输入地权重放在一行，所有神经元地参数依次堆叠，则得到了W(l).行数等于l层神经元数量，列数等于l-1层神经元数量。

Fully Connected Layer



Layer $l-1$

N_{l-1} nodes

Layer l

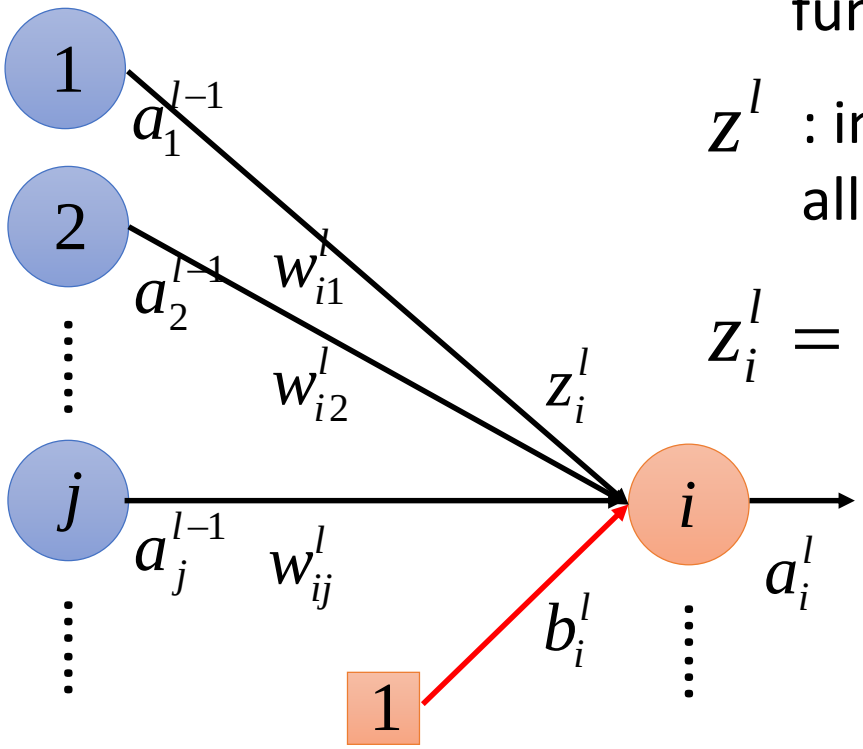
N_l nodes

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

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

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

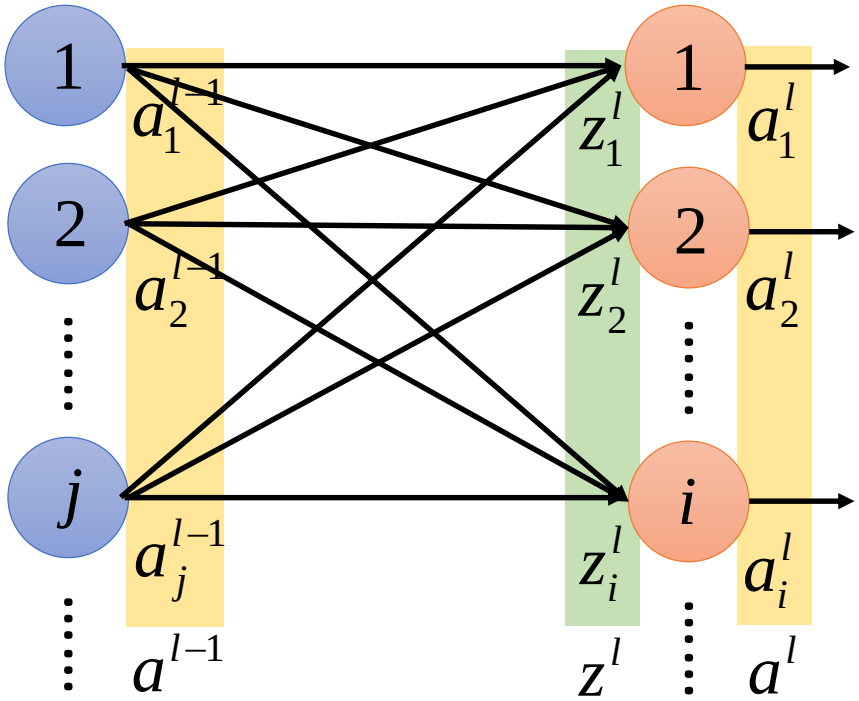
Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

Relations between Layer Outputs



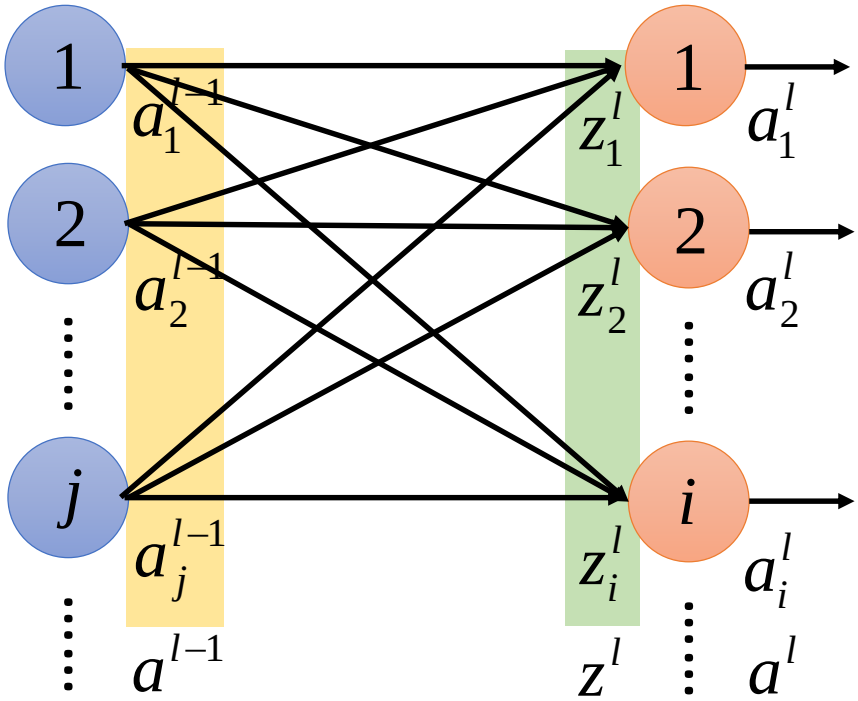
Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

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

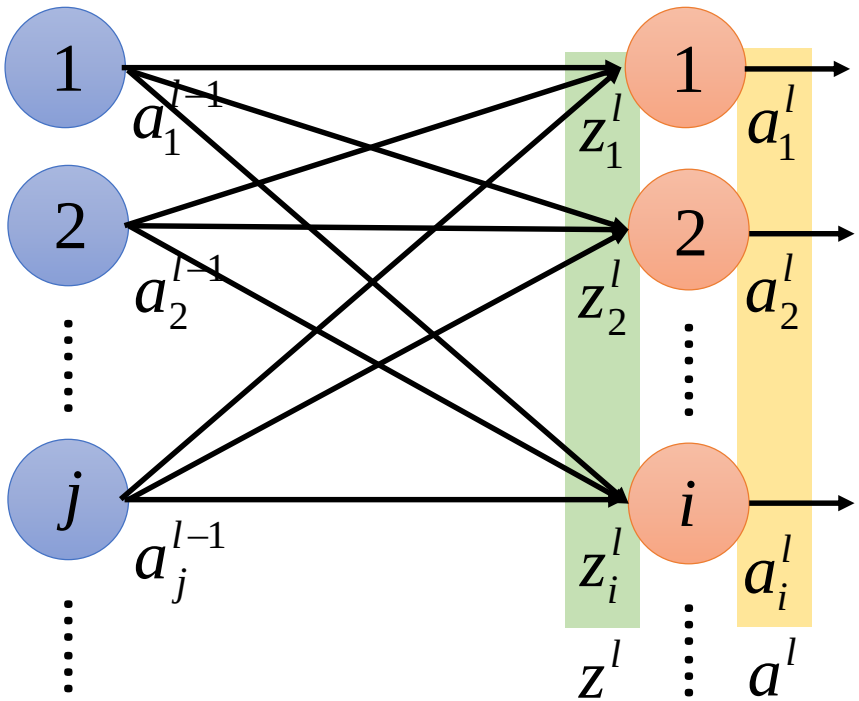
$$\vdots$$

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

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

前一层地输出与后一层地输入地关系。

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

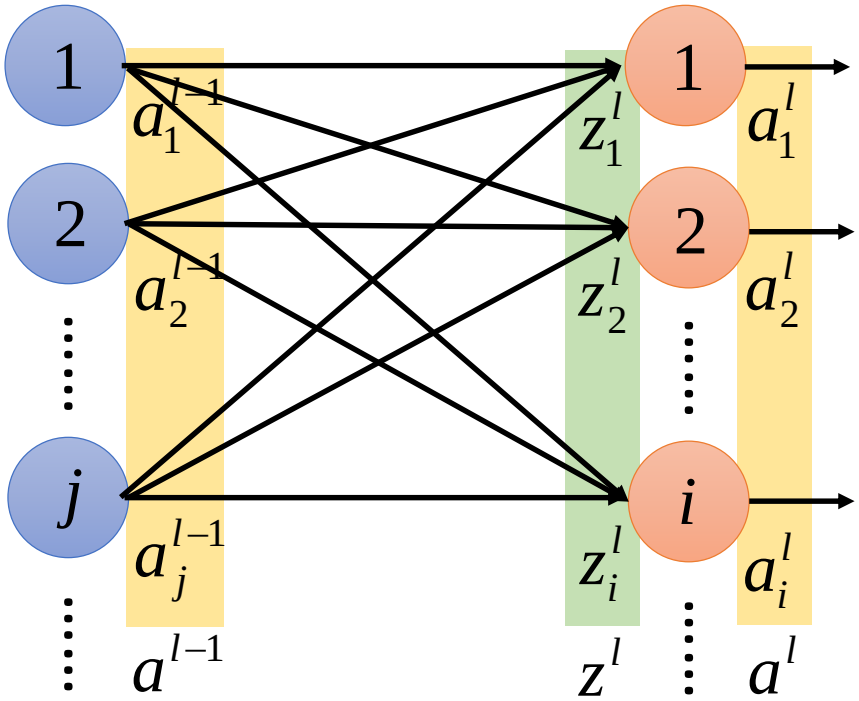
N_l nodes

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

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

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

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

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

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

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

RNN主要用在语言相关地上面，不看

Basic Structure:
Recurrent Structure

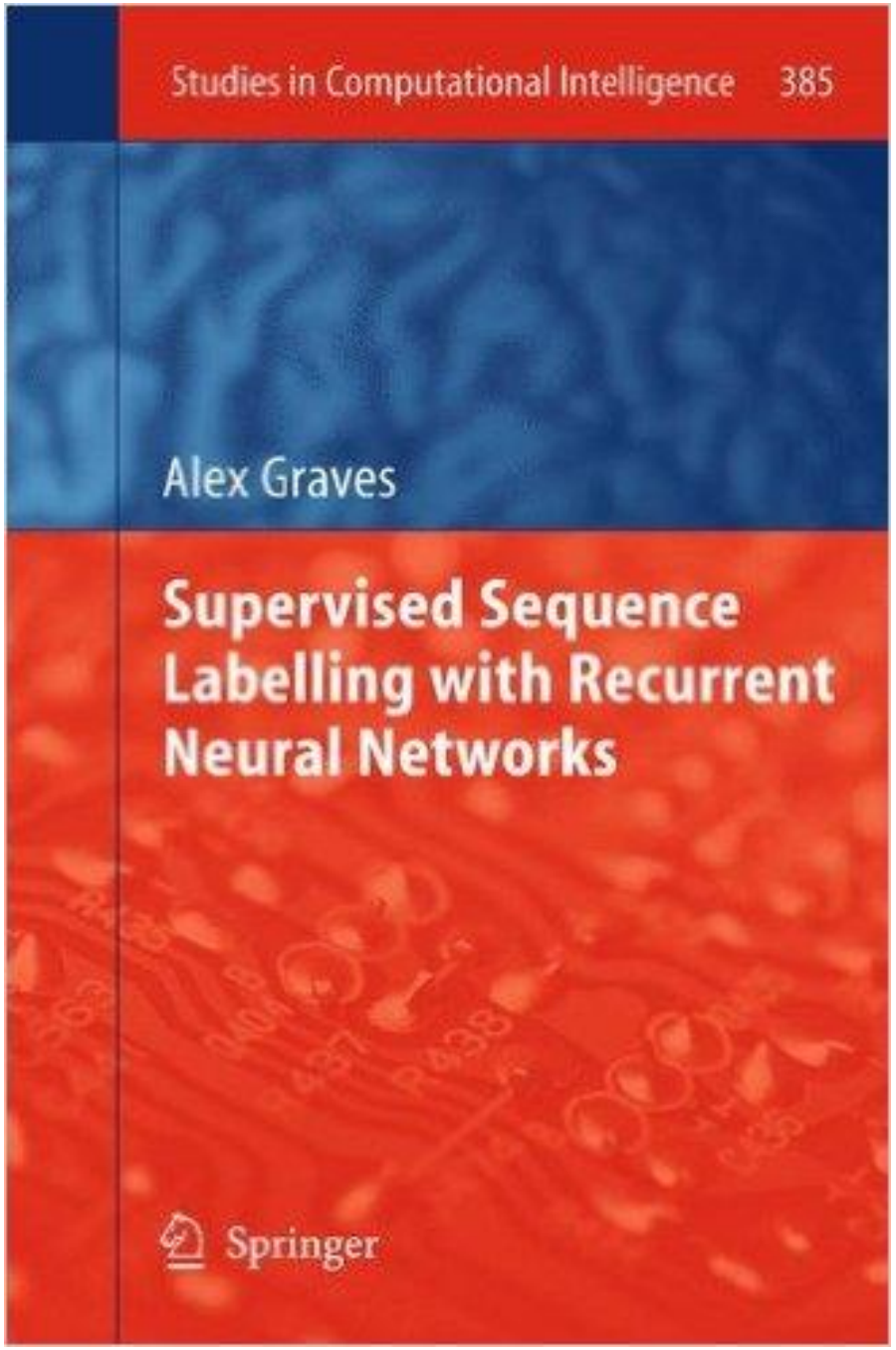
Simplify the network
by using the same function again and again

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

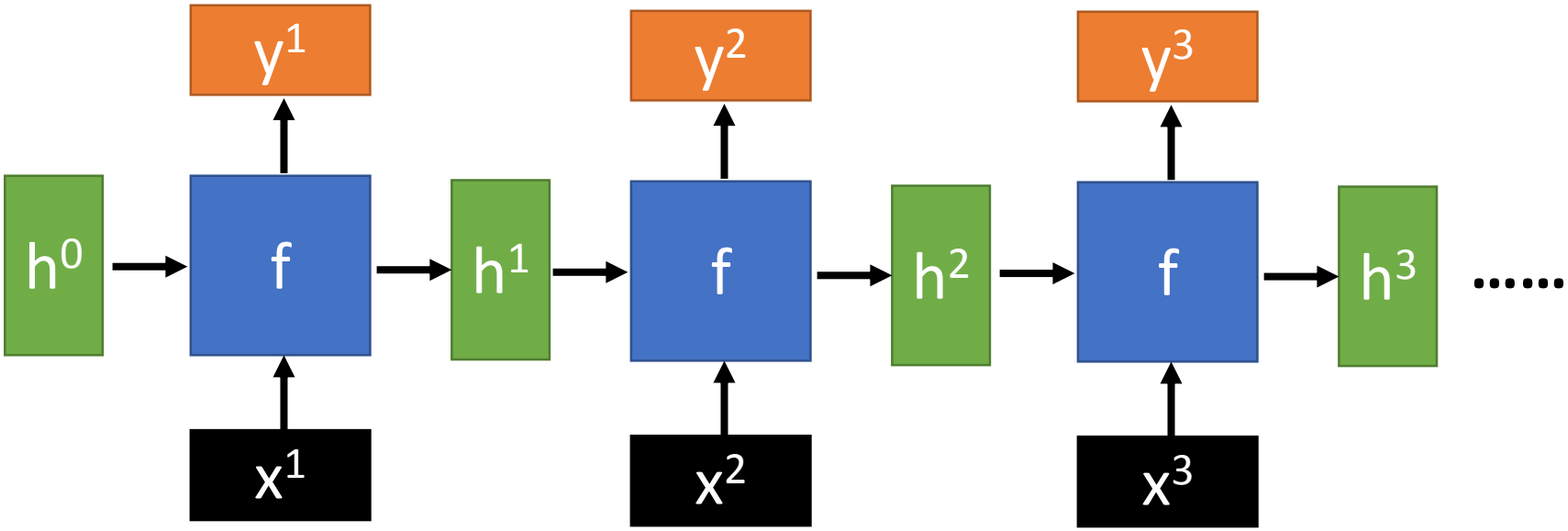
https://www.cs.toronto.edu/~graves/preprint.pdf



Recurrent Neural Network

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

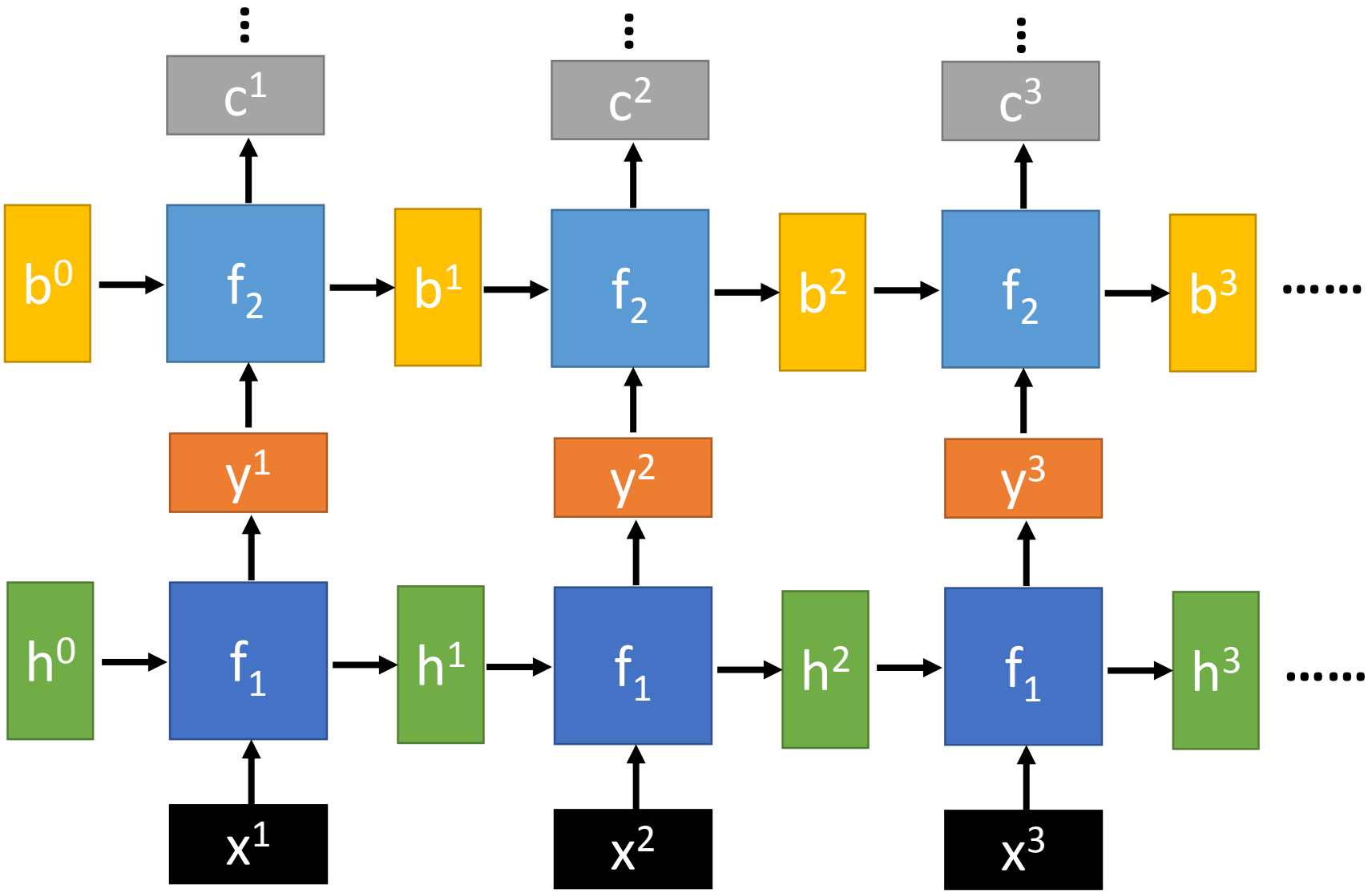
h and h' are vectors with the same dimension



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

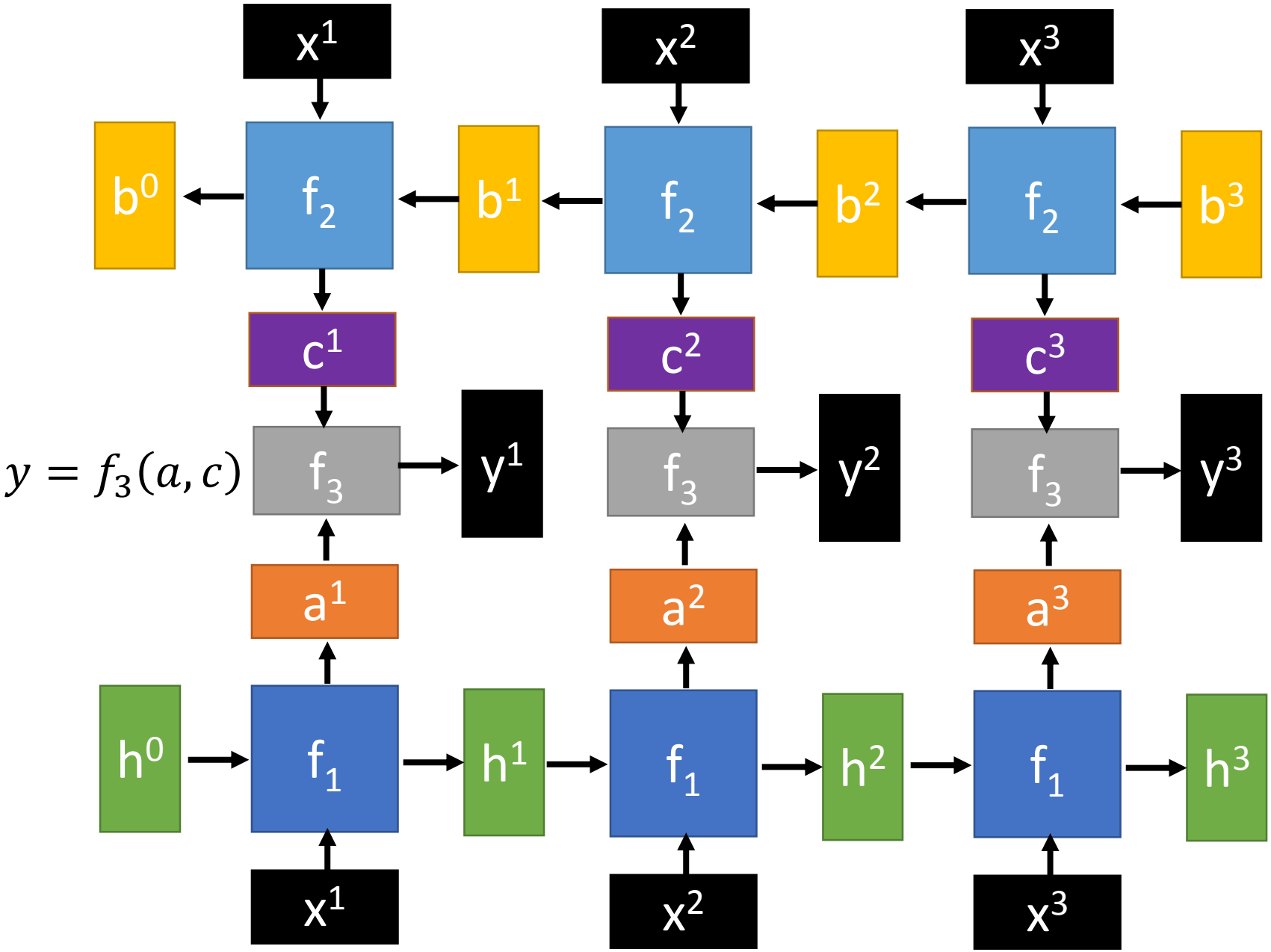
Deep RNN

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



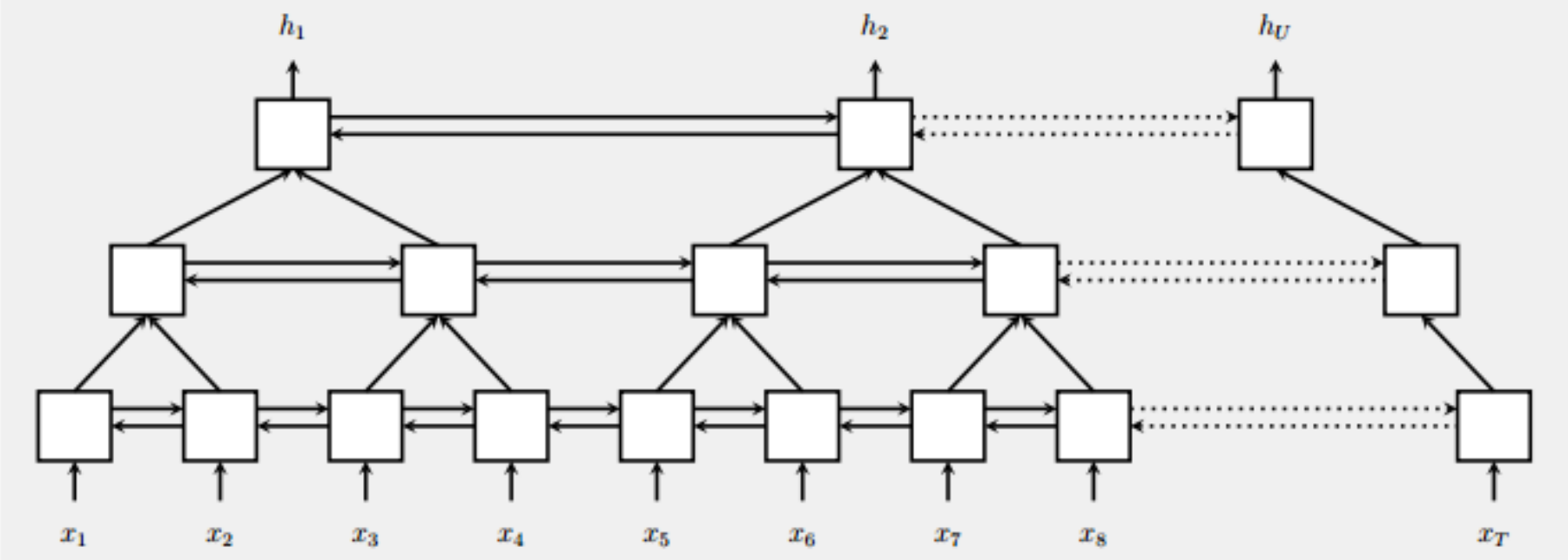
Bidirectional RNN

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



Pyramidal RNN

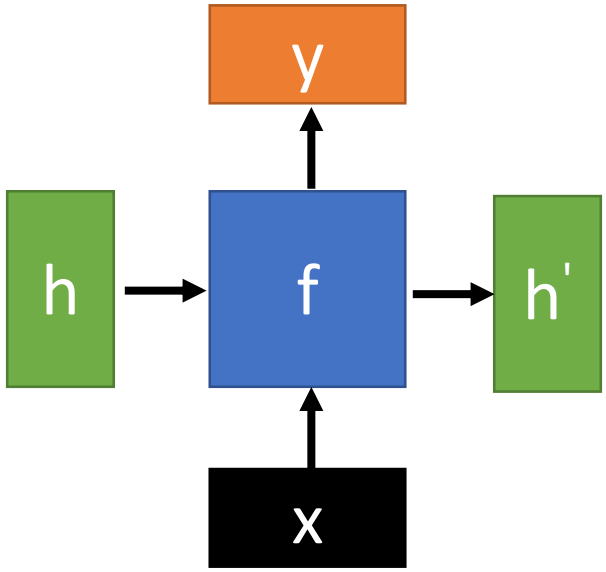
- Reducing the number of time steps



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: $h', y = f(h, x)$



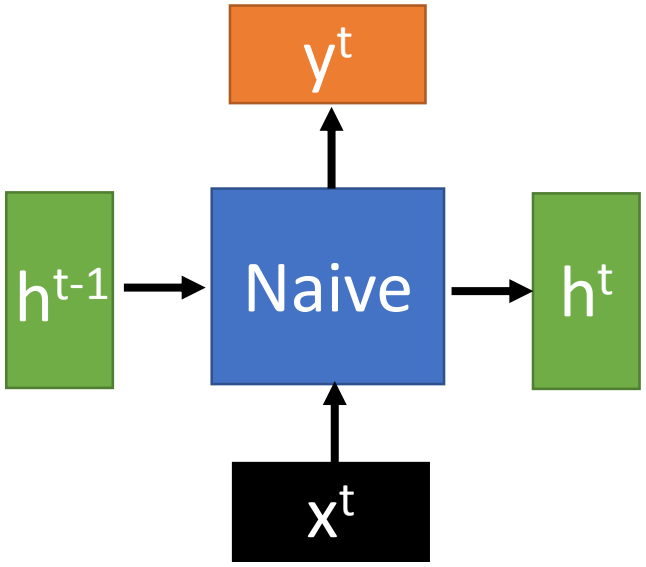
$$h' = \sigma\left(W^h h + W^i x \right)$$

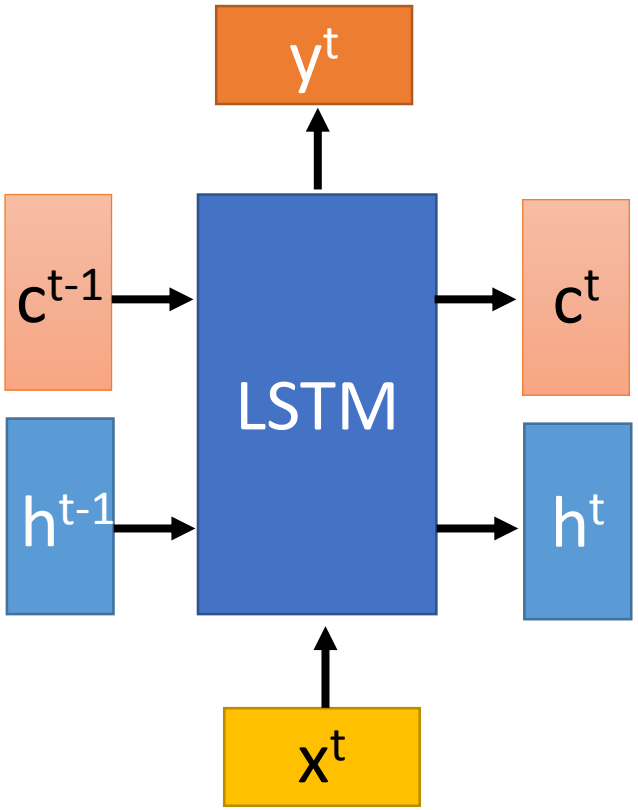
$$y = \sigma\left(W^o h' \right)$$

softmax

Ignore bias here

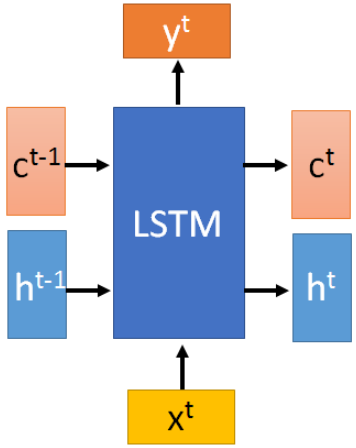
LSTM



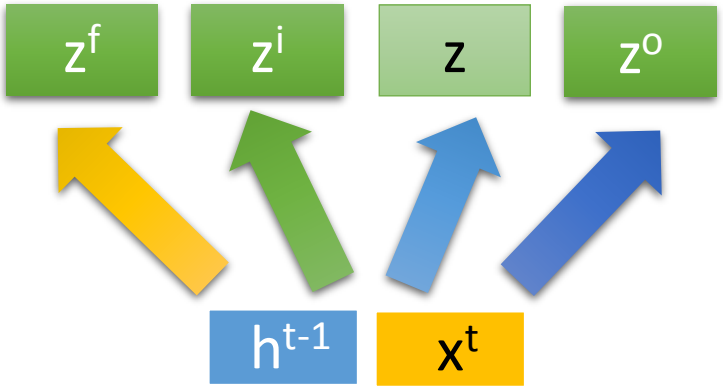


c change slowly → c^t is c^{t-1} added by something

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



c^{t-1}

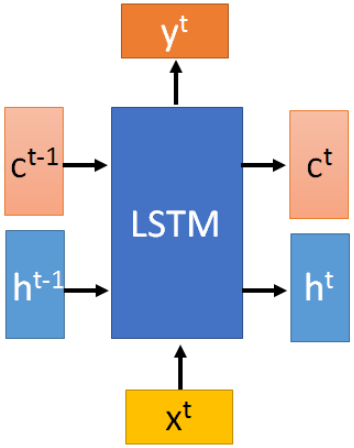


$$z = tanh\left(W \begin{matrix} x^t \\ h^{t-1} \end{matrix} \right)$$

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

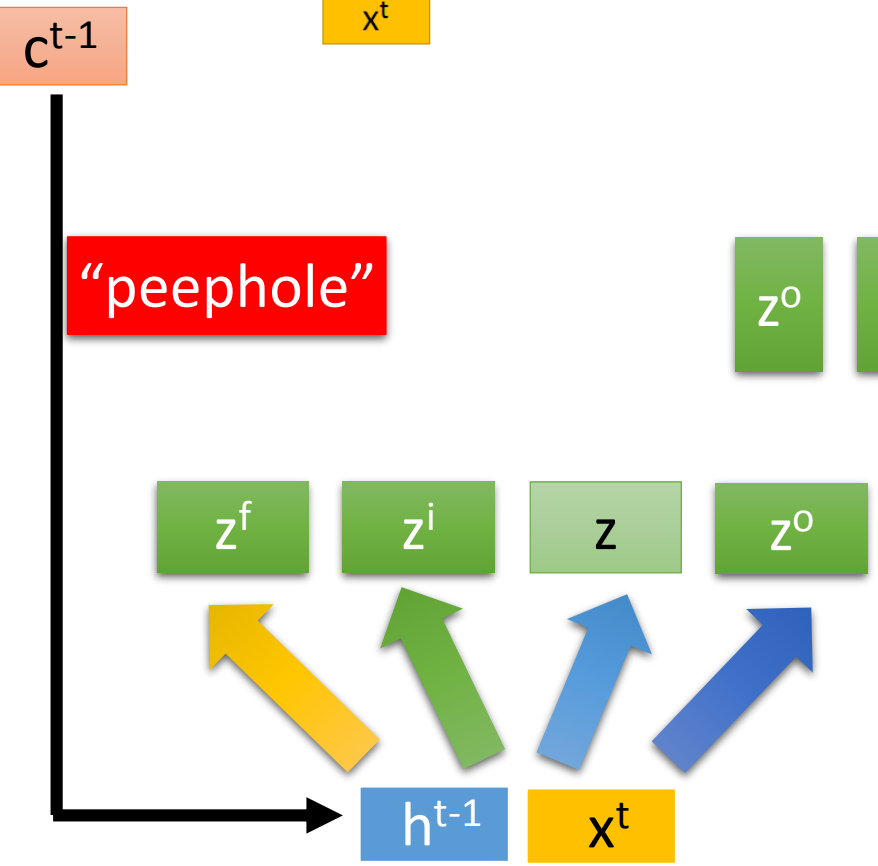
$$z^f = \sigma\left(W^f \begin{matrix} x^t \\ h^{t-1} \end{matrix} \right)$$

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

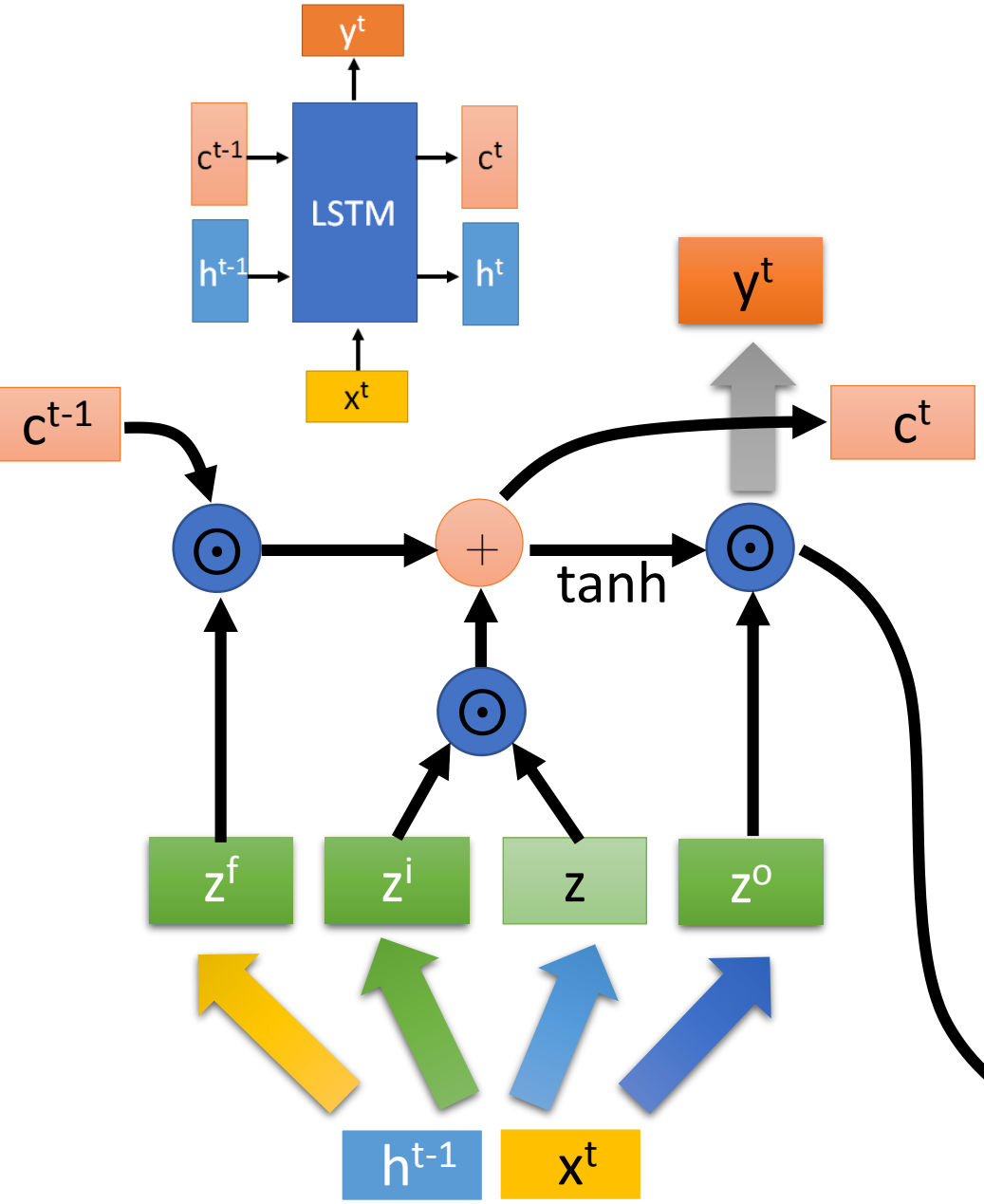


$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \\ c^{t-1} \end{bmatrix} \right)$$

diagonal



z^o z^f z^i obtained by the same way

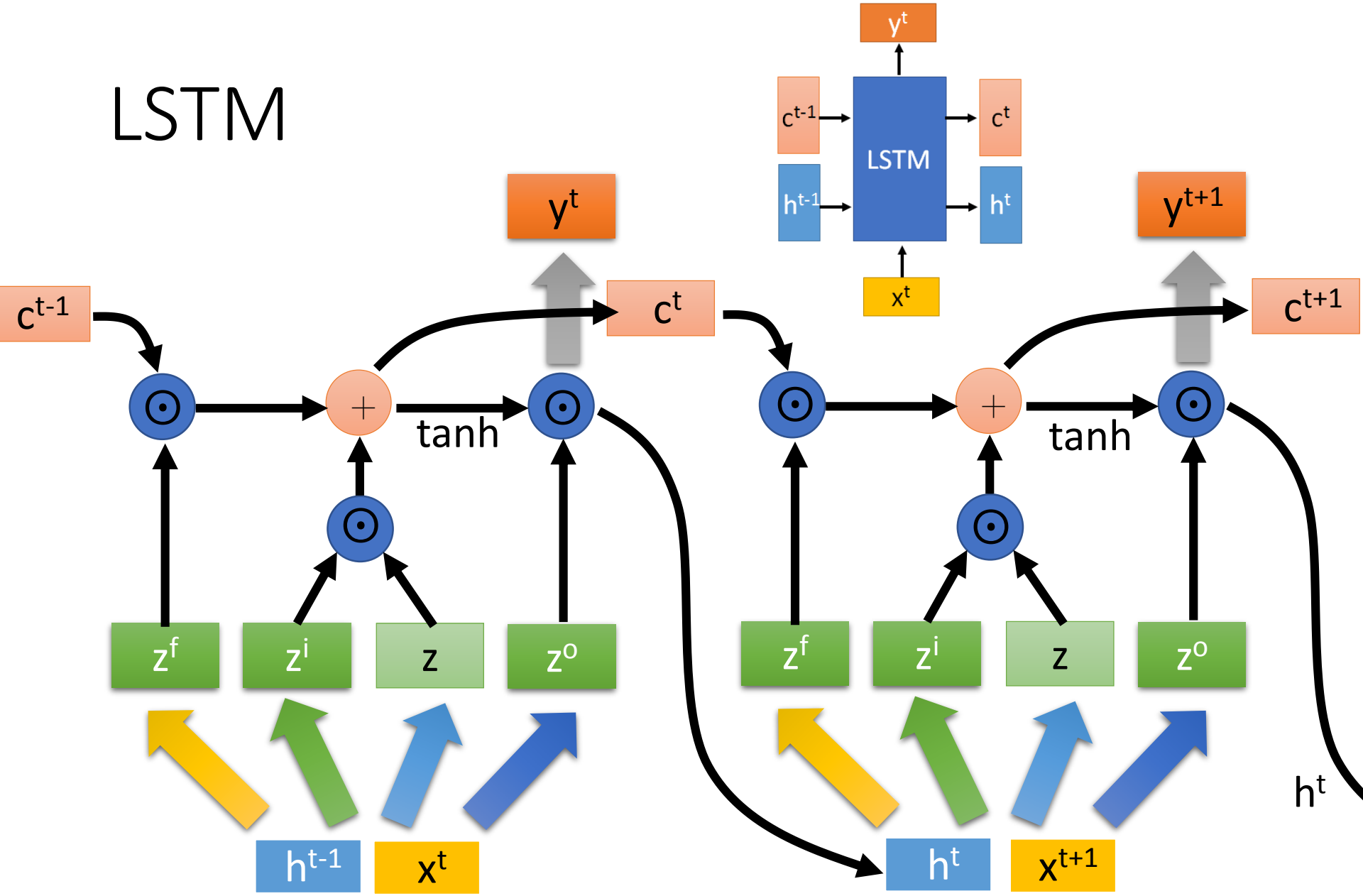


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

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

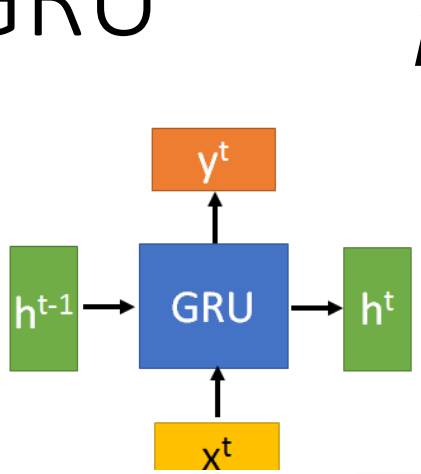
$$y^t = \sigma(W' h^t)$$

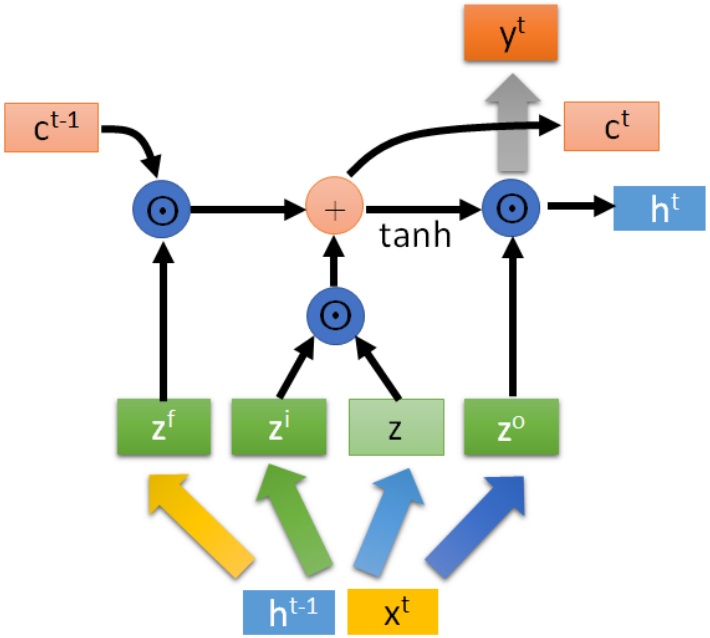
LSTM

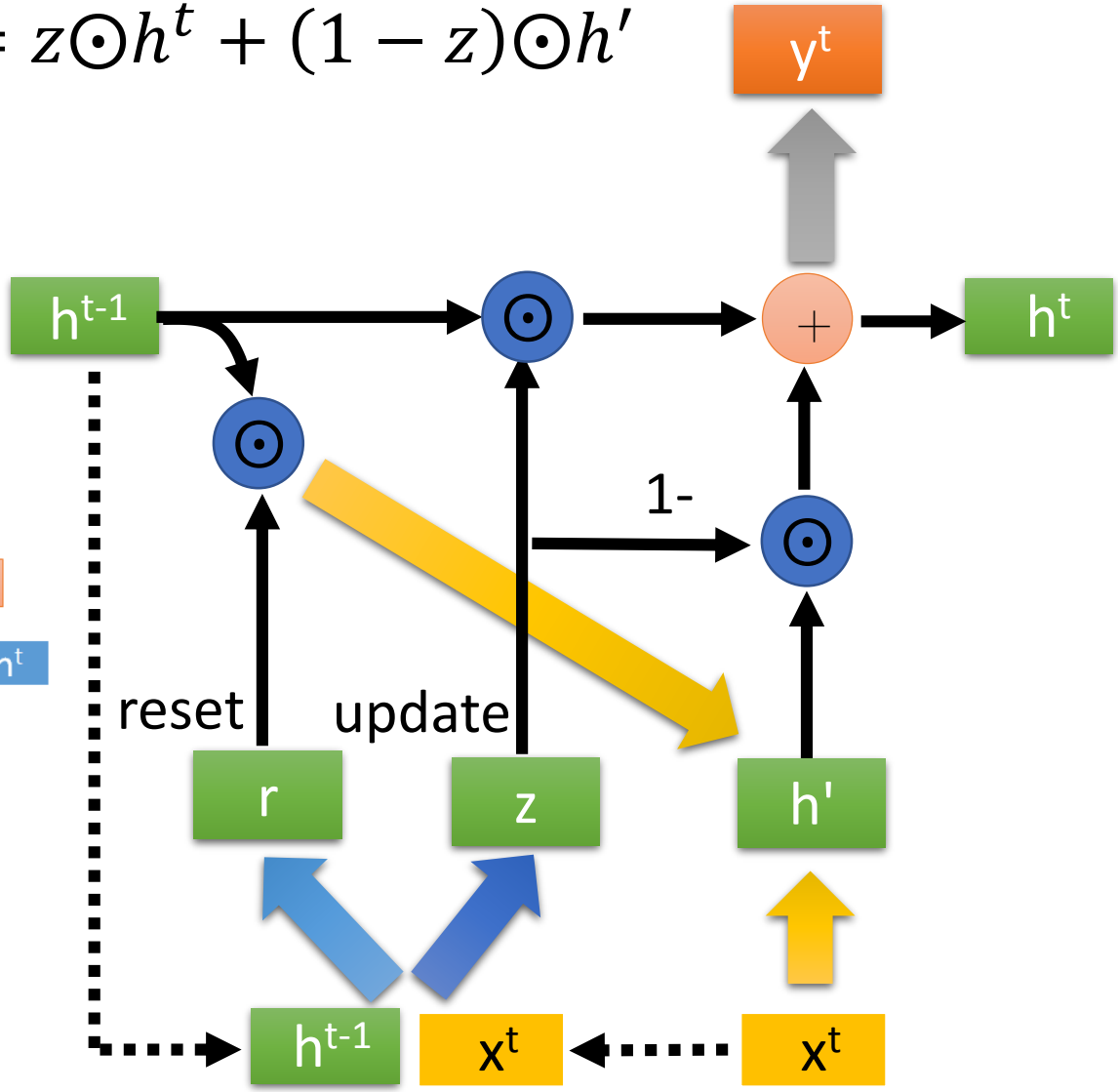

y^t
c^{t-1}
LSTM
c^t
h^{t-1}
h^t
x^t
y^t
y^{t+1}
c^{t-1}
c^t
c^{t+1}
tanh
tanh
z^f
z^i
z
z^o
z^f
z^i
z
z^o
h^{t-1}
x^t
h^t
x^{t+1}
h^t

GRU

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

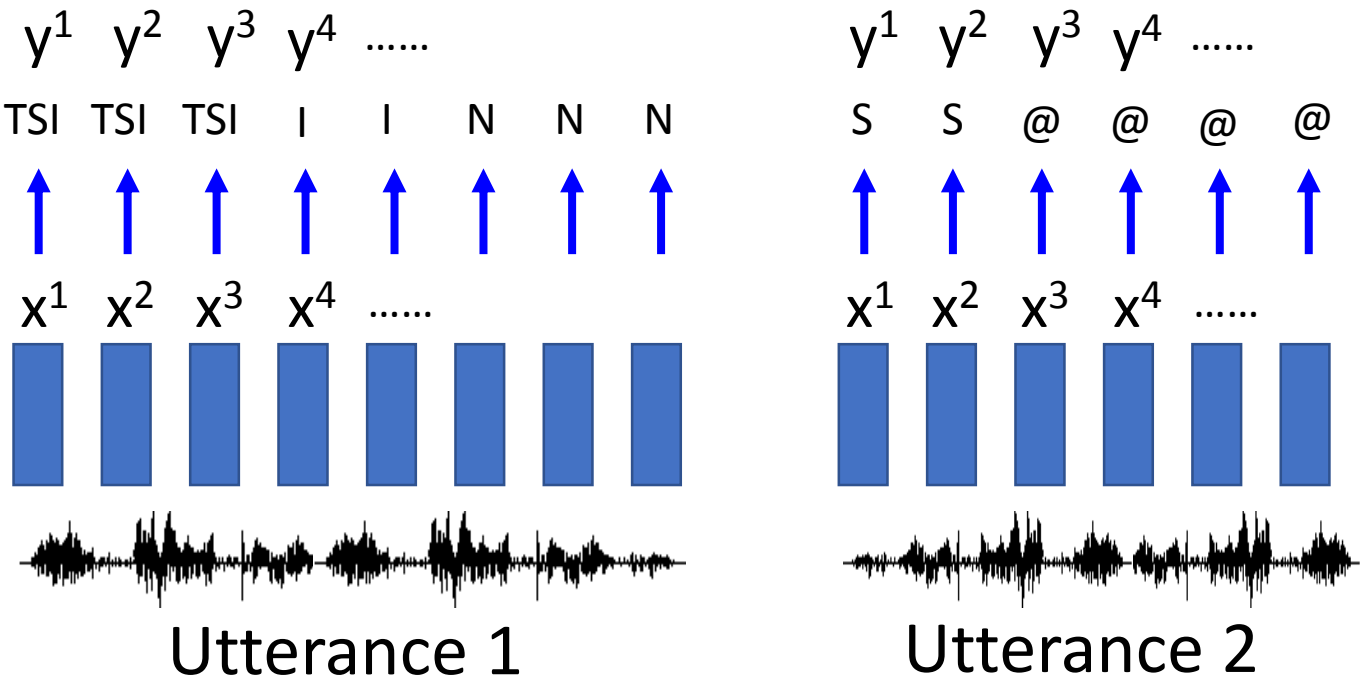






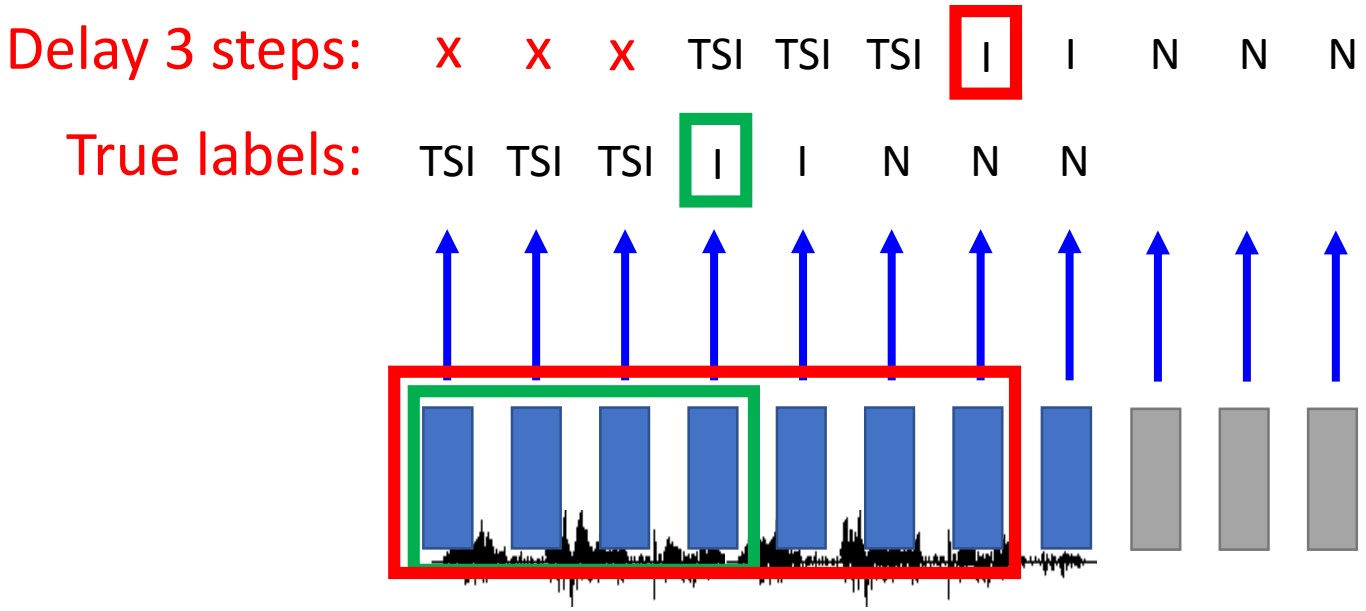
Example Task

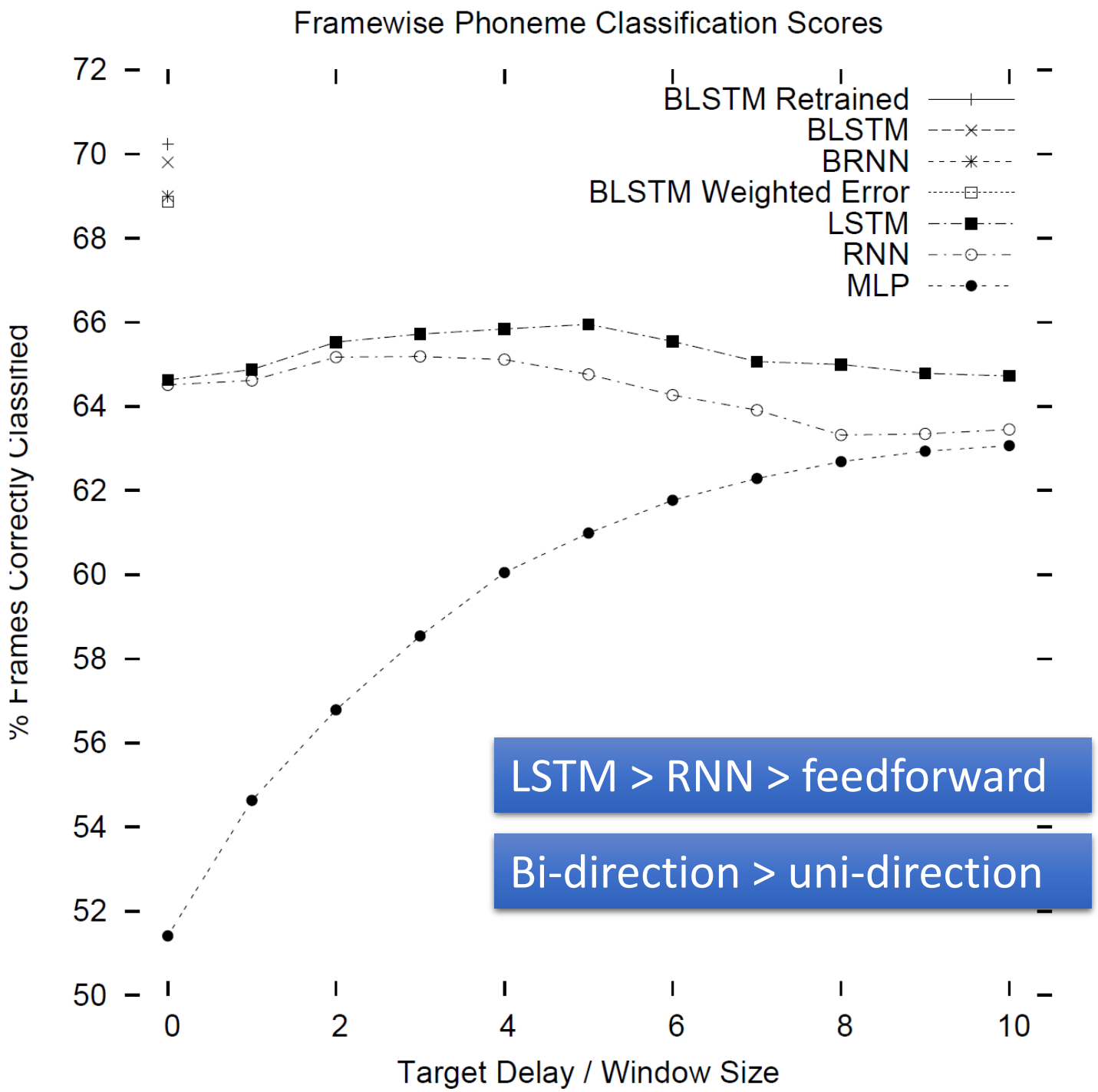
- (Simplified) Speech Recognition: Frame classification on TIMIT



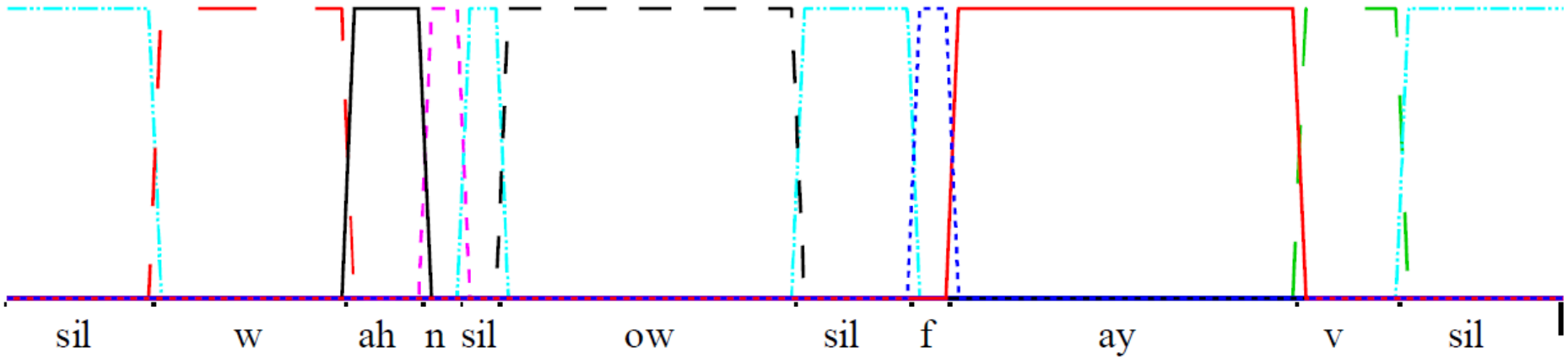
Target Delay

- Only for unidirectional RNN



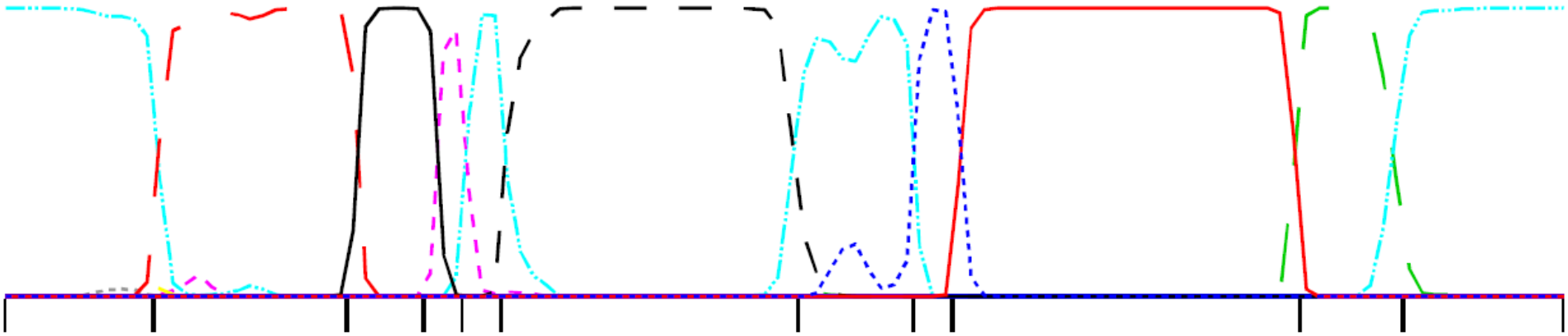

Framewise Phoneme Classification Scores
% Frames Correctly Classified
Target Delay / Window Size
BLSTM Retrained
BLSTM
BRNN
BLSTM Weighted Error
LSTM
RNN
MLP
72
70
68
66
64
62
60
58
56
54
52
50
0
2
4
6
8
10
LSTM > RNN > feedforward
Bi-direction > uni-direction

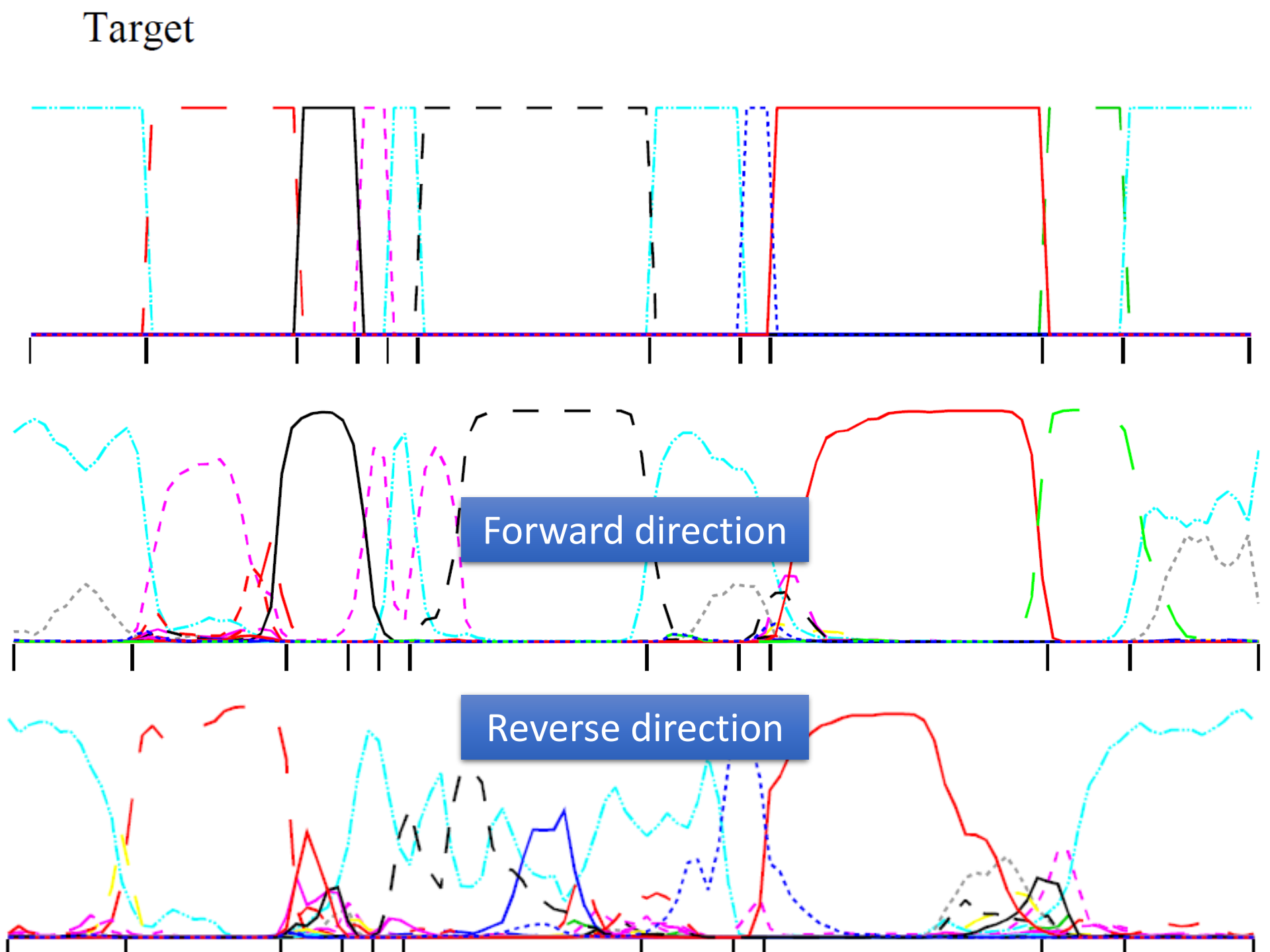
Target

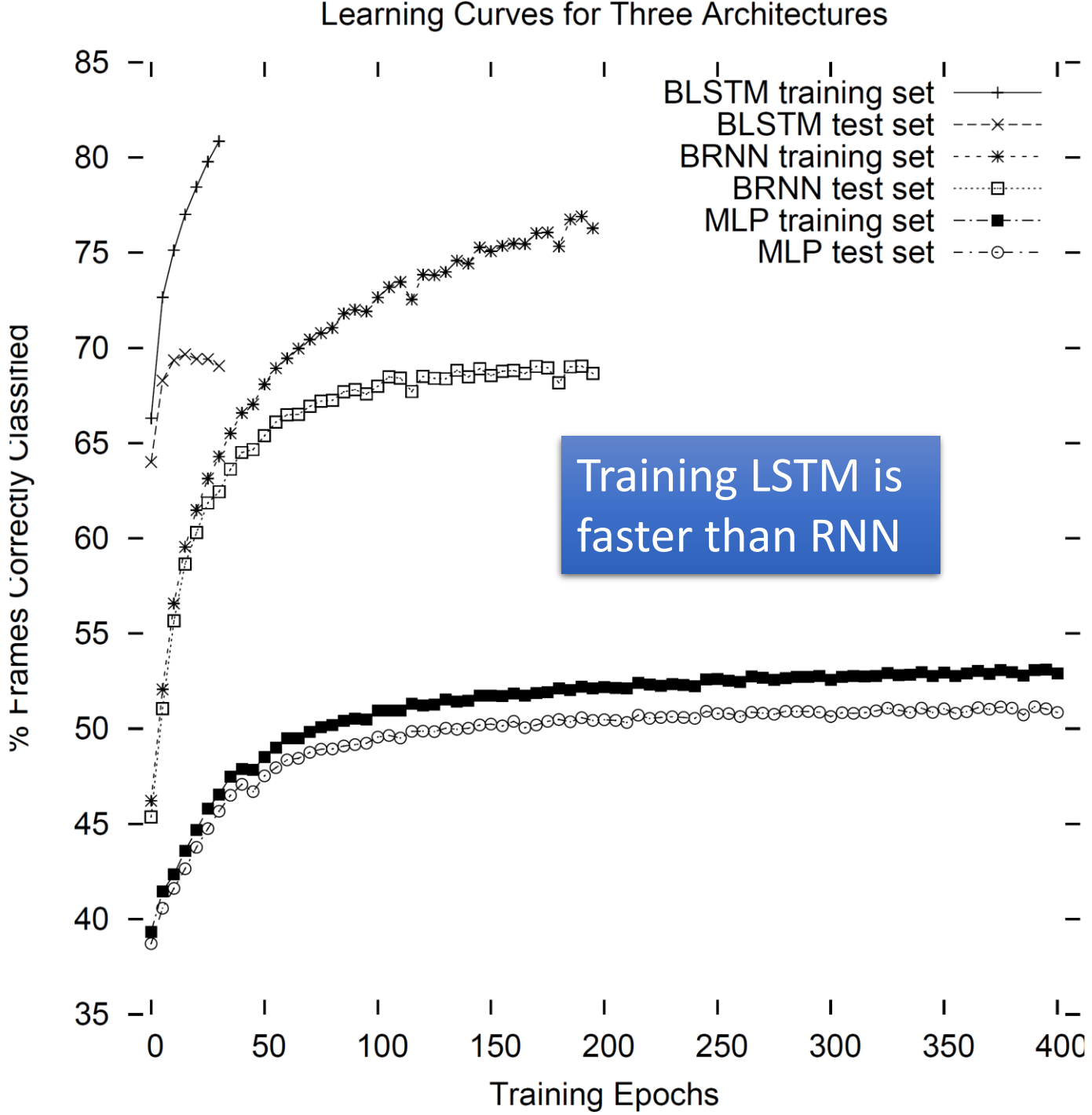


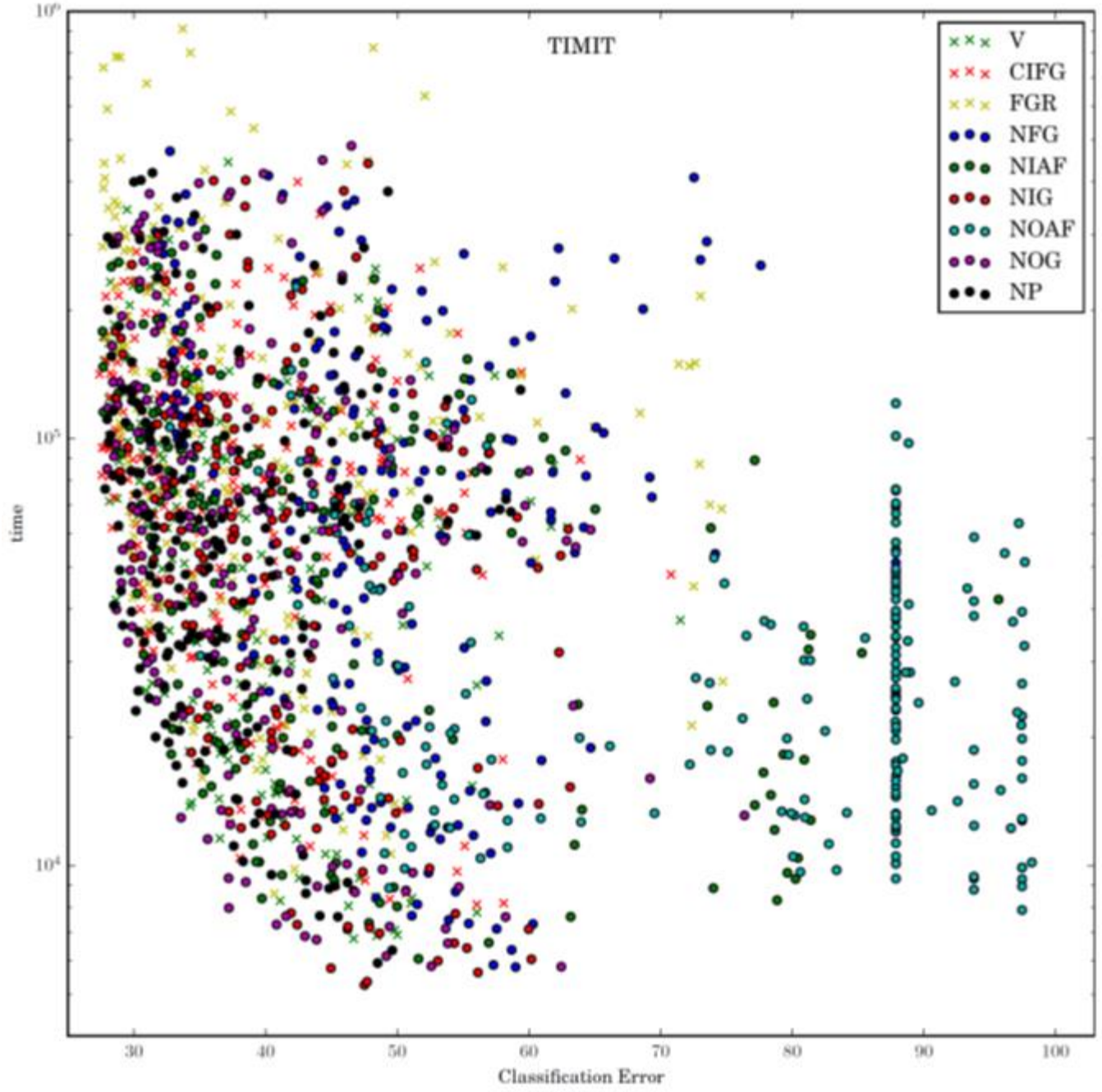
one oh five

Bidirectional Output



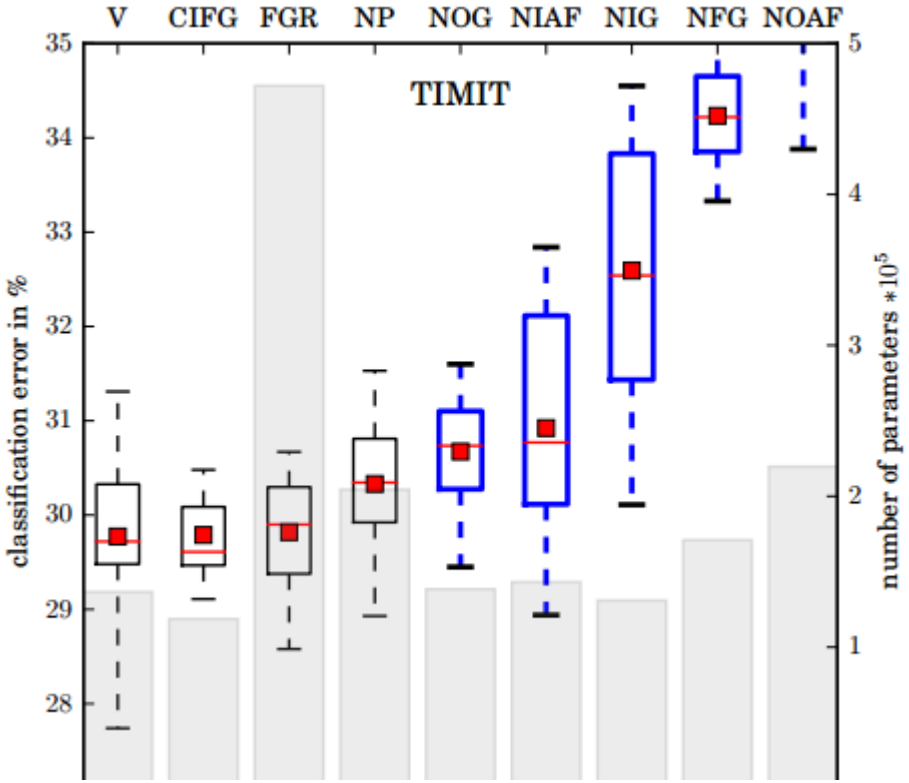

Target
Forward direction
Reverse direction


Learning Curves for Three Architectures
% Frames Correctly Classified
Training Epochs
BLSTM training set
BLSTM test set
BRNN training set
BRNN test set
MLP training set
MLP test set
Training LSTM is faster than RNN
85
80
75
70
65
60
55
50
45
40
35
0
50
100
150
200
250
300
350
400


TIMIT
V
CIFG
FGR
NFG
NIAF
NIG
NOAF
NOG
NP
time
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:

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

MUT2:

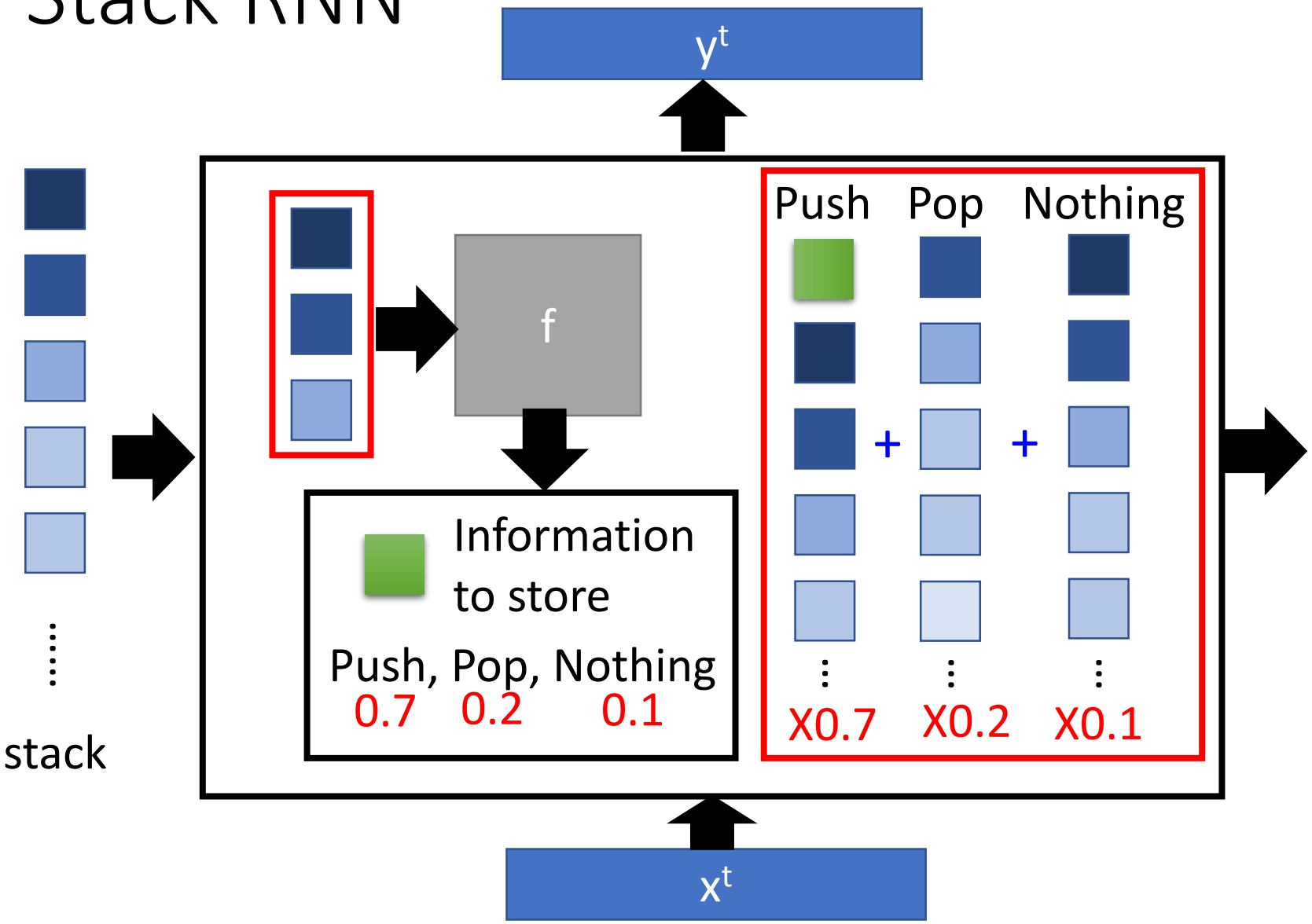
$$
\begin{aligned}
z &= \text{sigm}(W_{\text{xz}} x_t + W_{\text{hz}} h_t + b_{\text{z}}) \\
r &= \text{sigm}(x_t + W_{\text{hr}} h_t + b_{\text{r}}) \\
h_{t+1} &= \tanh(W_{\text{hh}}(r \odot h_t) + W_{xh} x_t + b_{\text{h}}) \odot z \\
&+ \quad h_t \odot (1 - z)
\end{aligned}
$$

MUT3:

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

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

Stack RNN

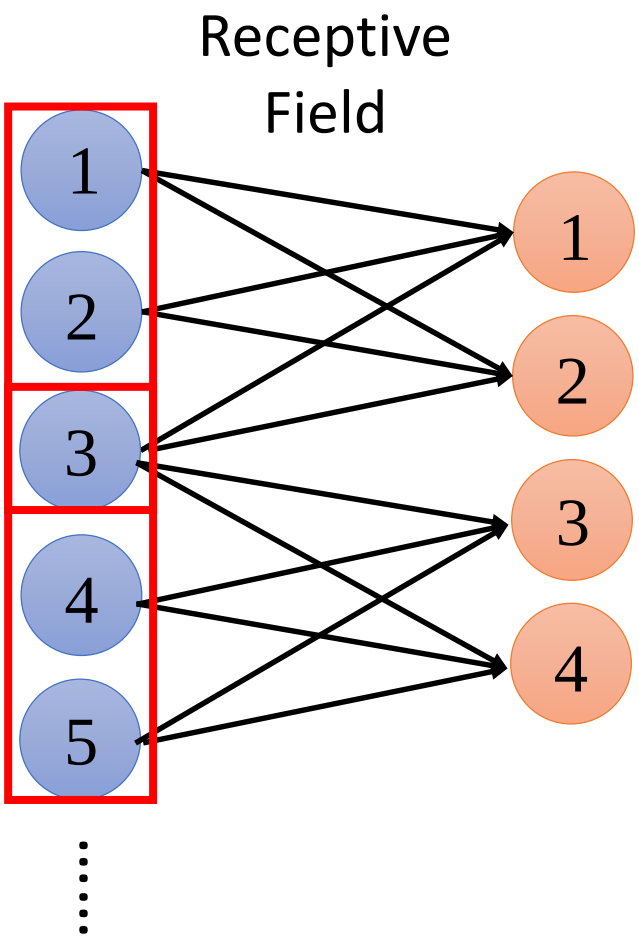


Basic Structure:
Convolutional / Pooing Layer

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

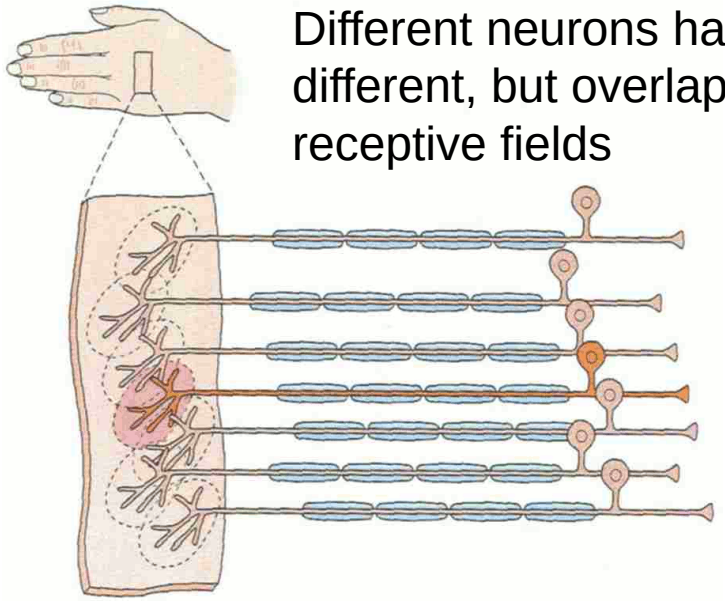
在生物中，每一个神经元只接受皮肤上一部分区域地感受，并不会接受所有所有皮肤地感受，每个神经元接受地区域叫做这个神经元地感受域（receotive field）。
将感受域地概念引入CNN中，卷积操作中，与后一层地神经元a相连地前一层地神经元叫做a的感受域。换句话说，后一层的神经元与前一层是稀疏连接的，也就是sparse connectivity。

Convolutional Layer



Sparse Connectivity

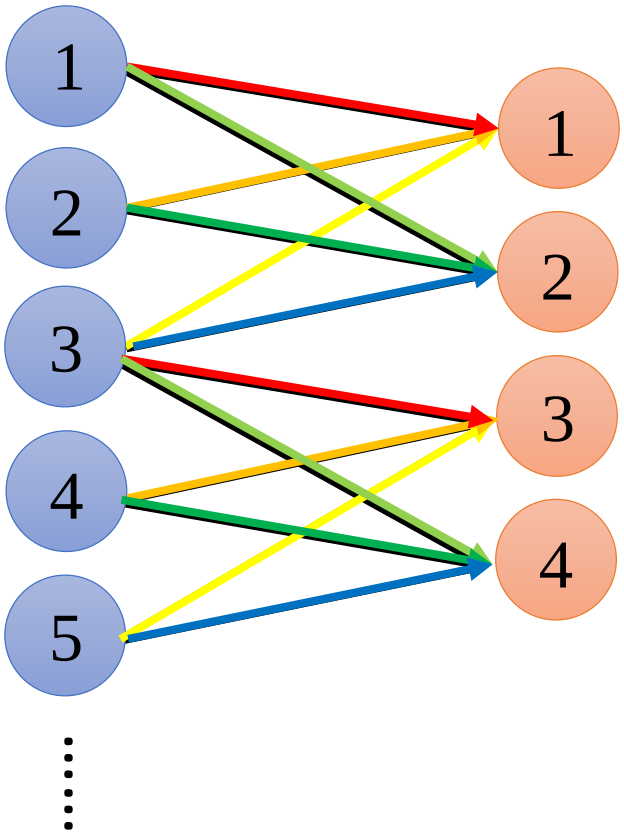
Each neural only connects to part of the output of the previous layer



Different neurons have different, but overlapping, receptive fields

后一层的某些神经元的感受域不同，但是系数相同，也就是parameter sharing，这能减少模型的参数数量，降低过拟合概率。

Convolutional Layer



Sparse Connectivity

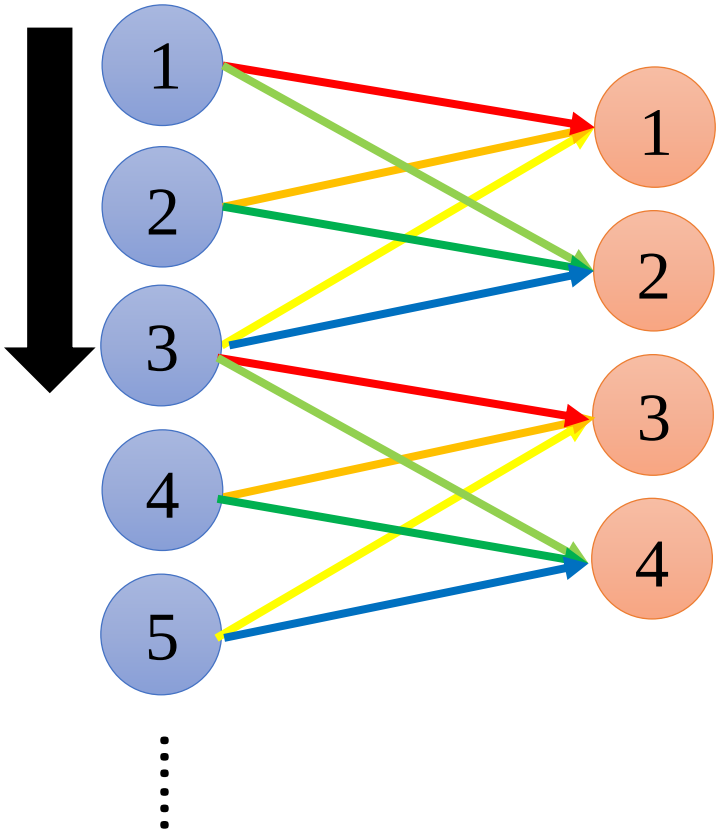
Each neural only connects to part of the output of the previous layer

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

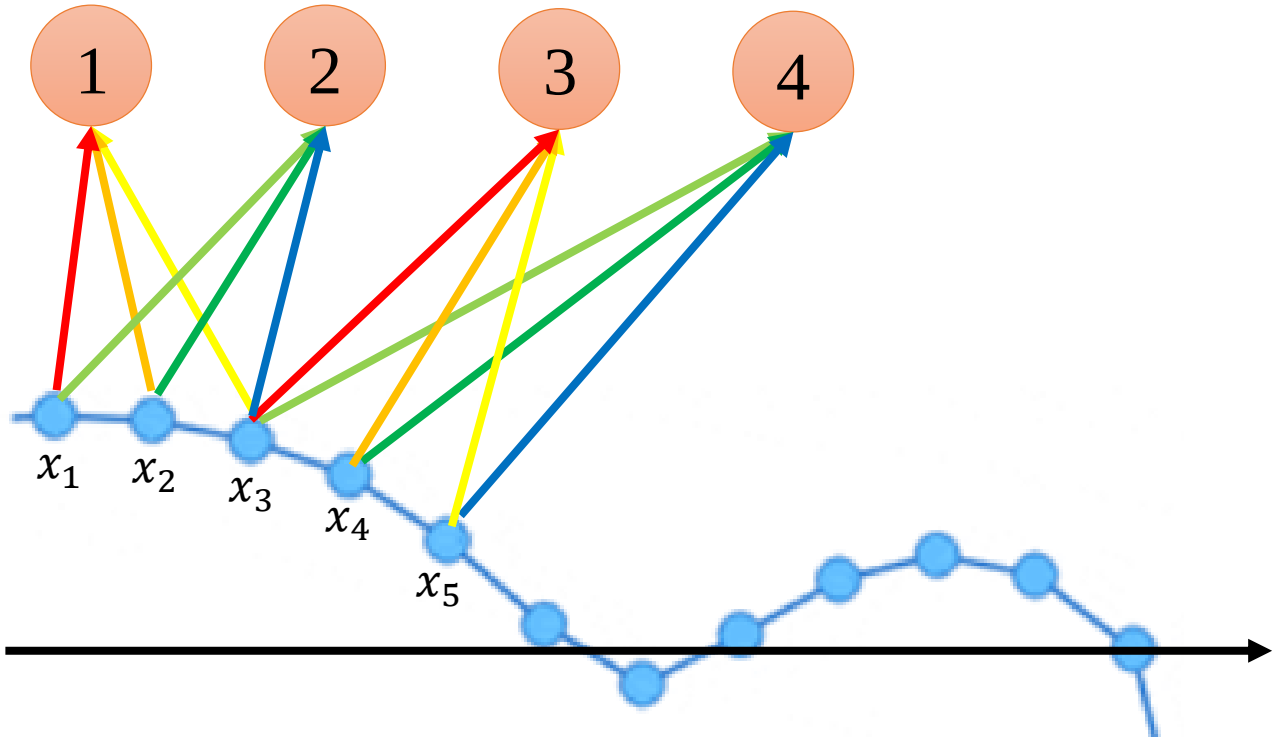
Considering neuron 2 and 4 as "filter 2" (kernel 2)

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

Example –
1D Signal + Single Channel

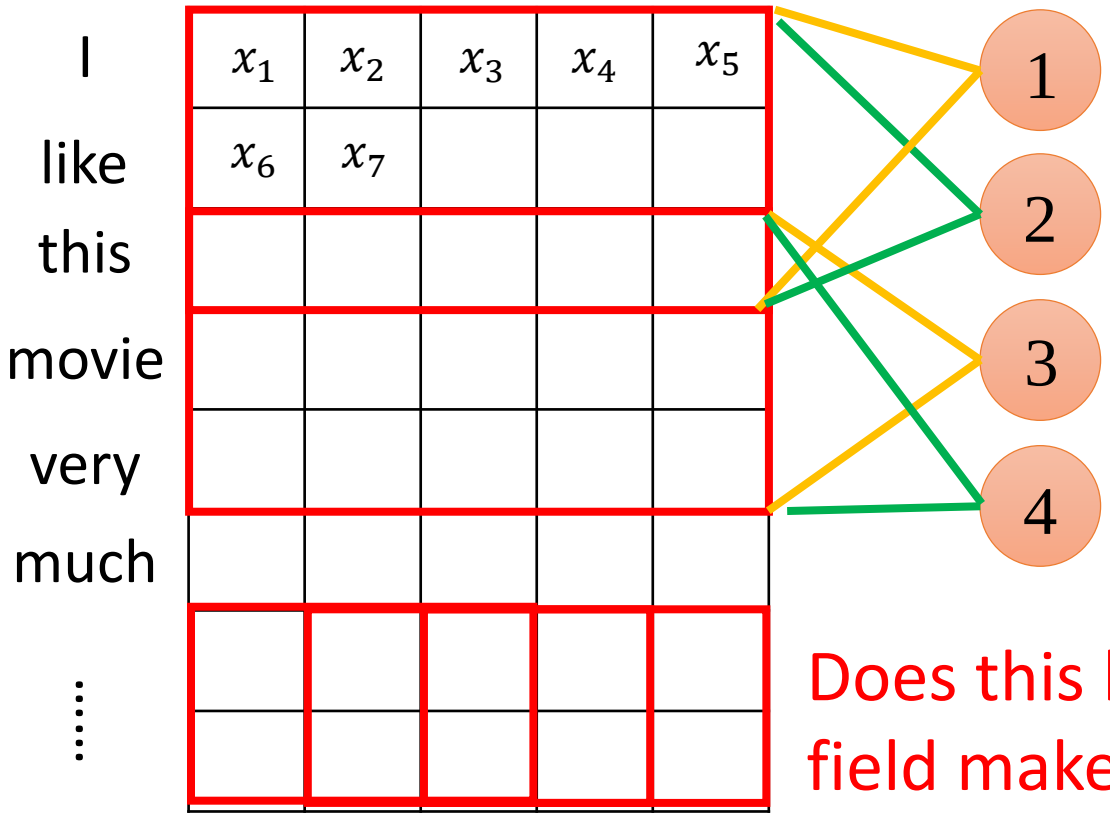
Classification, Predict the future …

Audio Signal,
Stock Value …



Example –
1D Signal + Multiple Channel

A document: each word is a vector



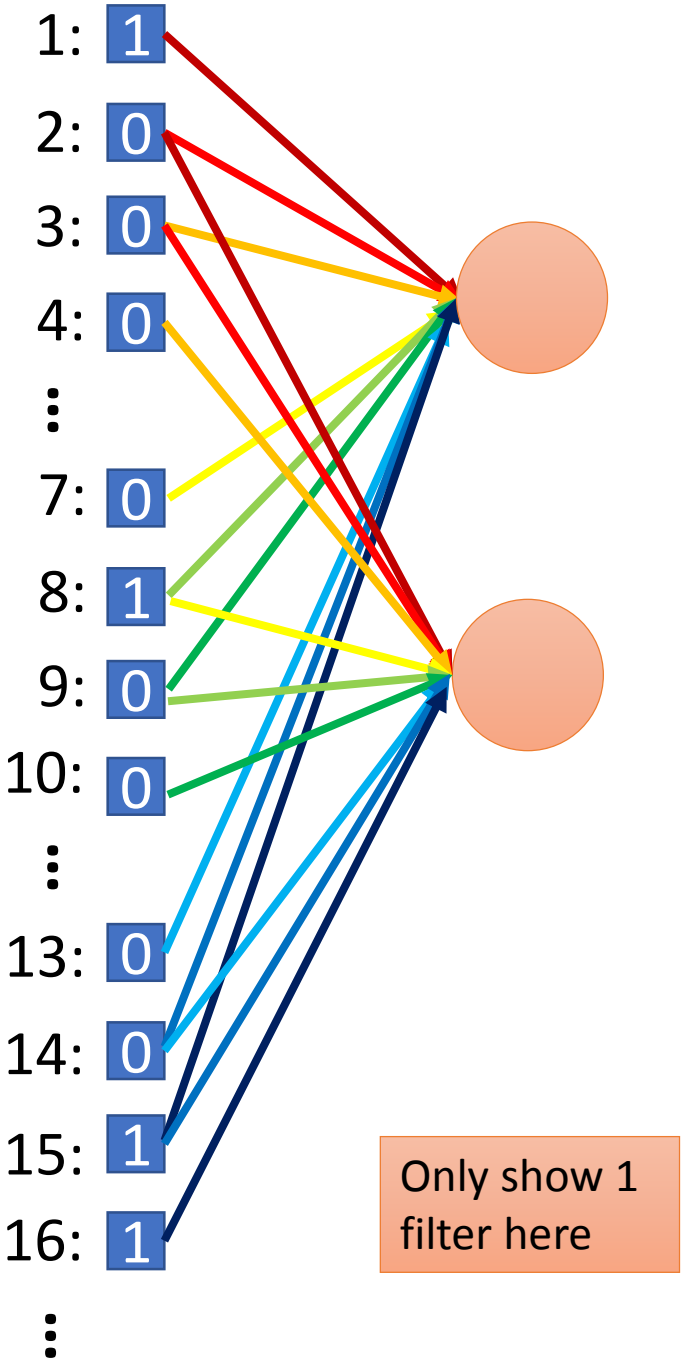
Does this kind of receptive field make sense?

Example –
2D Signal + Single Channel

Size of Receptive field is 3x3, Stride is 1

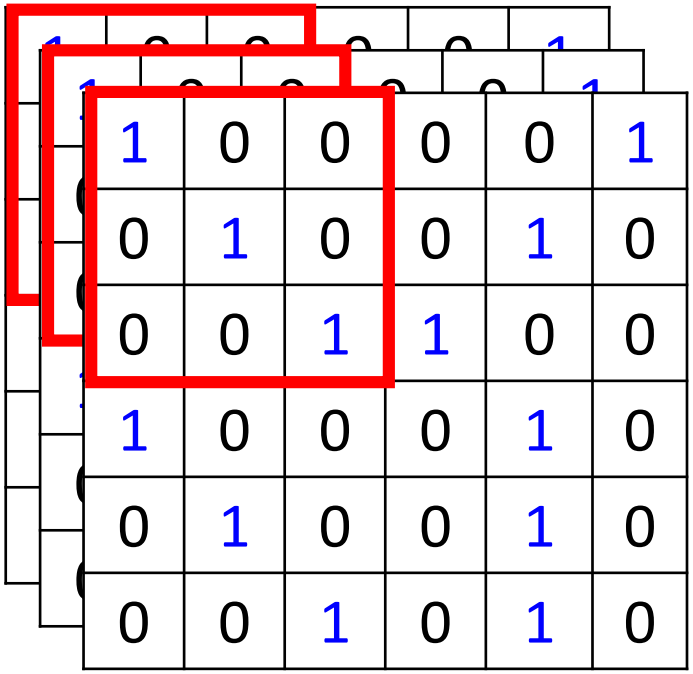
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 black & white picture image

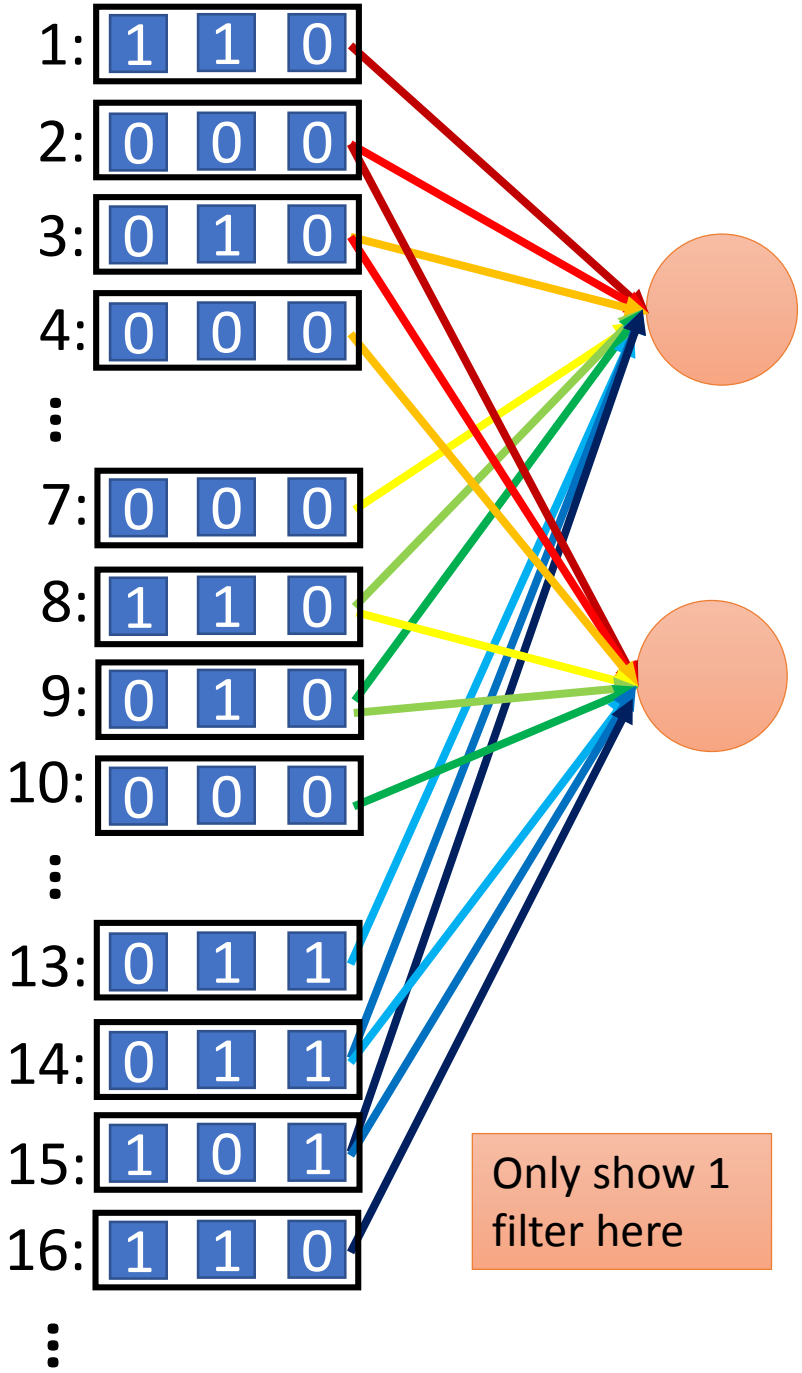


Example – 2D Signal + Multiple Channel

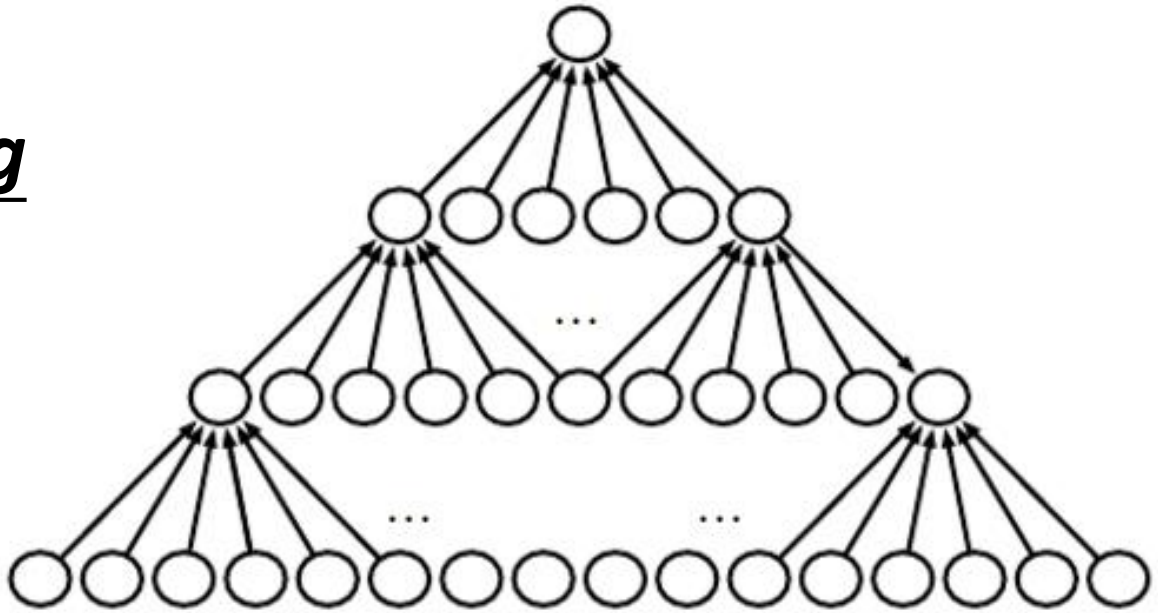
Size of Receptive field is 3x3x3, Stride is 1



6 x 6 colorful image

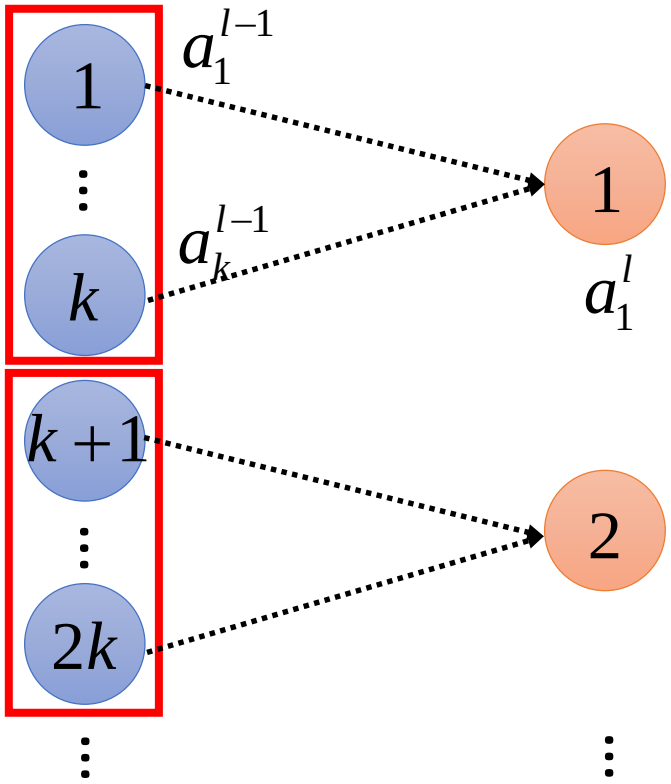


Without Zero Padding



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:

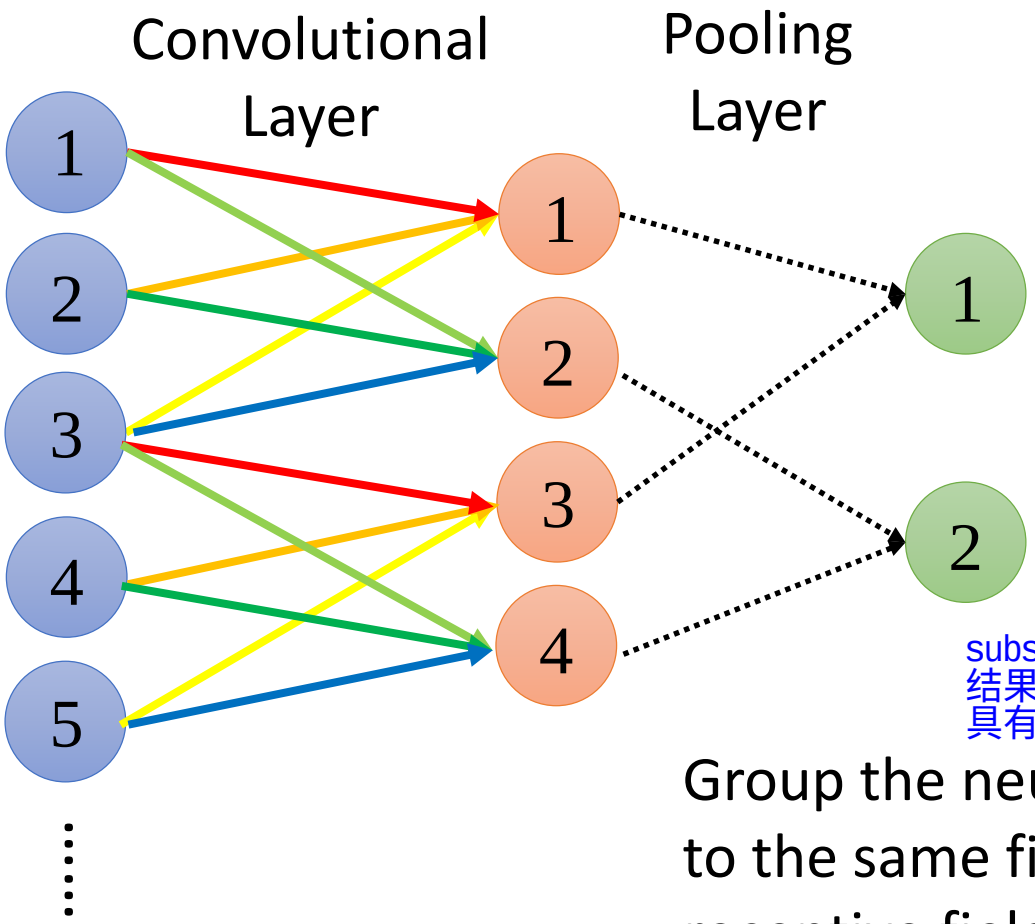
$$a_1^l = \max\left(a_1^{l-1}, a_2^{l-1}, \cdots, a_k^{l-1}\right)$$

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?



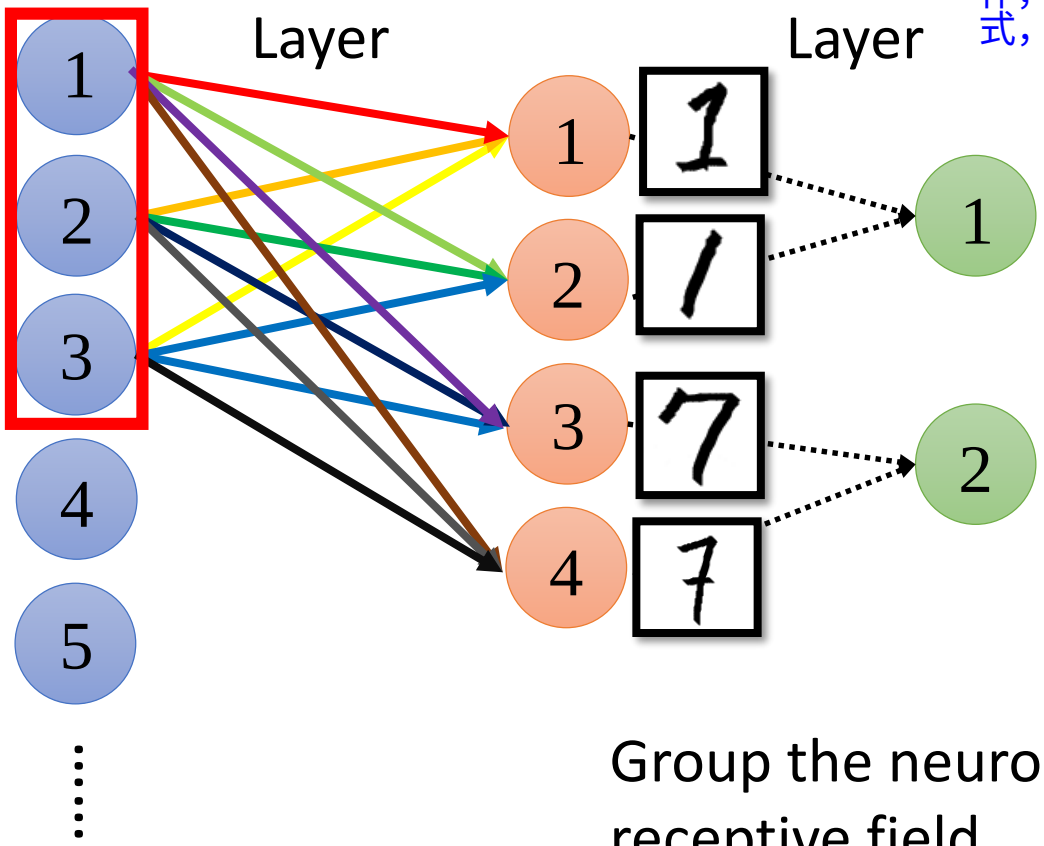
Subsampling

subsampling下采样：将相同filter计算出的结果组合在一起。可以理解为检测全图中是否具有某种特征。

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

Pooling Layer

Which outputs should be grouped together?



对感受域相同的filter的输出进行池化操作，可以检测出同一个事物的不同表现形式，如数字1的不同写法。

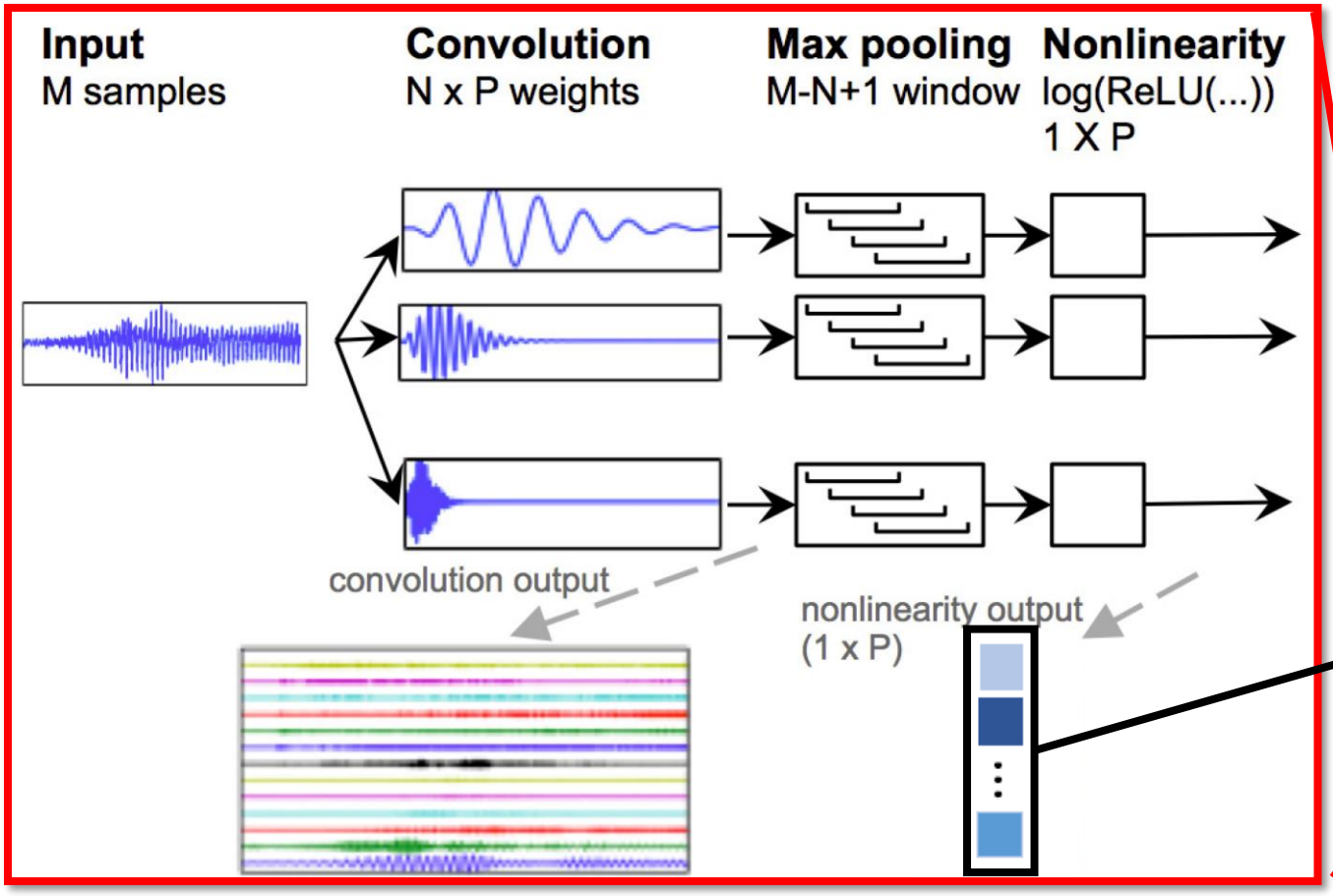
Maxout Network

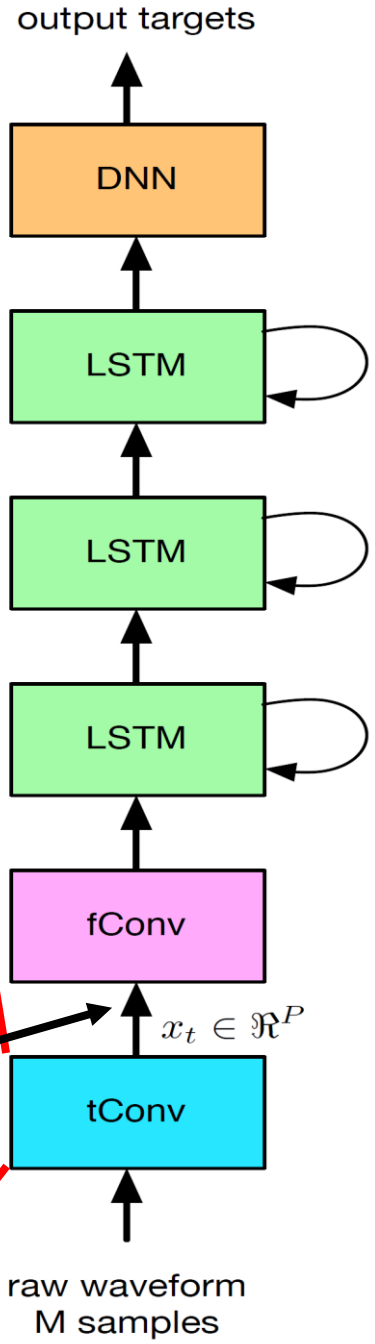
How can you know whether the neurons detect the same pattern?

Group the neurons with the same receptive field

Combination of Different Basic Layers

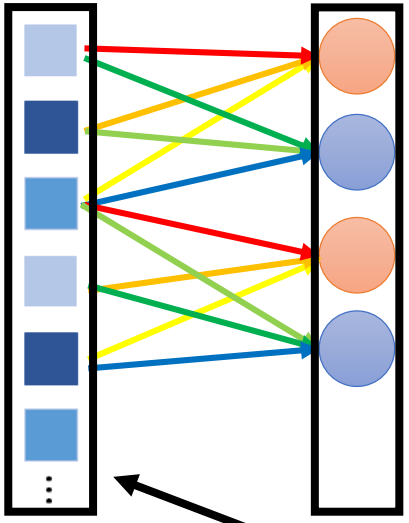
Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, Oriol Vinyals, “Learning the Speech Front-end With Raw Waveform CLDNNs,” In *INTERPSEECH 2015*

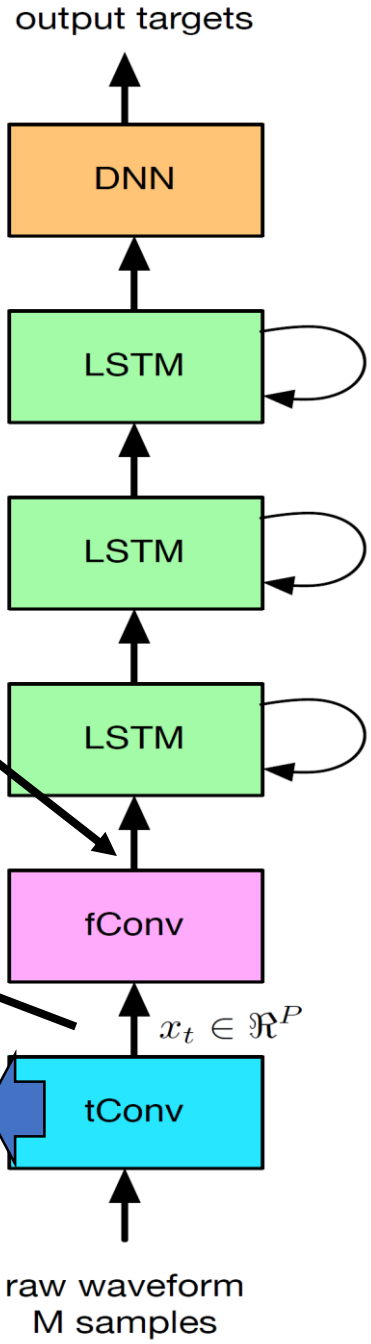


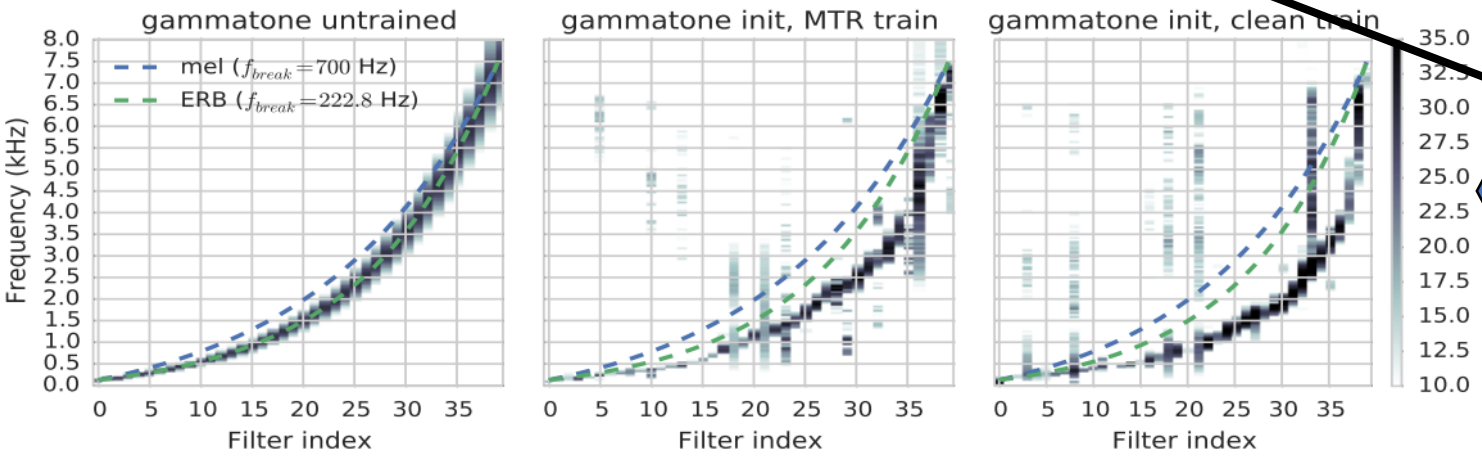


Combination of Different Basic Layers

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

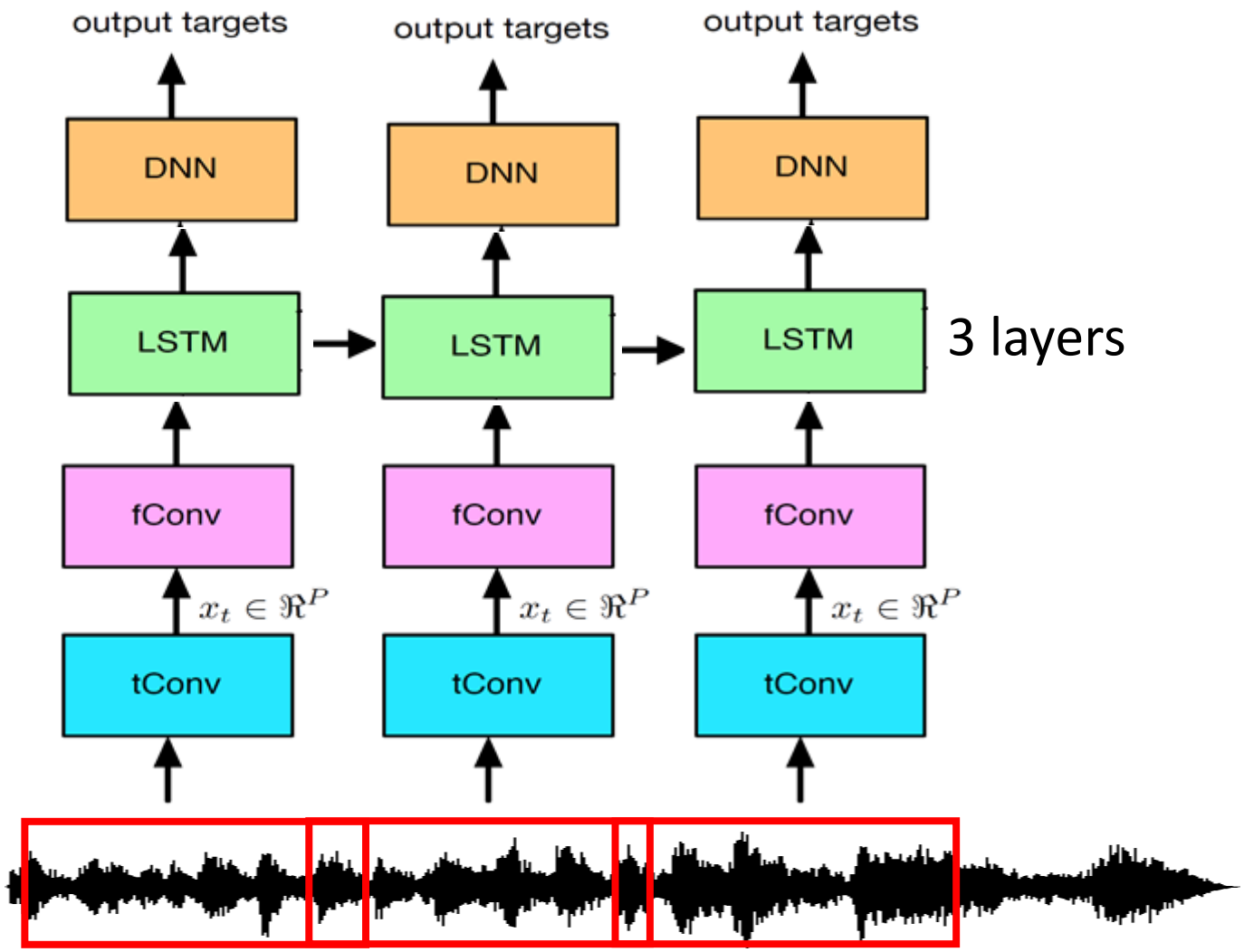


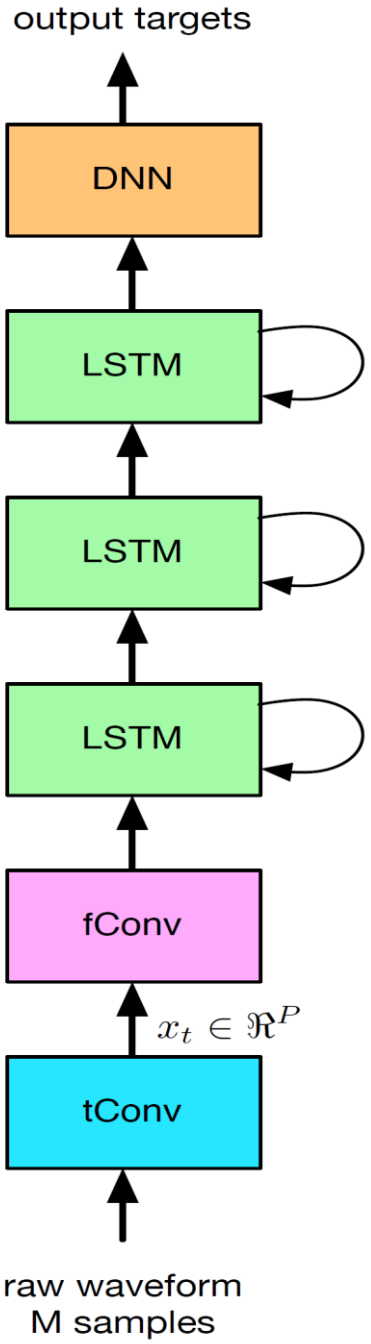




Combination of Different Basic Layers

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