Pretext tasks

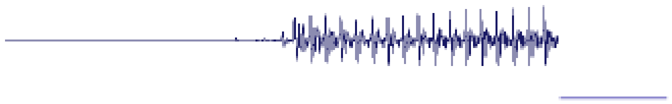
4. CPC

|| AR / LPC / VAR / RNN / APC / CPC ||

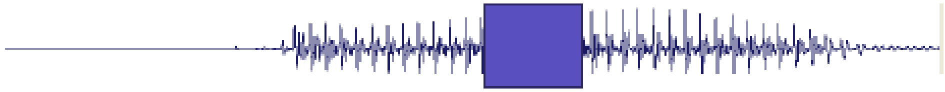
1 2	SELF-PREDICTION	INNATE RELATIONSHIP (Context-based)	1. ROTATION 2. RELATIVE POSITION	IMAGE
3	CONTRASTIVE LEARNING	INTER-SAMPLE CLASSIFICATION	1. Instance Discrimination 2. SimCLR [Contrastive Loss] 3. Theory – Guarantees / Bounds	IMAGE
4	CONTRASTIVE LEARNING	INTER-SAMPLE CLASSIFICATION	Contrastive Predictive Coding (CPC), [NCE, InfoNCE Loss]	AUDIO/ SPEECH
5	SELF-PREDICTION	GENERATIVE (VAE)	1. AE – Variational Bayes 2. VQ-VAE + AR	IMAGE AUDIO/ SPEECH
6	SELF-PREDICTION	GENERATIVE (AR)	1. AR-LM – GPT 2. Masked-LM – BERT	LANGUAGE
7	SELF-PREDICTION	MASKED-GEN (Masked LM for ASR)	1. Wav2Vec / 2.0 2. HuBERT	AUDIO/ SPEECH

Learning with or without supervision – speech and audio

- Next frame prediction



- Masked prediction



- Future prediction
  - To predict future audio features from the historical ones
    - Contrastive predictive coding (CPC) [Oord et al., 2018]
    - Autoregressive predictive coding (APC) [Chung et al., 2019]
    - wav2vec [Schneider et al., 2019]

- [Oord et al., 2018] Representation learning with contrastive predictive coding, arXiv
- [Chung et al., 2019] An unsupervised autoregressive model for speech representation learning, Interspeech
- [Schneider et al., 2019] wav2vec: Unsupervised pre-training for speech recognition, Interspeech

[Oord et al., 2018] Representation learning with contrastive predictive coding, arXiv

[Chung et al., 2019] An unsupervised autoregressive model for speech representation learning, Interspeech

[Schneider et al., 2019] wav2vec: Unsupervised pre-training for speech recognition, Interspeech

INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

An Unsupervised Autoregressive Model for Speech Representation Learning

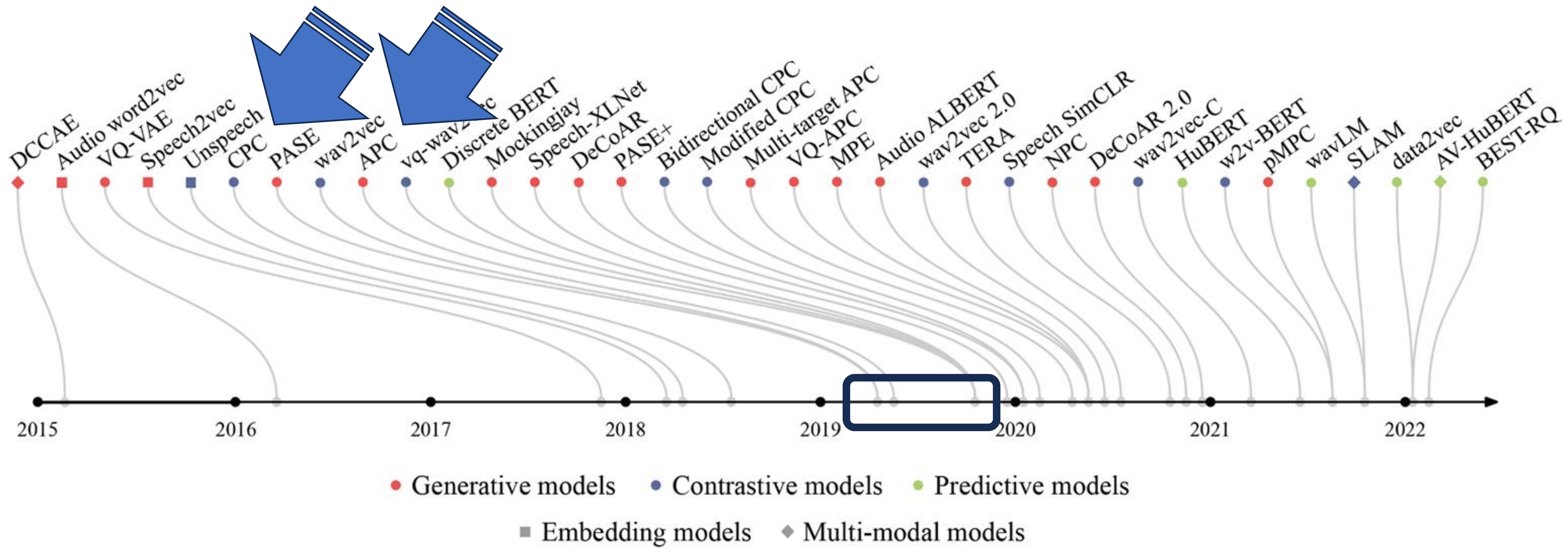
Yu-An Chung, Wei-Ning Hsu, Hao Tang, James Glass

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{andyyuan,wnhsu,haotang,glass}@mit.edu

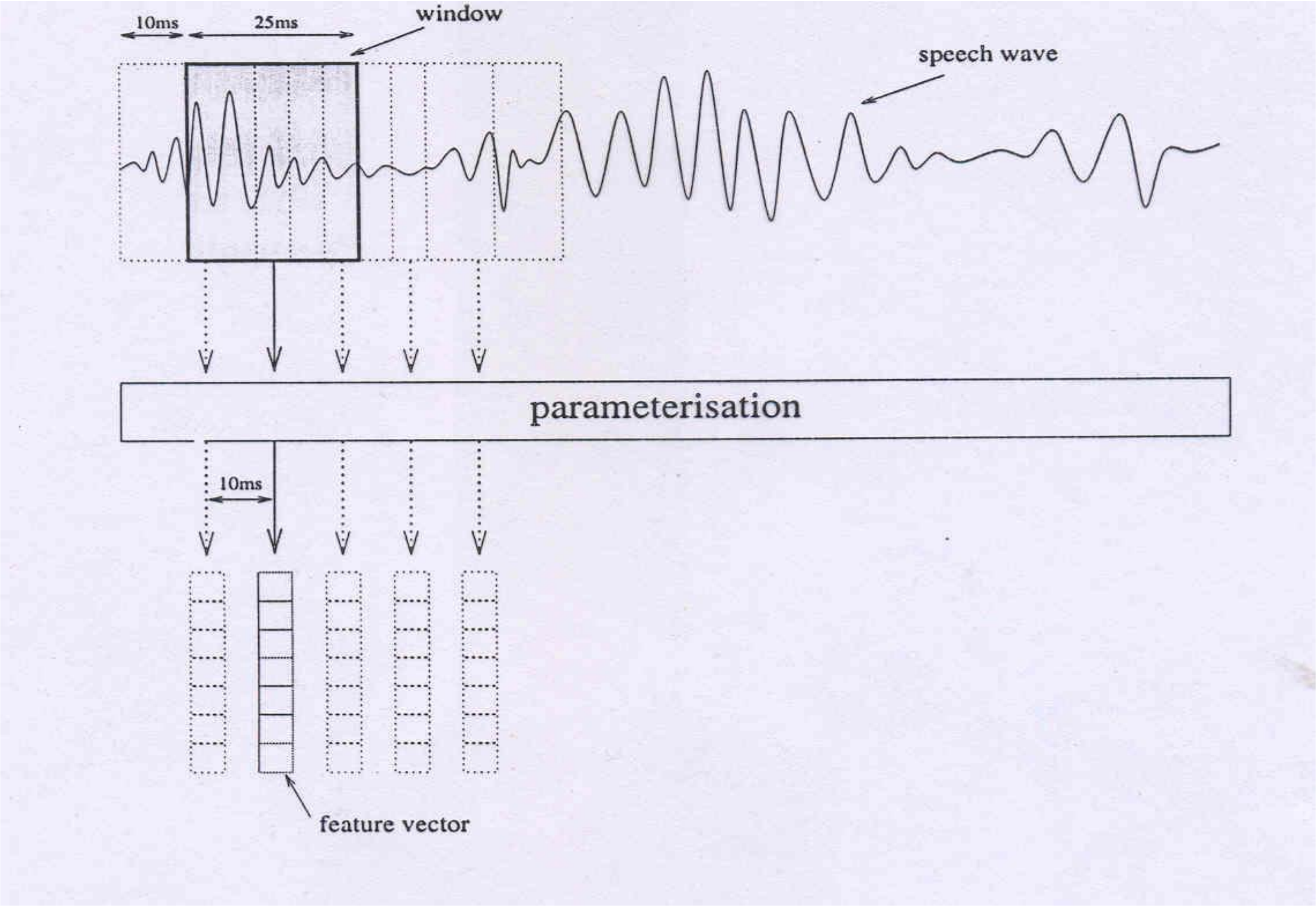
Autoregressive Predictive Coding: A Comprehensive Study

Gene-Ping Yang, Sung-Lin Yeh, Yu-An Chung, James Glass, and Hao Tang

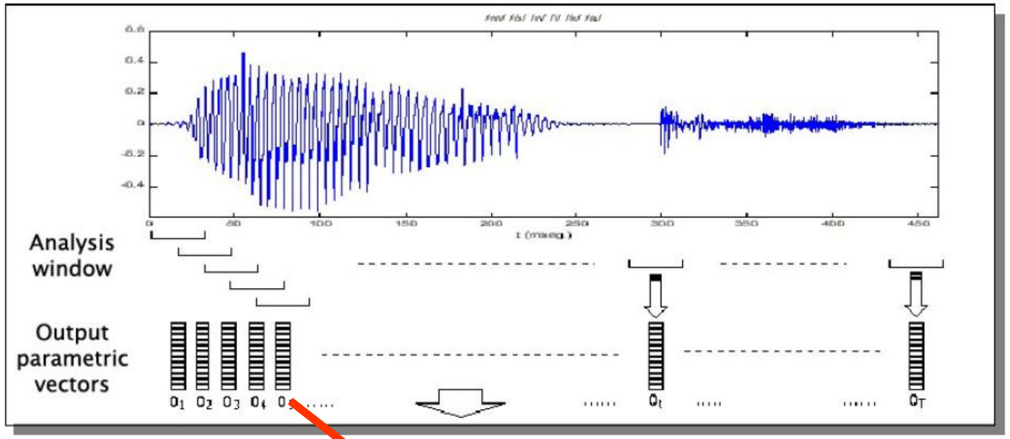
Speech representation learning methods



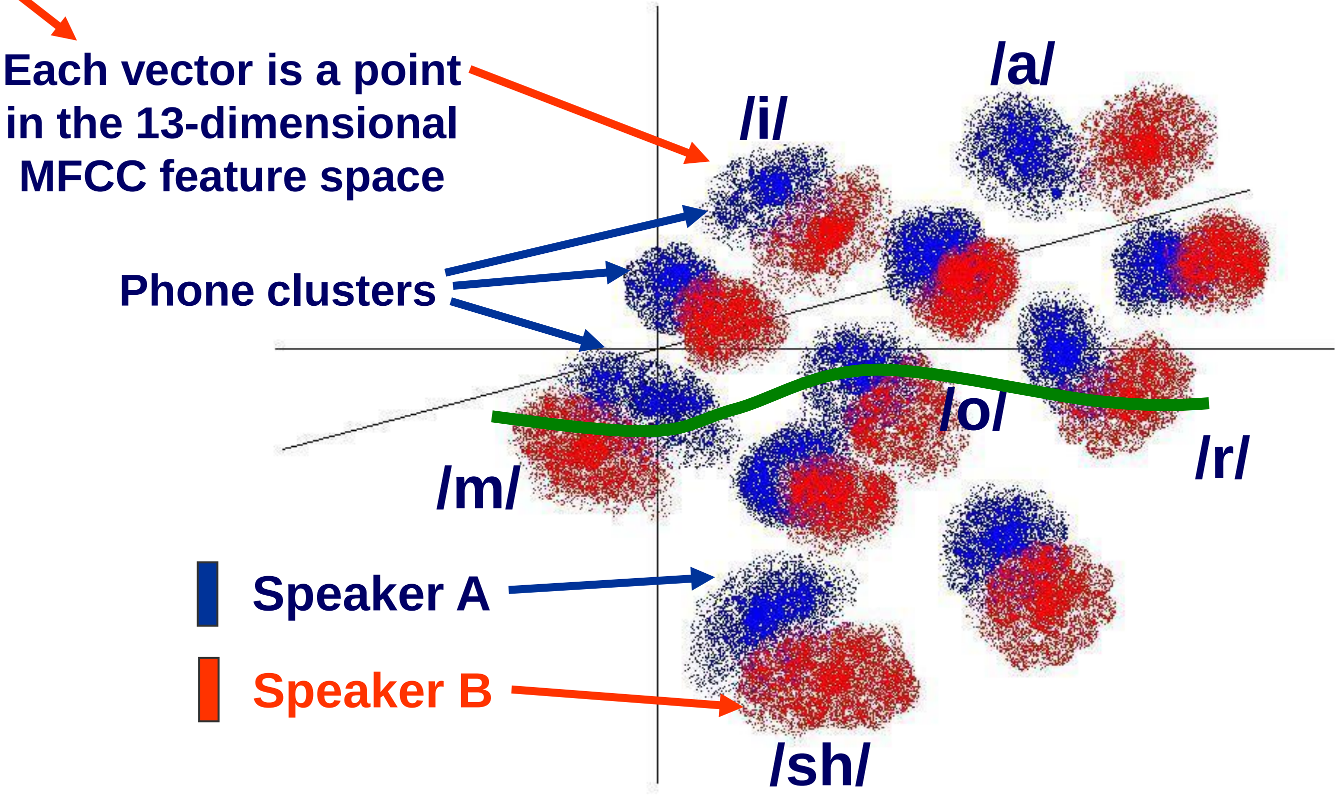
Short-time Analysis and Parameterization



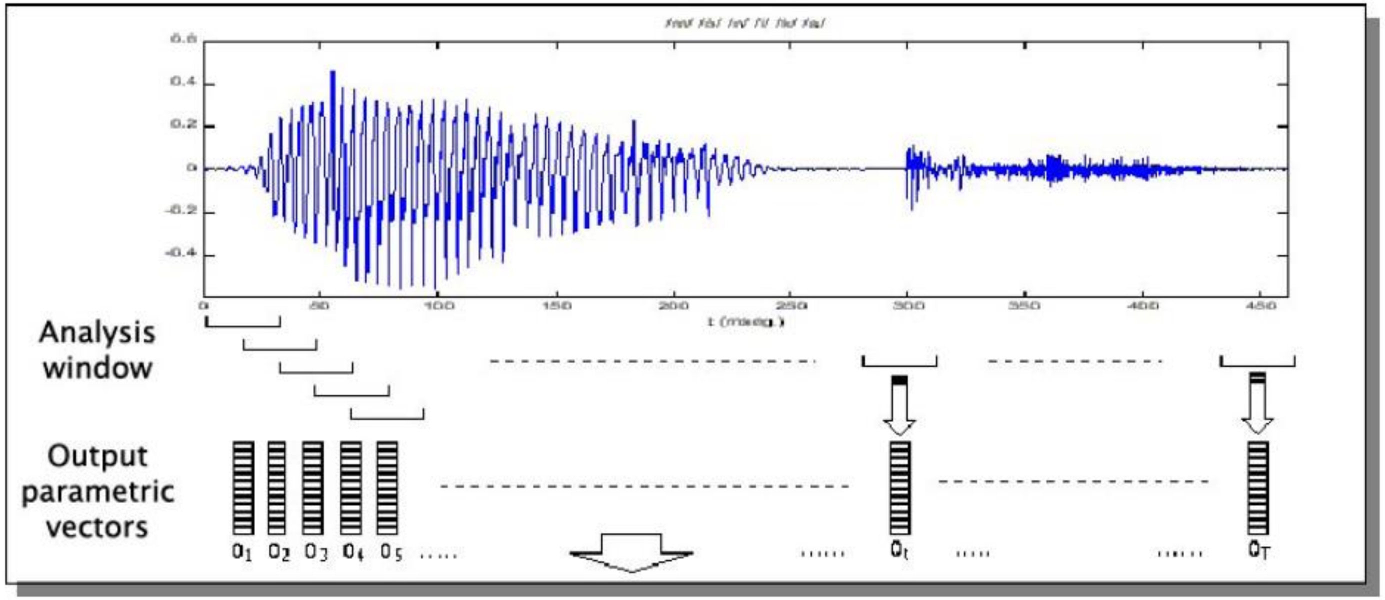
Feature Space



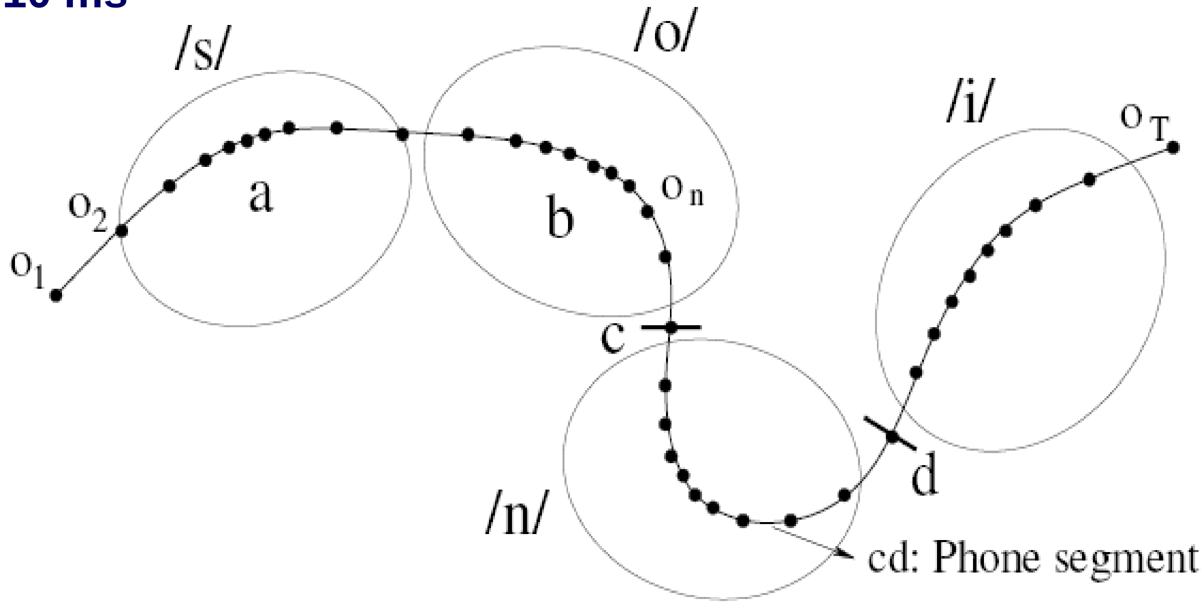
Bag of vectors representation of speaker's acoustic space



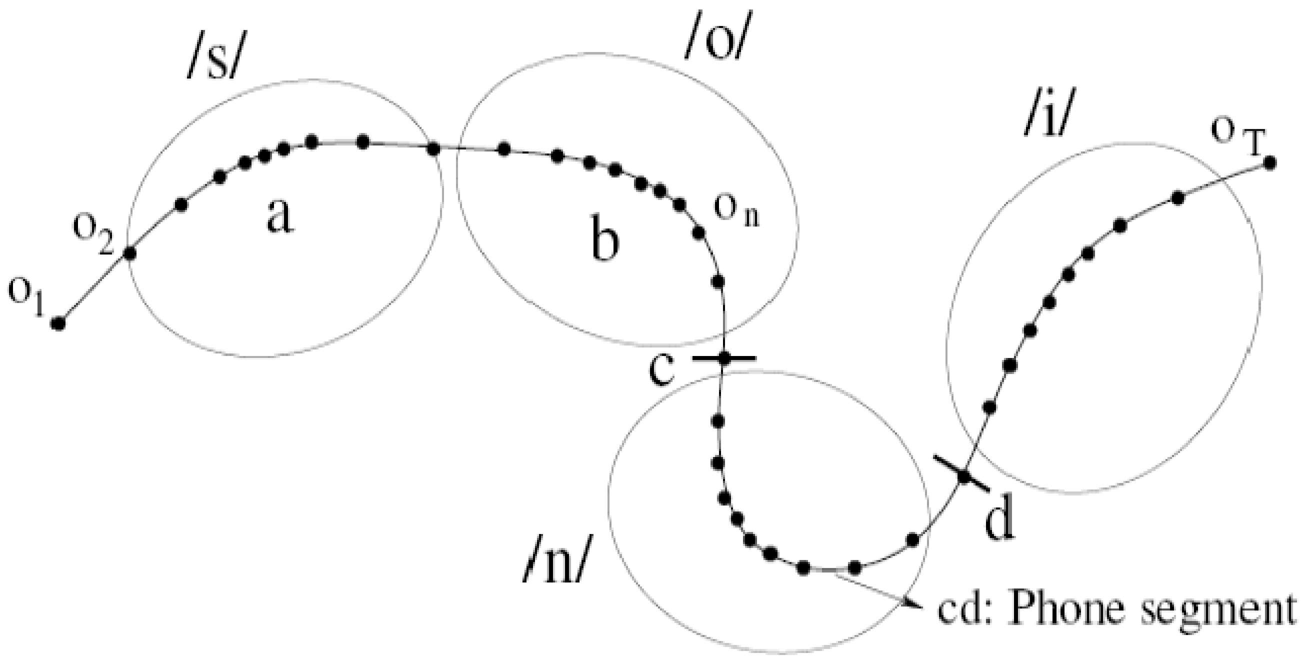
Feature Space



One feature vector every 10 ms



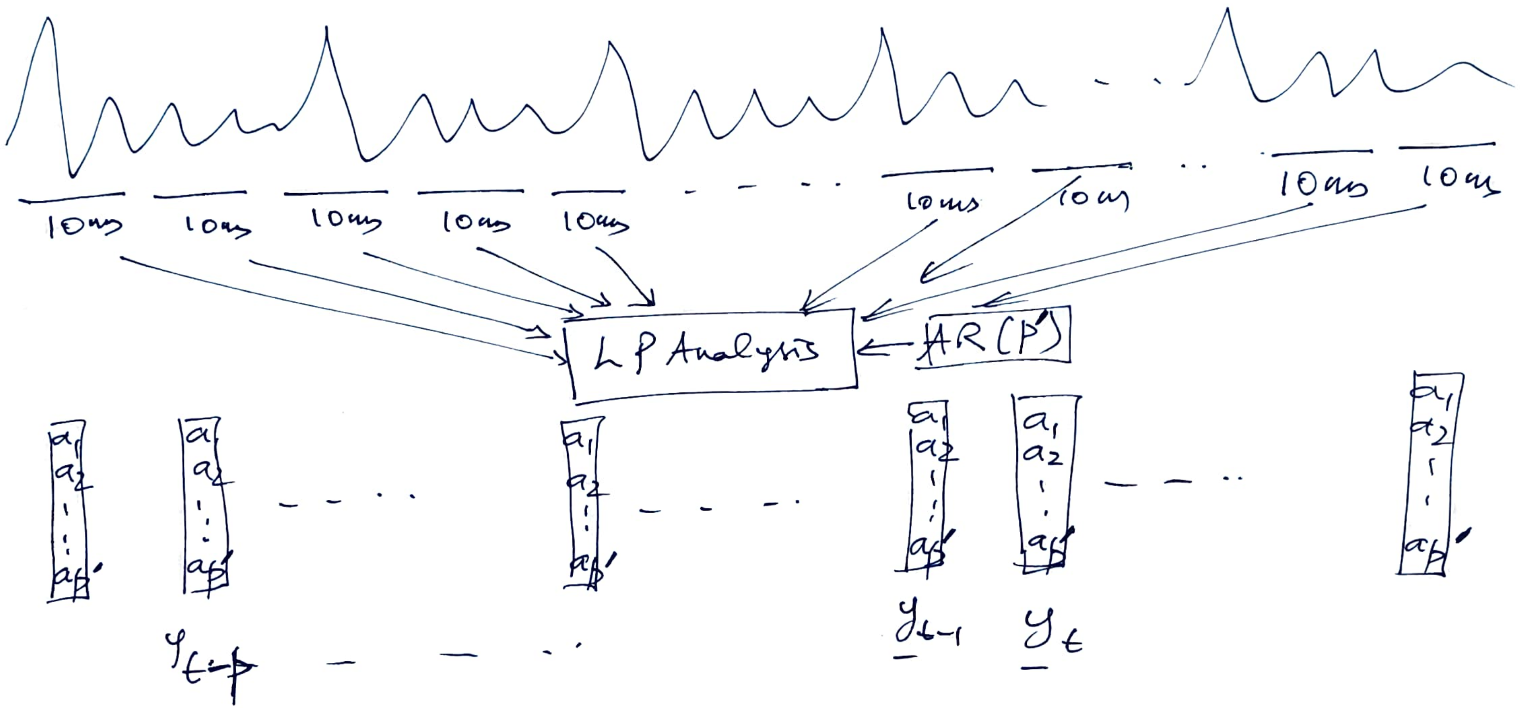
Feature Space



SPEECH RECOGNITION ALGORITHMS

- TAKE THIS FEATURE VECTOR SEQUENCE
- AS INPUT AND DETERMINE "WHAT HAS BEEN SAID"
- e.g. SEQUENCE OF PHONES / SEQUENCE OF WORDS etc.

VAR: Vector Autoregression



$$\underline{y}_t = c + A_1 \underline{y}_{t-1} + A_2 \underline{y}_{t-2} + \dots + A_i \underline{y}_{t-i} + \dots + A_p \underline{y}_{t-p} + \underline{e}_t$$

$p \times 1$

$p \times 1$ intercept constant

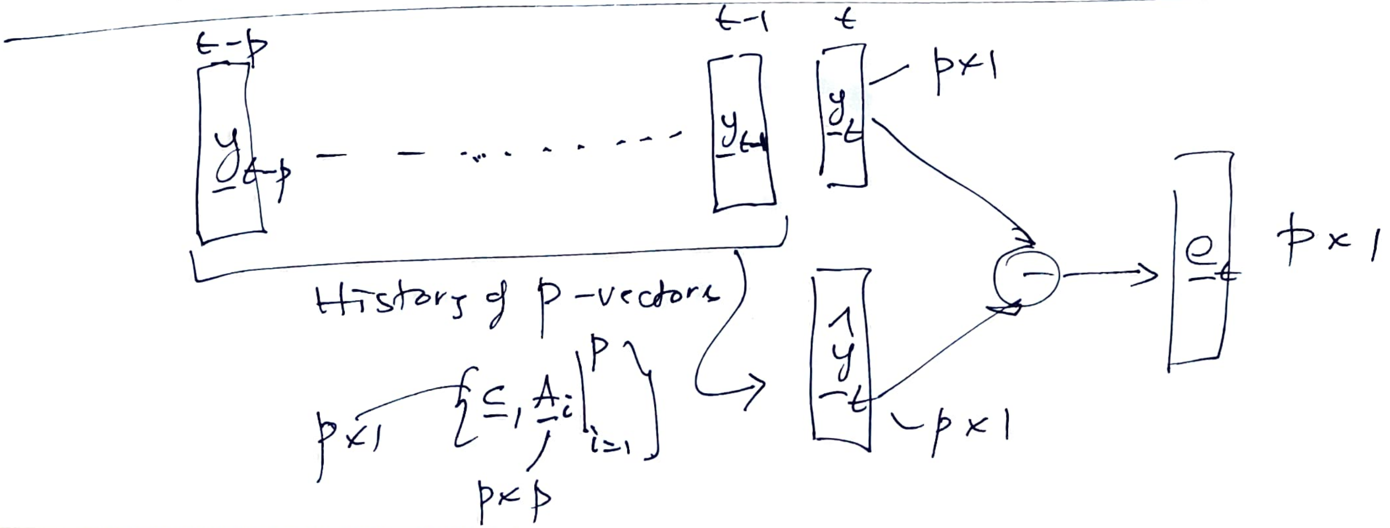
A_i: $p \times p$ time invariant matrix

error vector $p \times 1$

$p = 1 \Rightarrow$ VAR(1), 2-d case

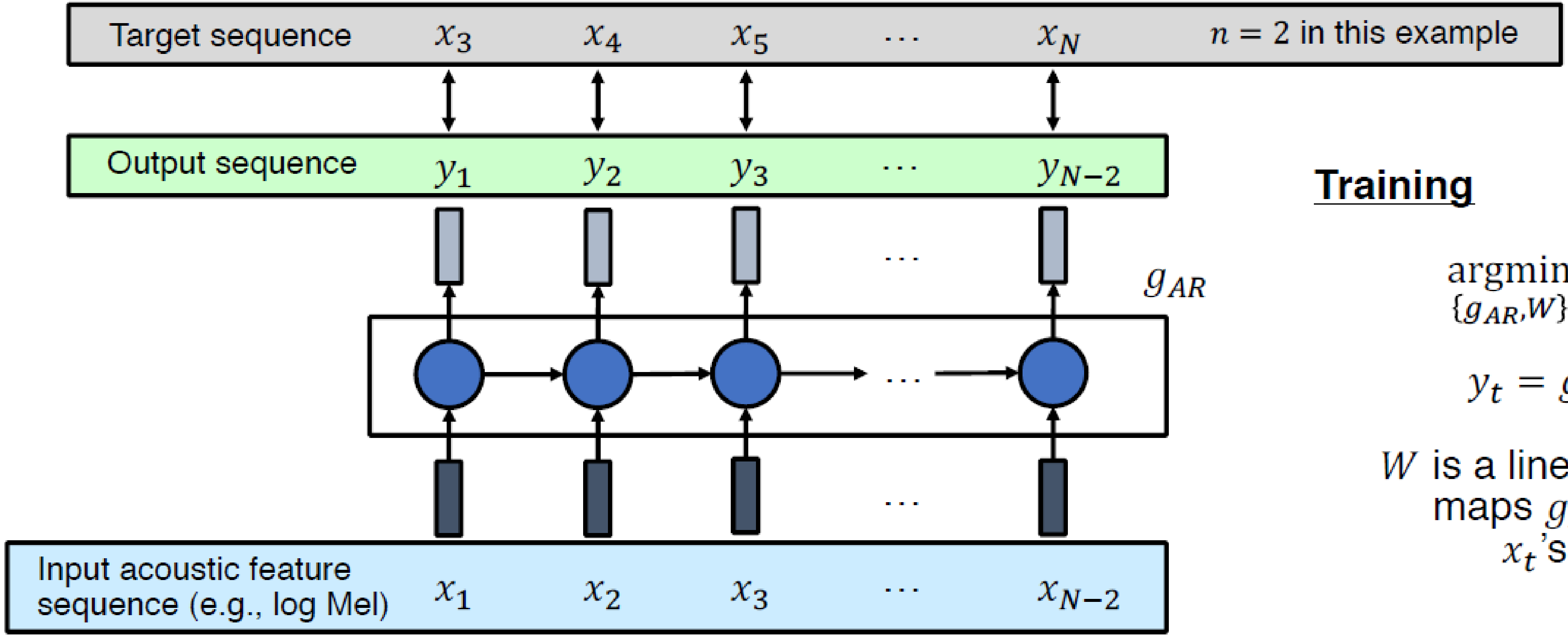
$$\underbrace{\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix}}_{\underline{y}_t} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \overbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}^{A_1} \overbrace{\begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix}}^{\underline{y}_{t-1}} + \begin{bmatrix} e_{1,t} \\ e_{2,t} \end{bmatrix}$$

$$\Rightarrow \left.\begin{aligned} y_{1,t} &= c_1 + a_{11} y_{1,t-1} + a_{1,2} y_{2,t-1} + e_{1,t} \\ y_{2,t} &= c_2 + a_{21} y_{1,t-1} + a_{2,2} y_{2,t-1} + e_{2,t} \end{aligned}\right]$$



Autoregressive Predictive Coding (APC)

- Given a previous context $(\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_t})$, APC tries to predict a future audio feature $\boldsymbol{x_{t+n}}$ that is $\boldsymbol{n}$ steps ahead of $\boldsymbol{x_t}$
  - Uses an autoregressive model g_{AR} to summarize history and produce output
  - $n \geq 1$ encourages g_{AR} to infer more global underlying structures of the data rather than simply exploiting local smoothness of speech signals



Training

$$\underset{\{g_{AR}, W\}}{\operatorname{argmin}} \ \sum_{t=1}^{N-n} |x_{t+n} - y_t|,$$

$$y_t = g_{AR}(x_1, \dots, x_t) \cdot W$$

W is a linear transformation that maps g_{AR}'s output back to x_t's dimensionality

Types of autoregressive model g_{AR}

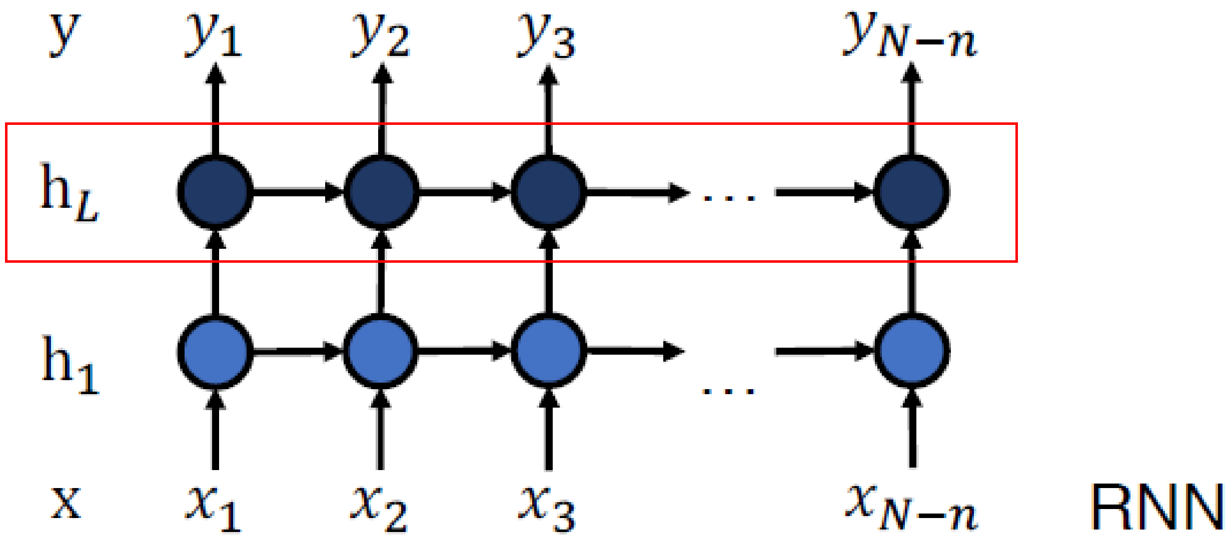
- g_{AR}
  - Input: $\mathbf{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N})$
  - Output: $\mathbf{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_N})$

- L-layer Unidirectional RNN:

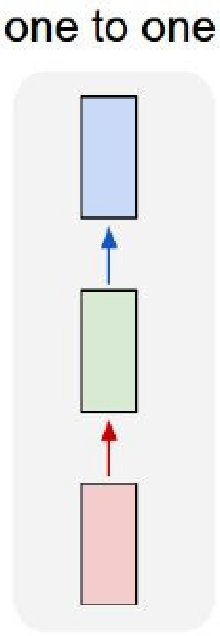
$$\mathrm{h}_0 = \mathrm{x}$$

$$\mathrm{h}_l = \mathrm{RNN}^{(l)}(h_{l-1}), \forall l \in [1, L]$$

$$\mathrm{y} = \mathrm{h}_L \cdot W$$

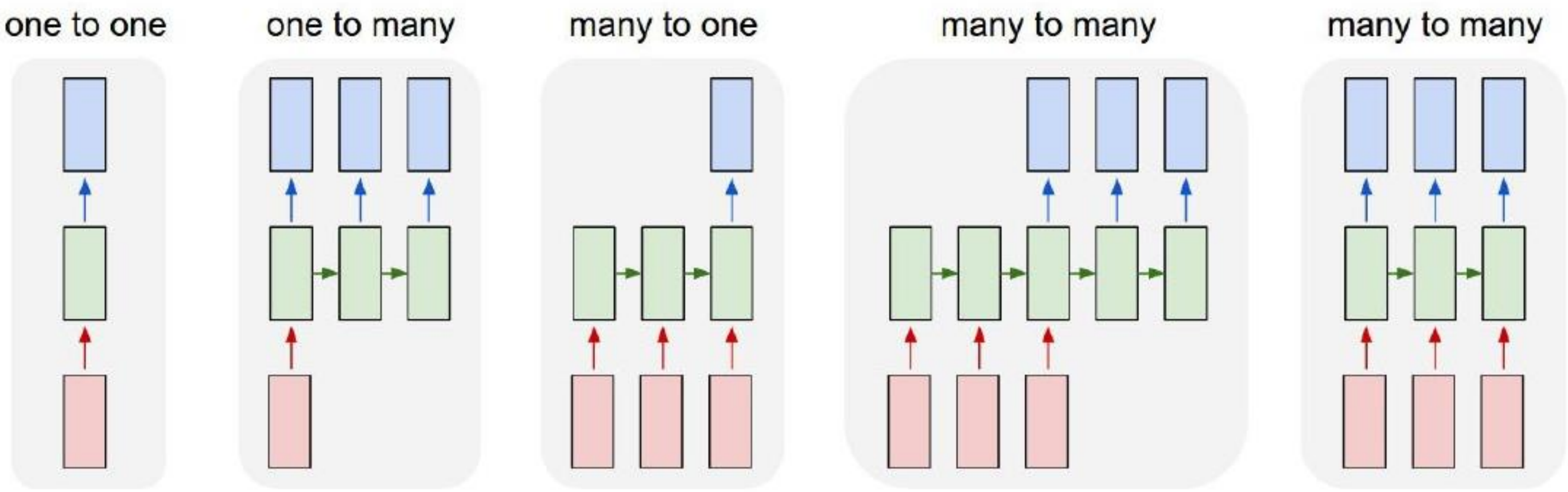


- Feature extraction: $\mathbf{h}_L$



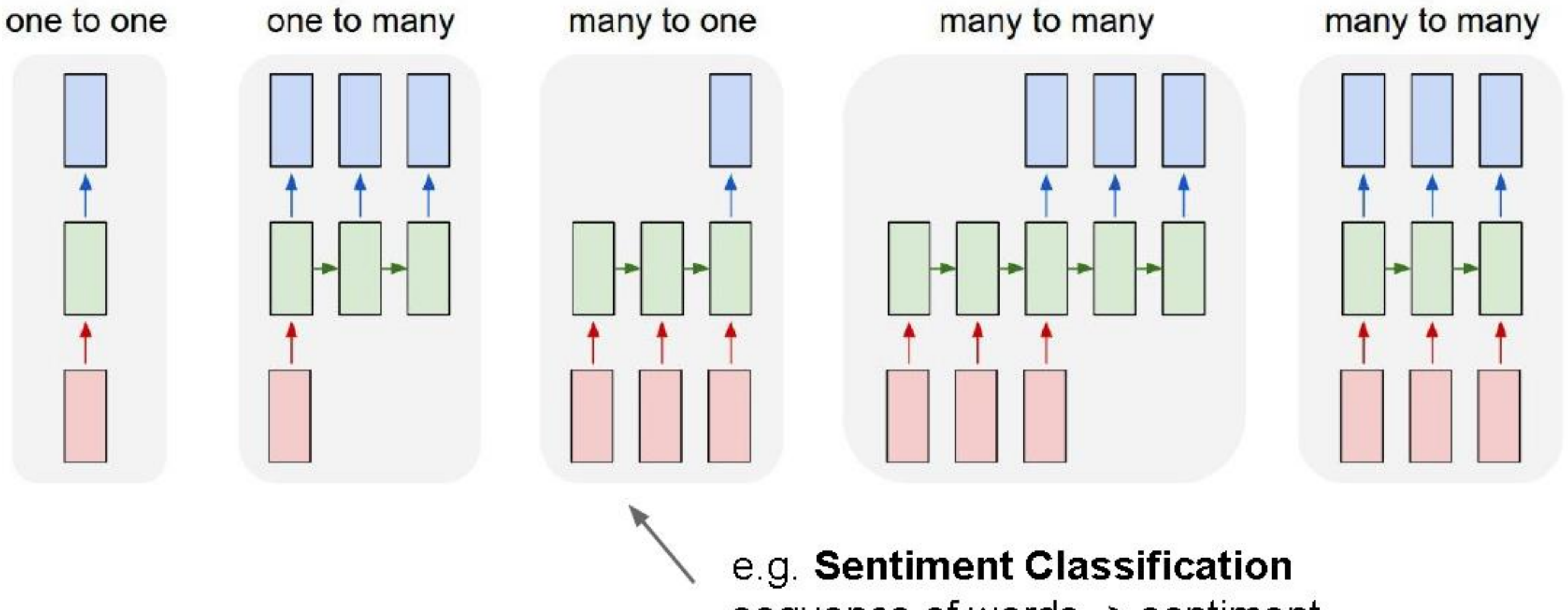
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



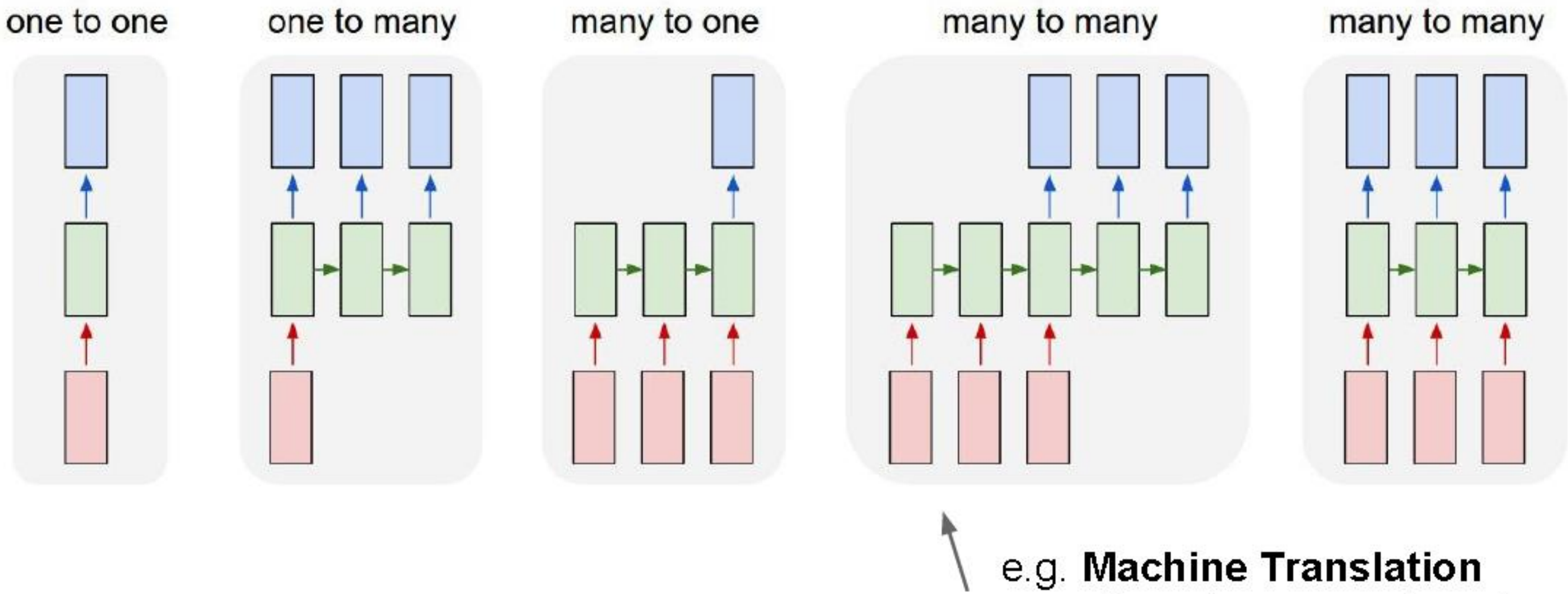
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



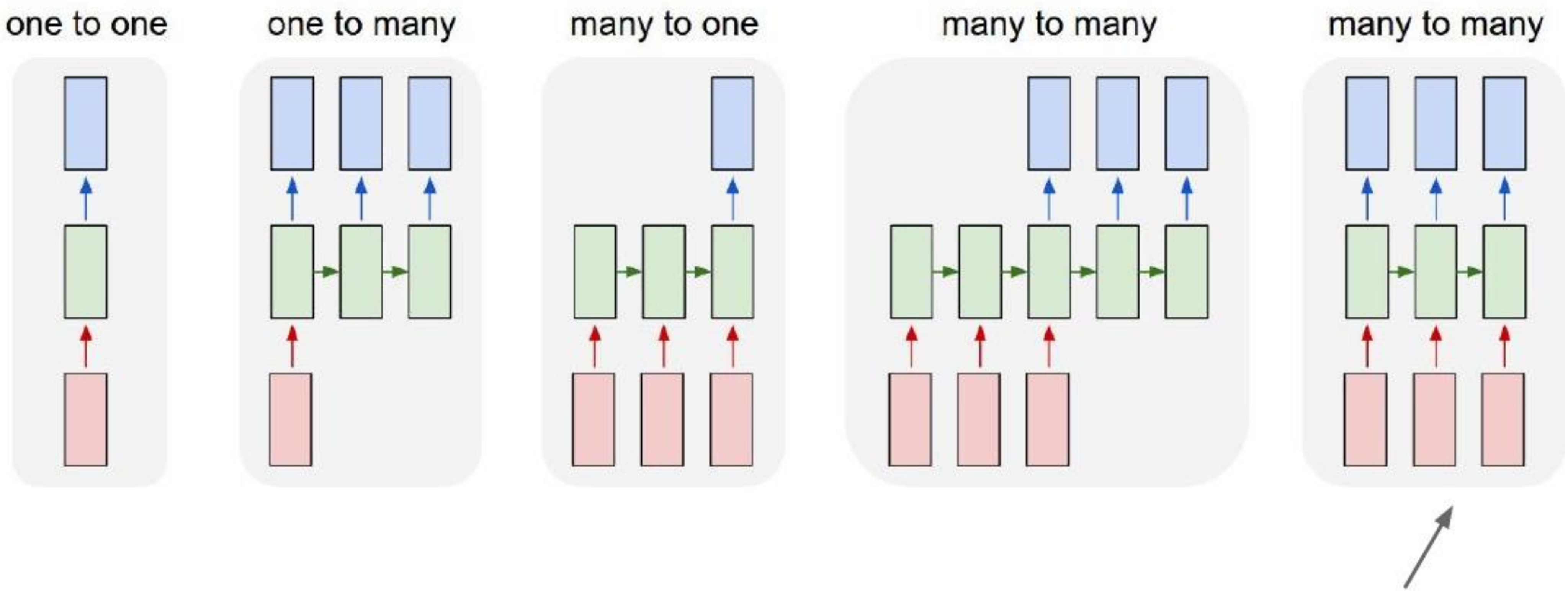
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences



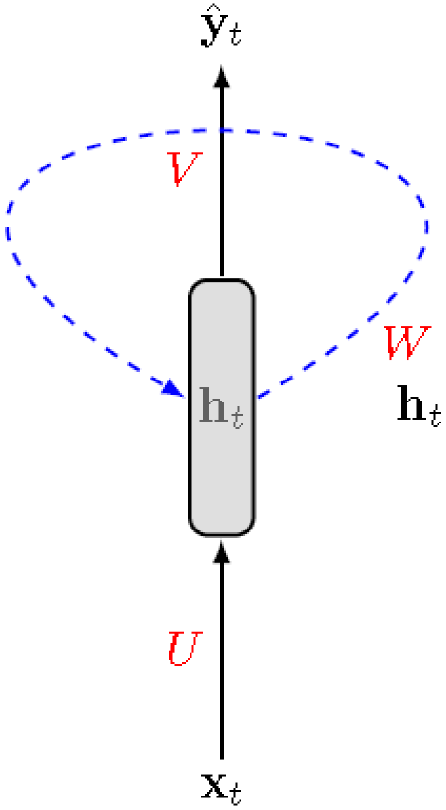
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences



e.g. **Video classification on frame level**

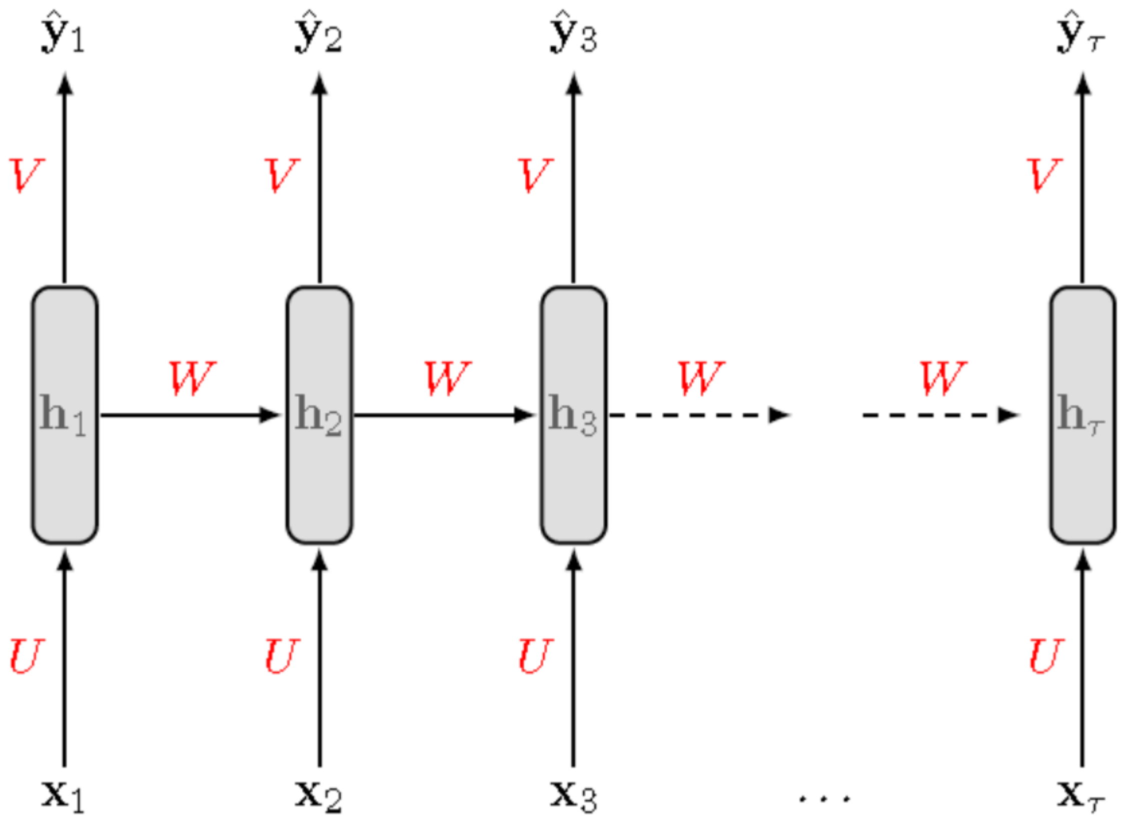
Unrolling the Recurrence



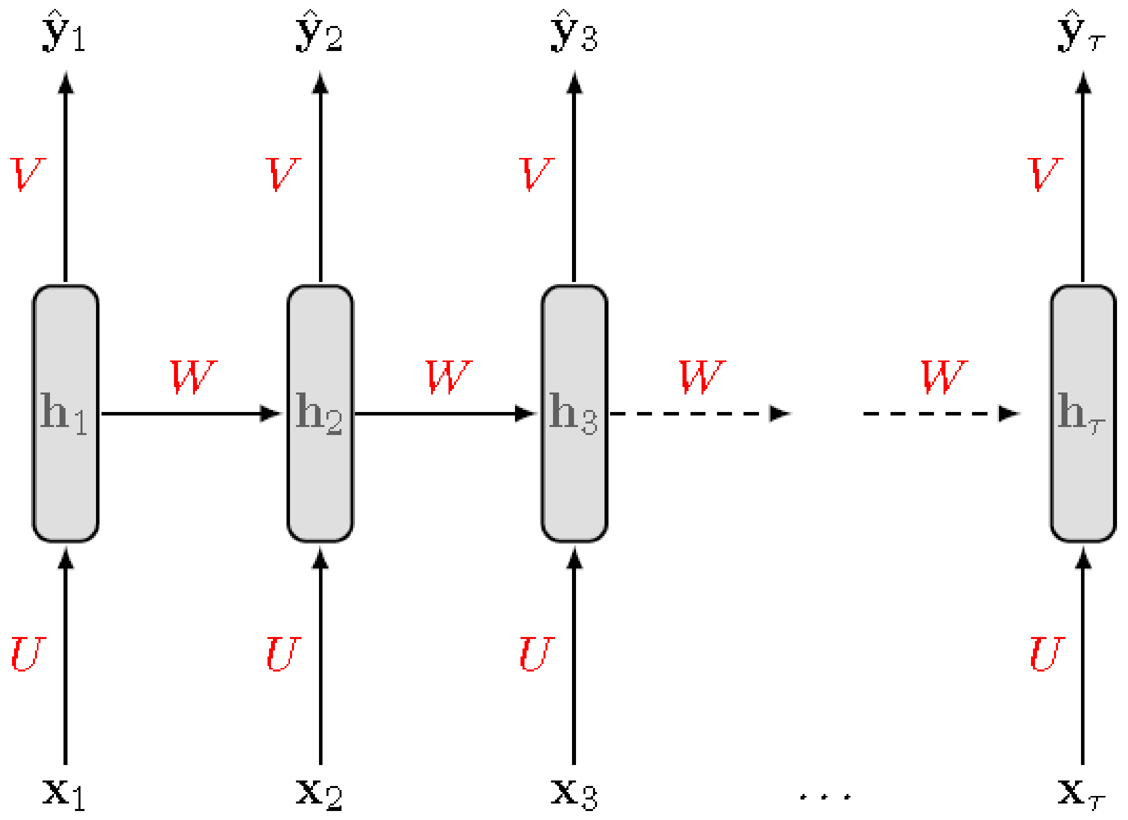
$$\hat{\mathbf{y}}_t = \phi(V\mathbf{h}_t)$$

$$\mathbf{h}_t = \psi(U\mathbf{x}_t + W\mathbf{h}_{t-1})$$

ψ can be tanh and ϕ can be softmax



Unrolling the Recurrence

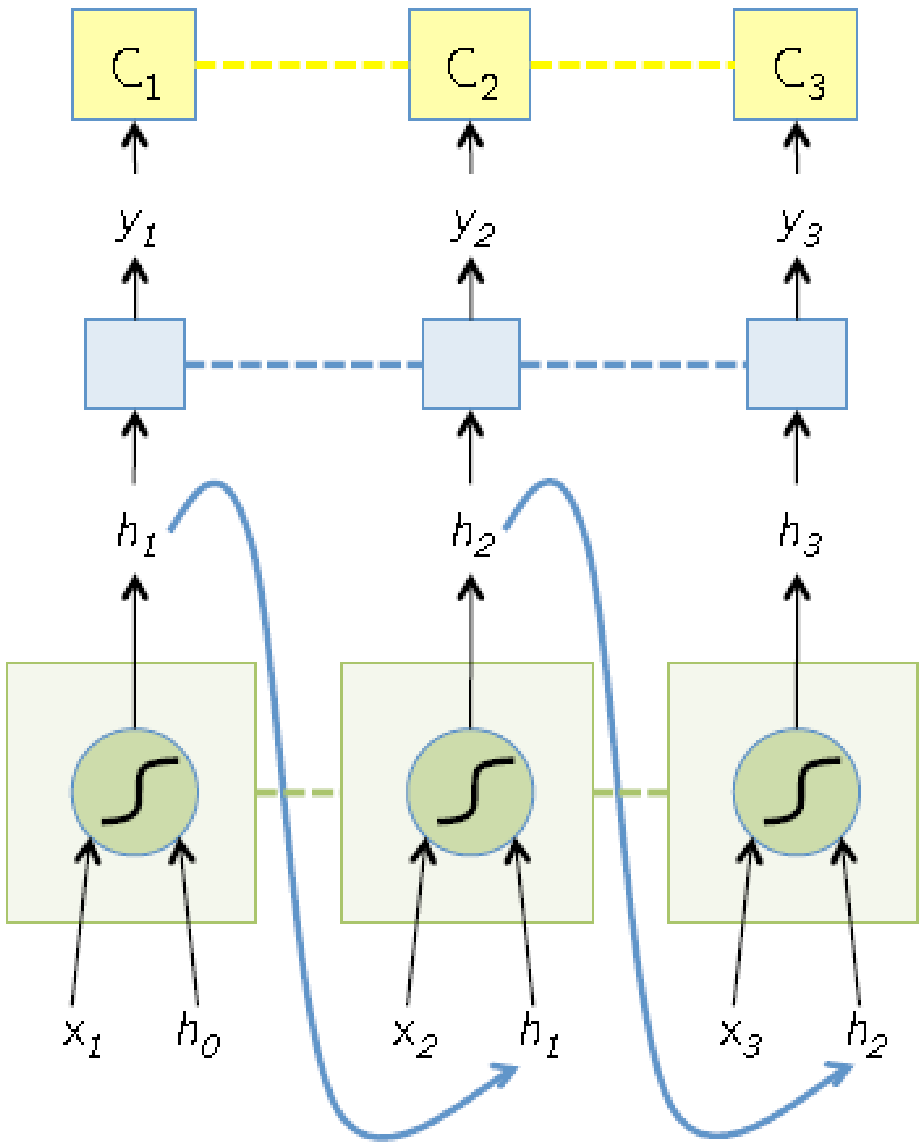


$$\mathbf{a}_t = b + W\mathbf{h}_{t-1} + U\mathbf{x}_t$$

$$\mathbf{h}_t = \tanh \mathbf{a}_t$$

$$\mathbf{o}_t = c + V\mathbf{h}_t$$

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{o}_t)$$



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

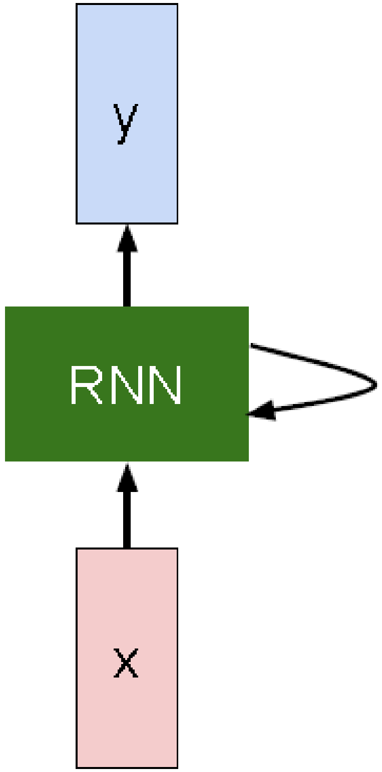
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

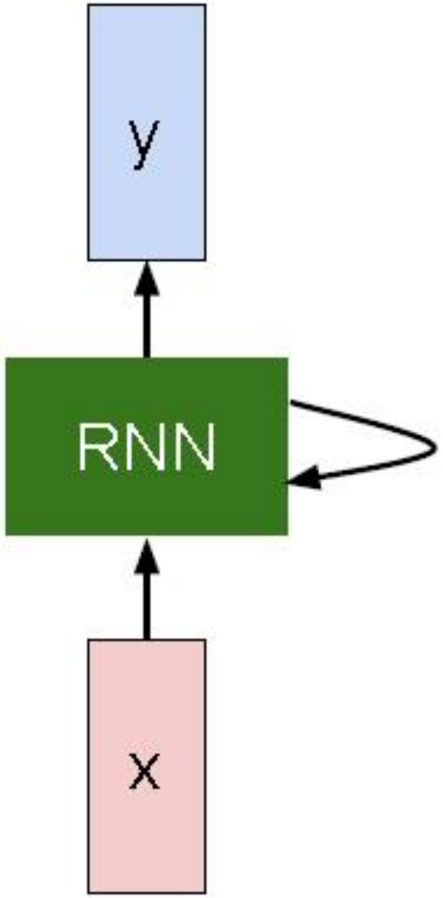
old state

input vector at some time step

some function with parameters W



Notice: the same function and the same set of parameters are used at every time step.

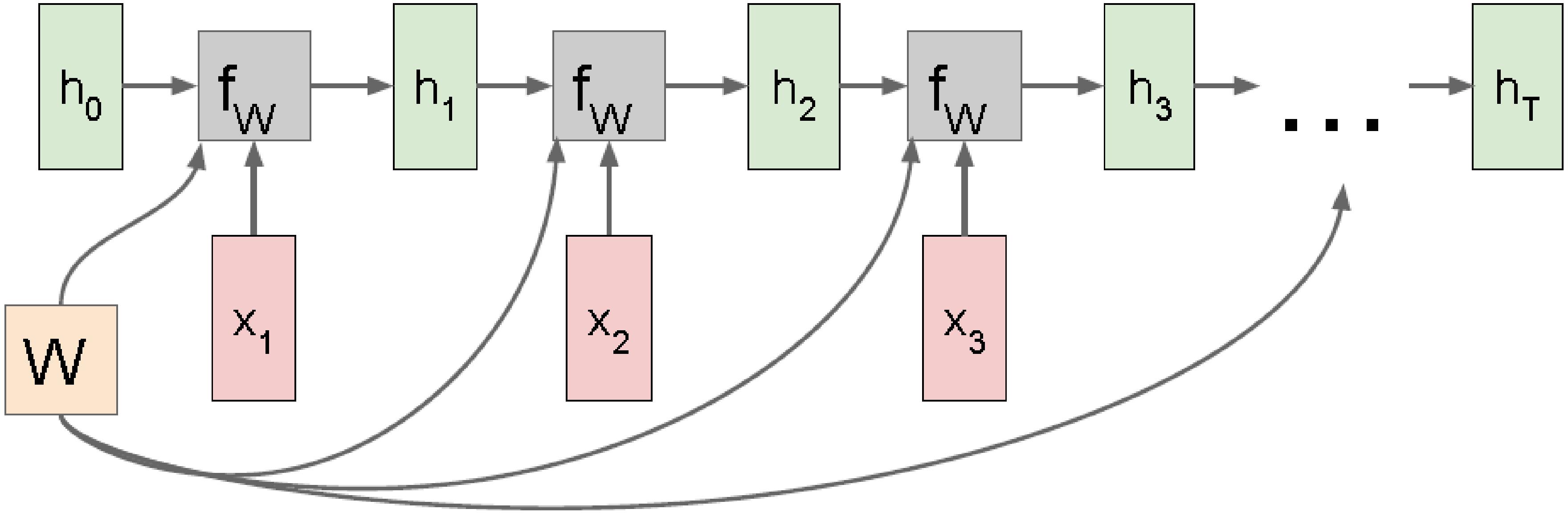


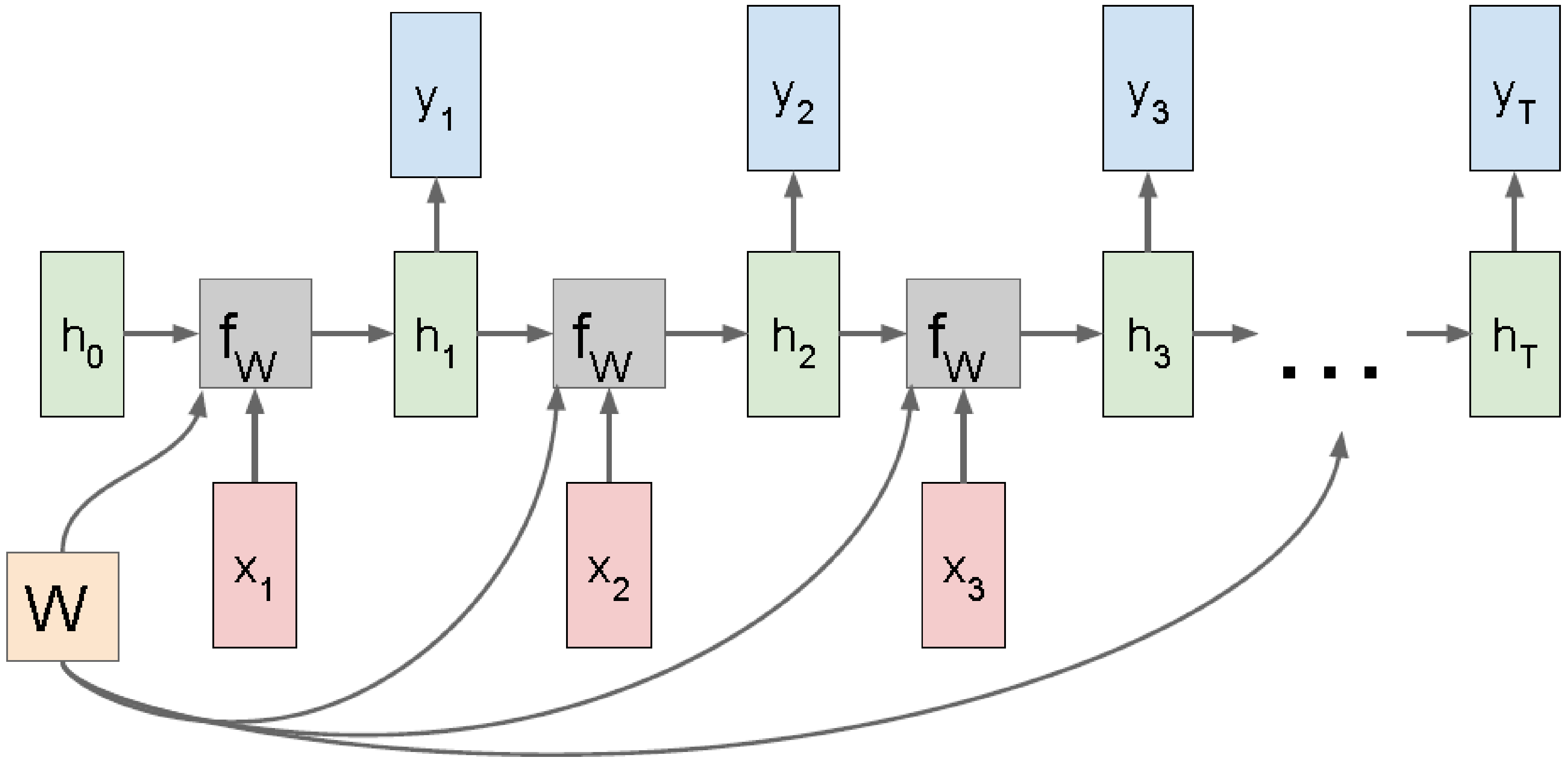
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

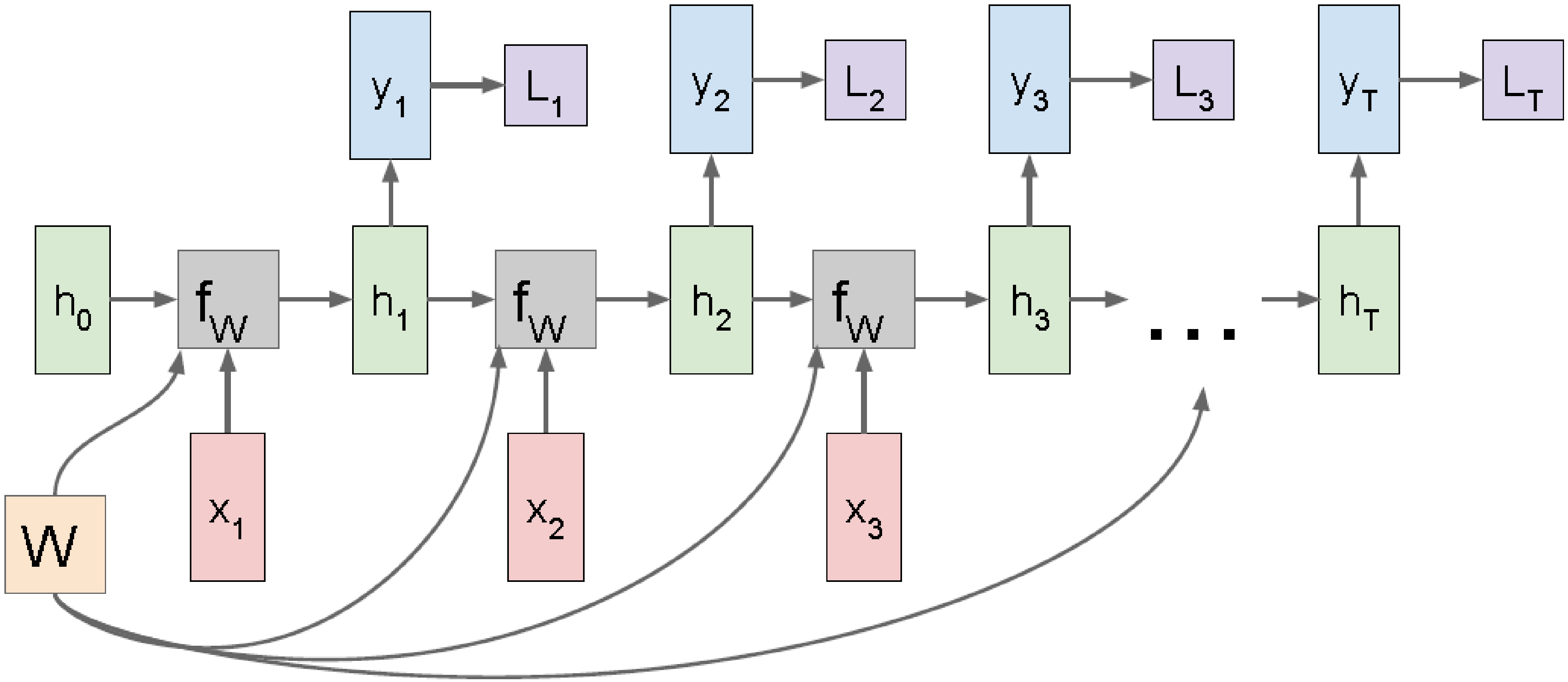
$$y_t = W_{hy}h_t$$

Re-use the same weight matrix at every time-step



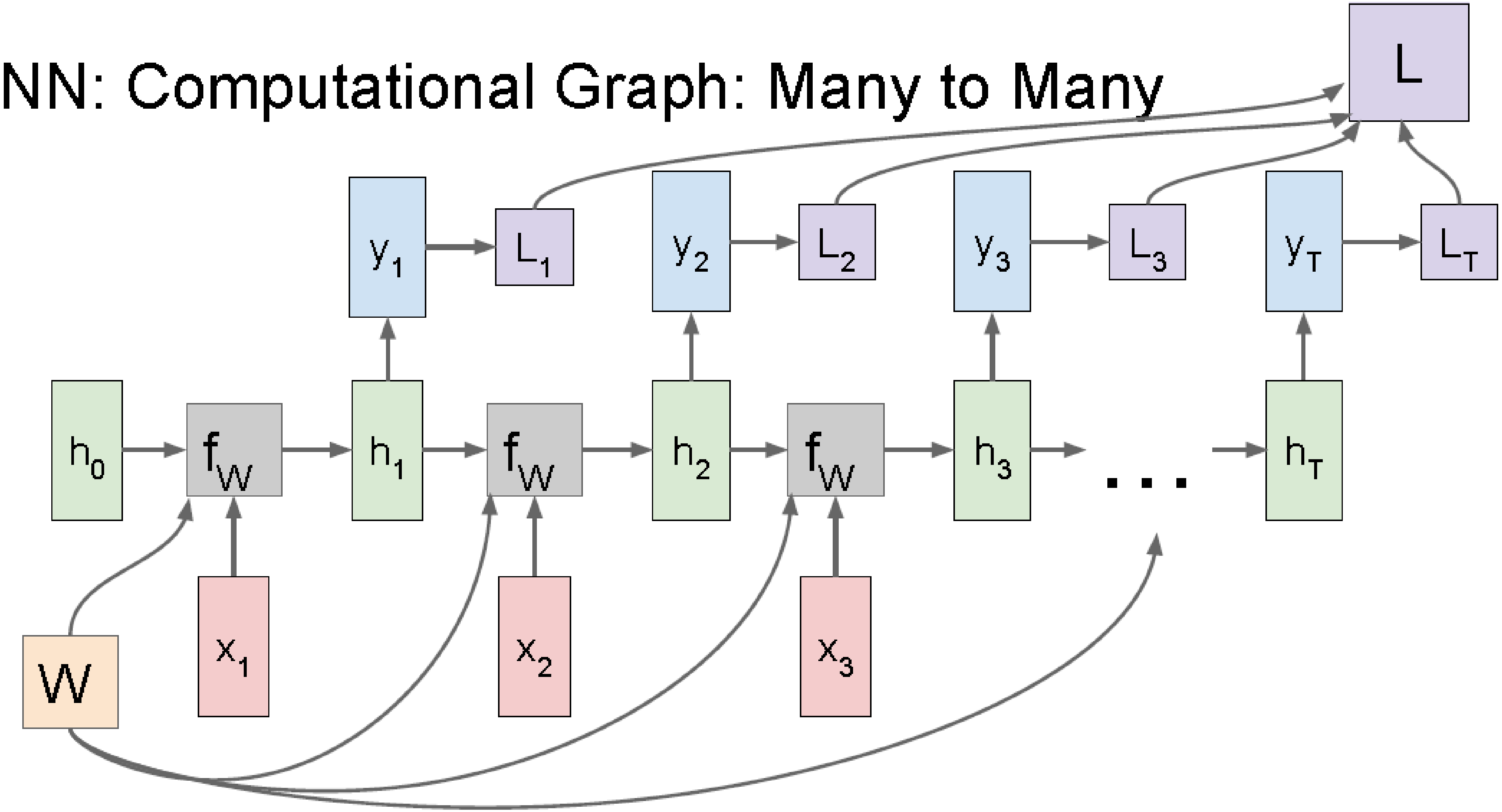


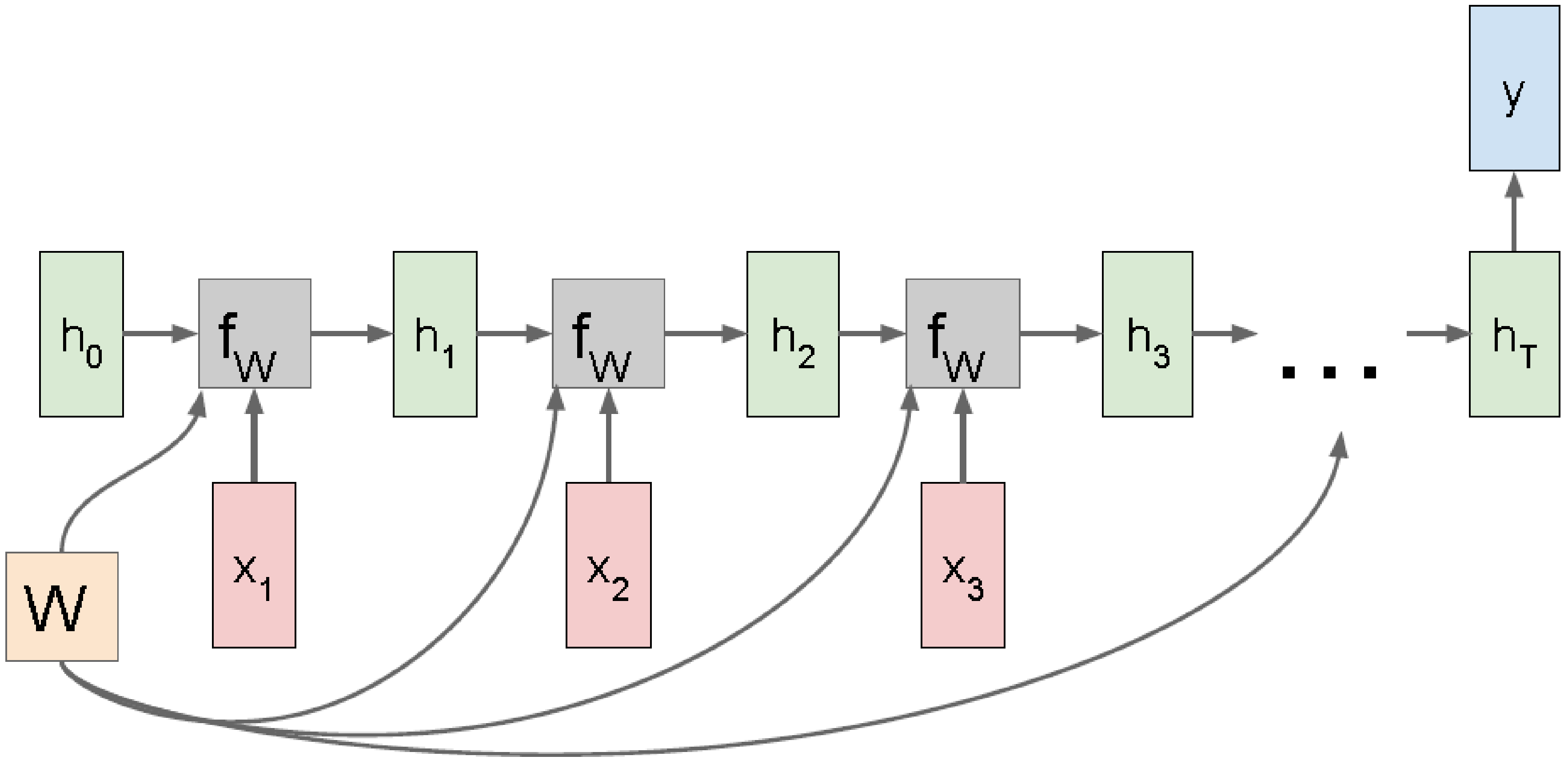
y_1
y_2
y_3
y_T
h_0
f_W
h_1
f_W
h_2
f_W
h_3
...
h_T
x_1
x_2
x_3
W

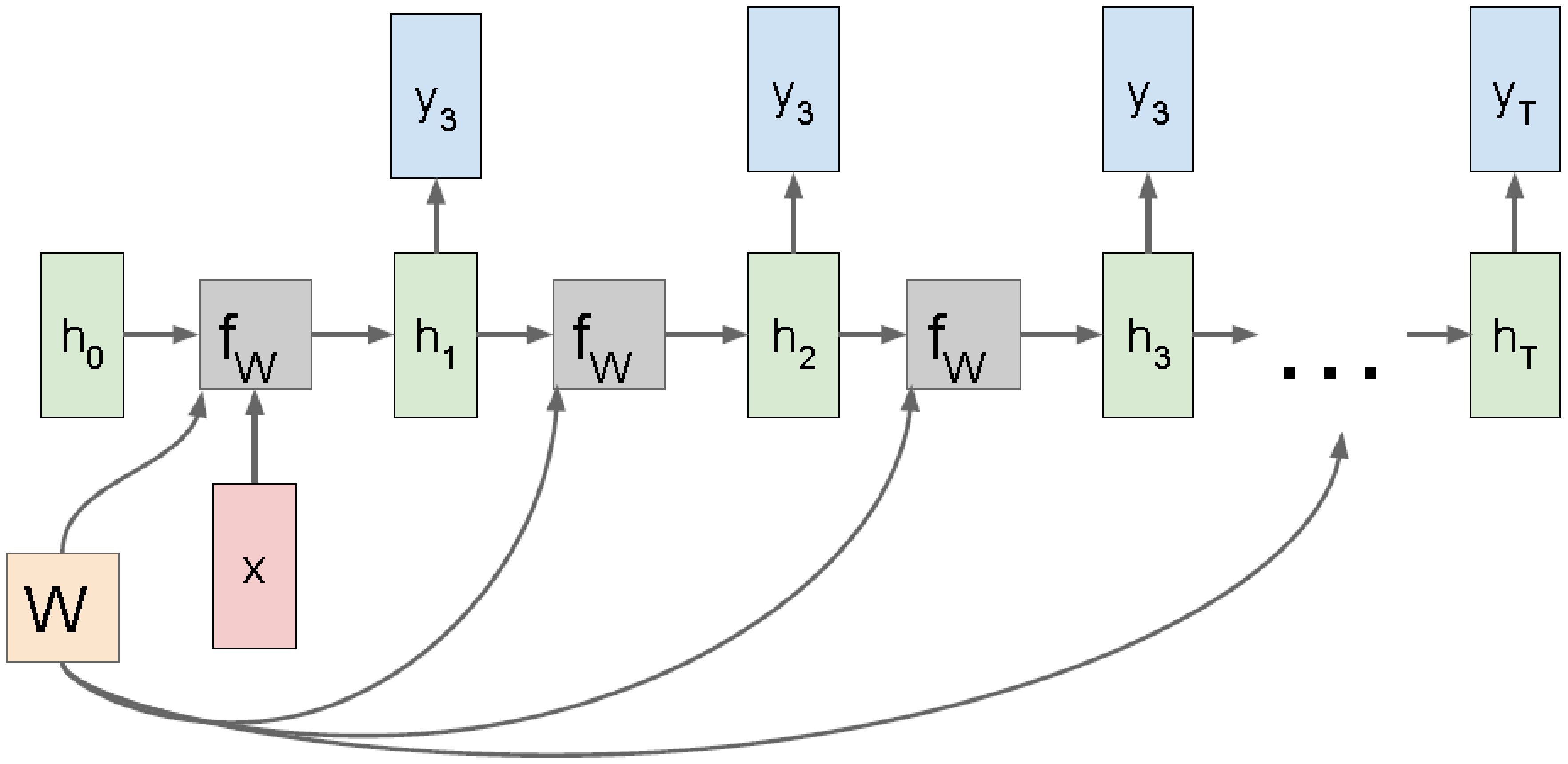


y_1
L_1
y_2
L_2
y_3
L_3
y_T
L_T
h_0
f_W
h_1
f_W
h_2
f_W
h_3
...
h_T
x_1
x_2
x_3
W

RNN: Computational Graph: Many to Many

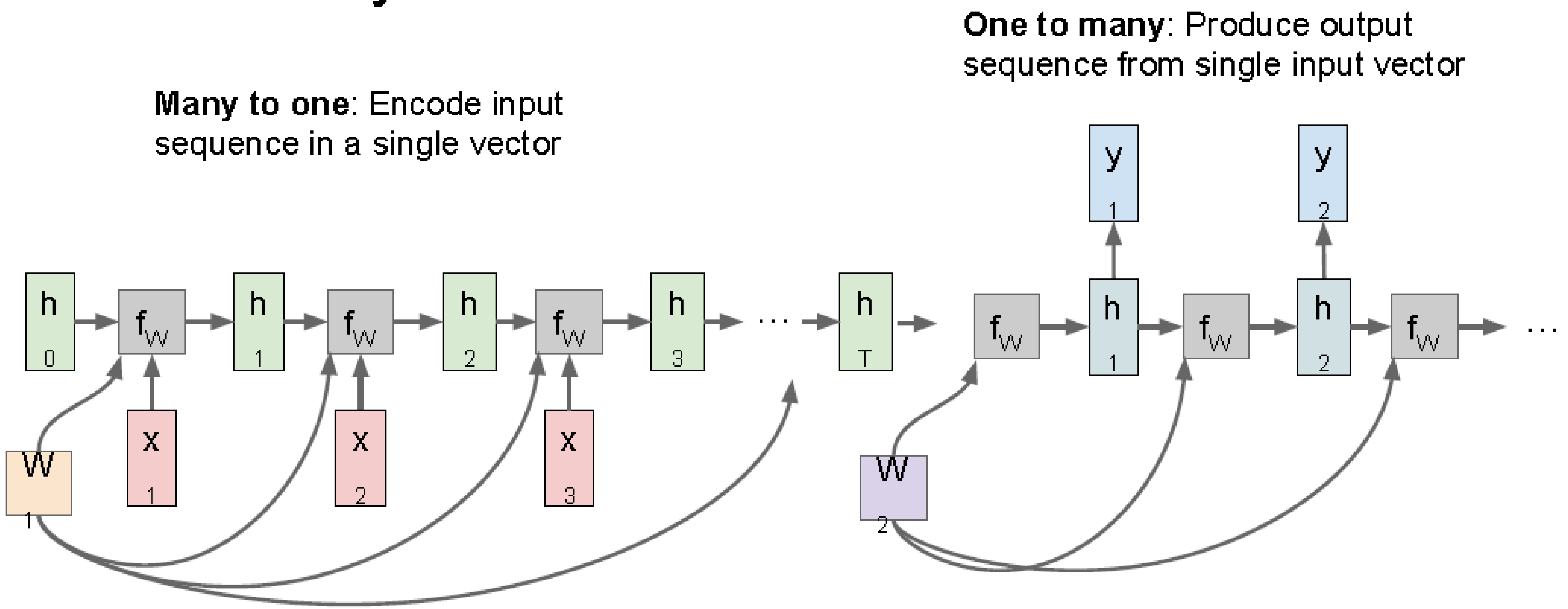



y
h0
fW
h1
fW
h2
fW
h3
...
hT
W
x1
x2
x3



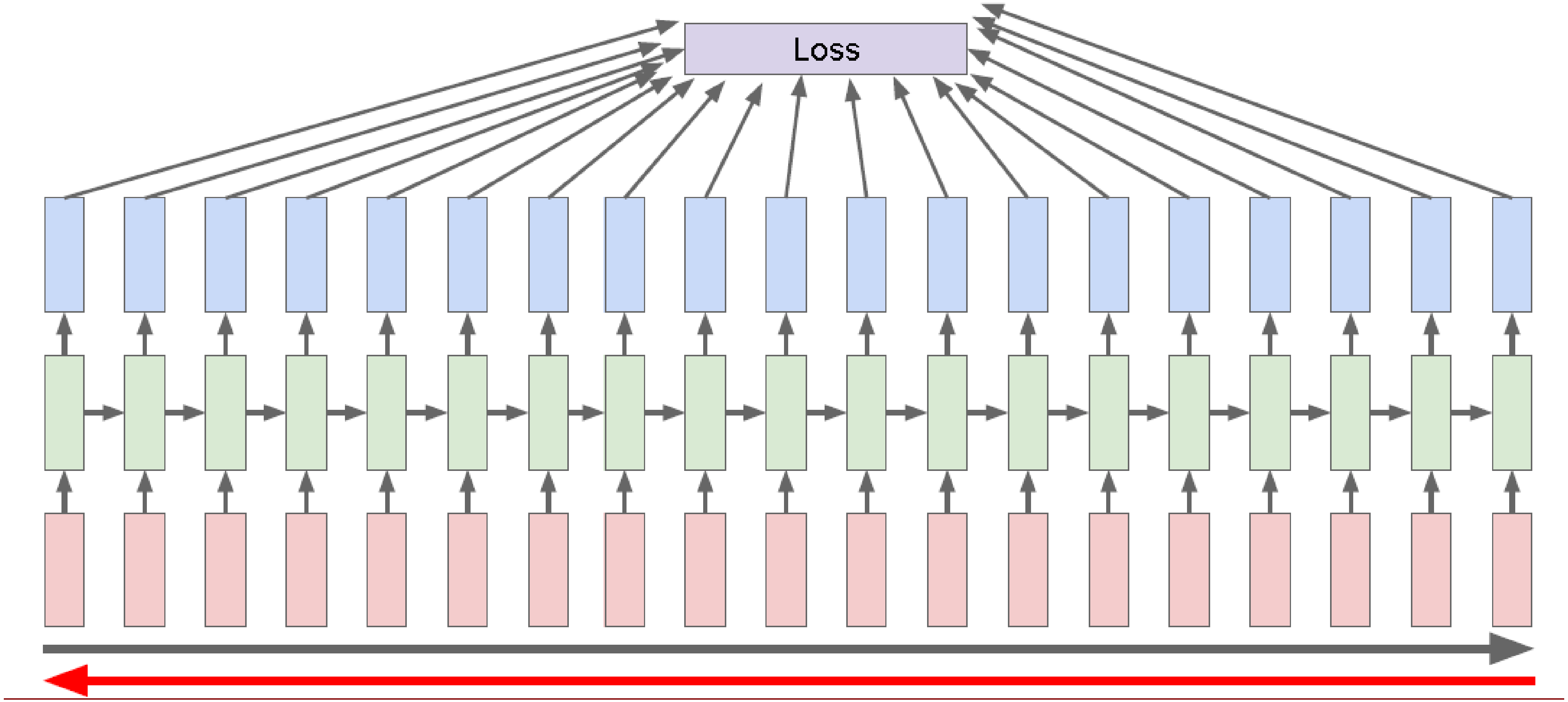
y_3
y_3
y_3
y_T
h_0
f_W
h_1
f_W
h_2
f_W
h_3
. . .
h_T
W
x

Sequence to Sequence: Many-to-one + one-to-many

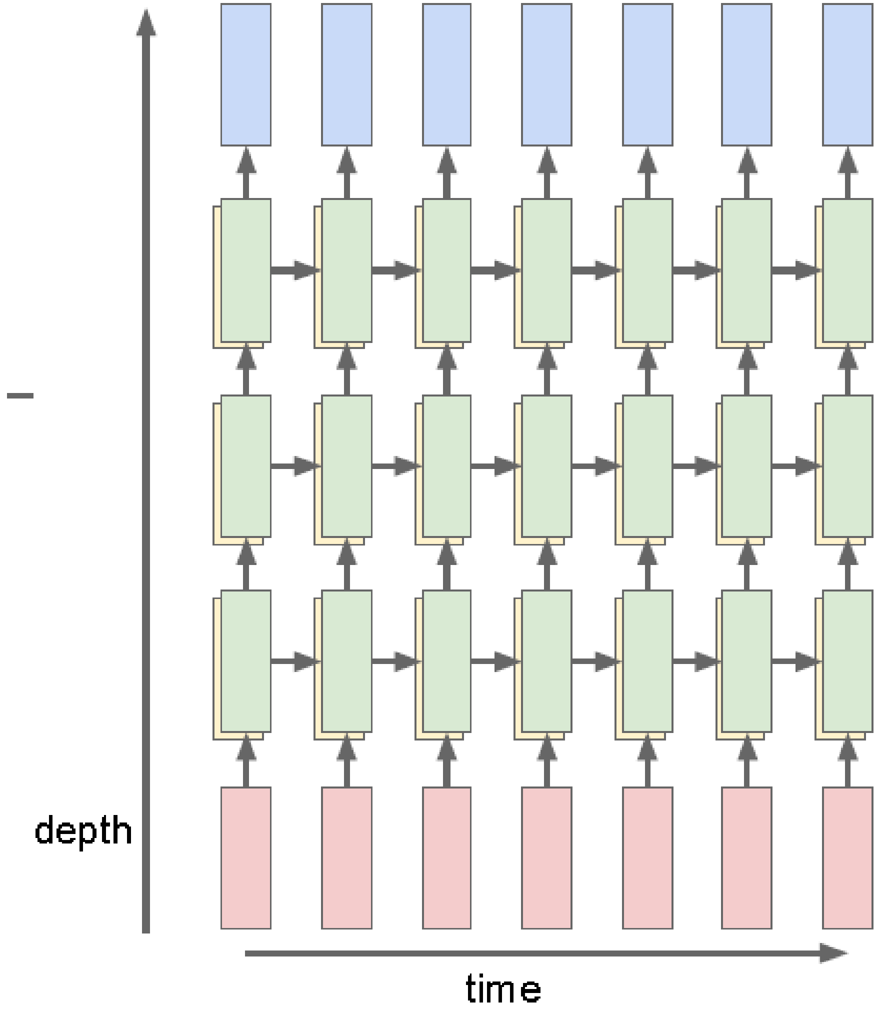


Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



Multilayer RNNs



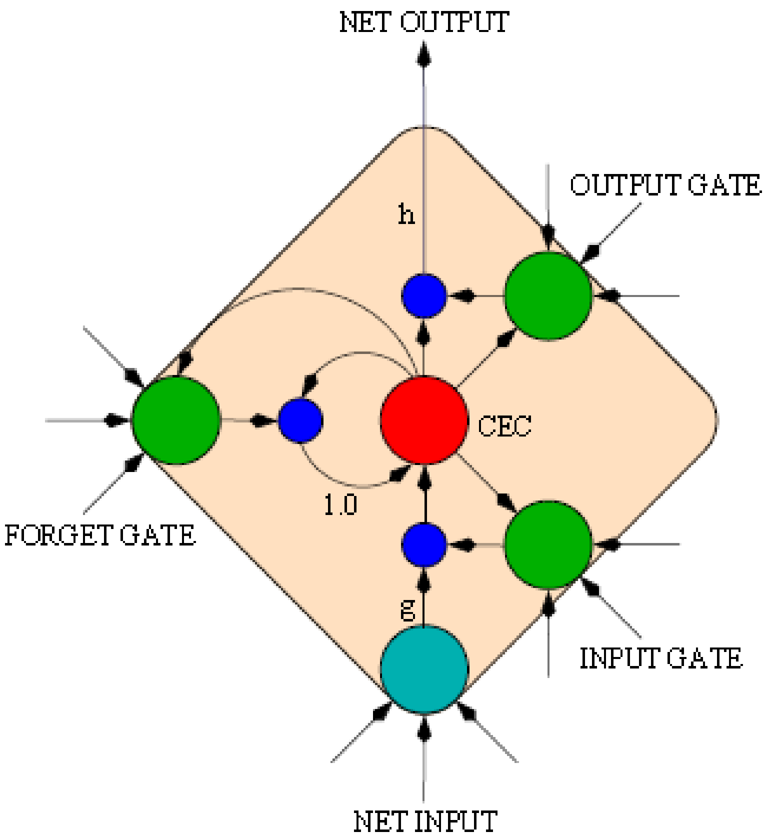
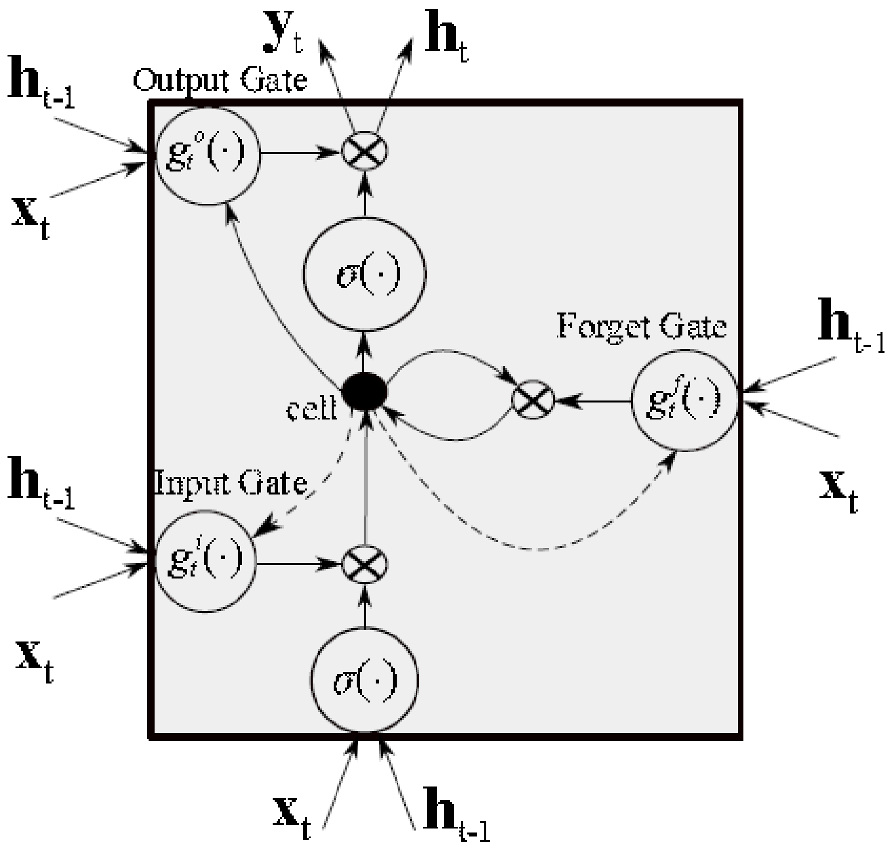
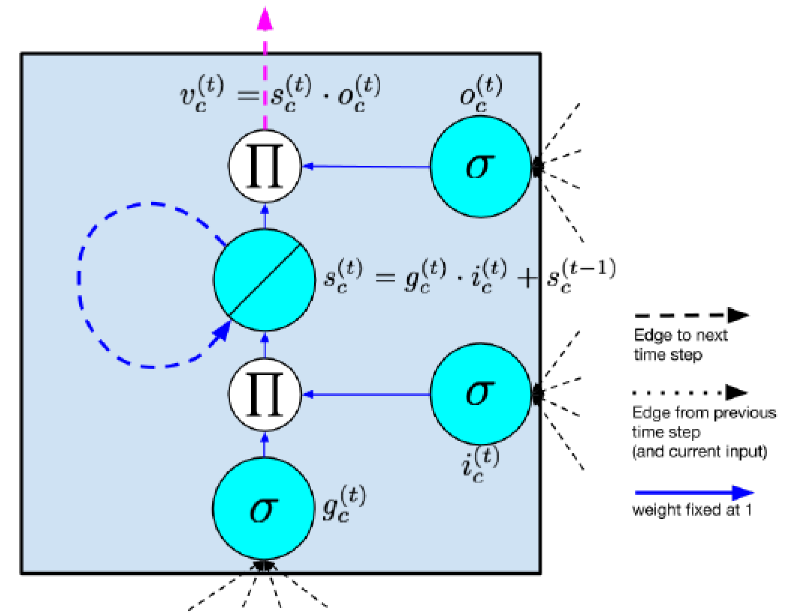
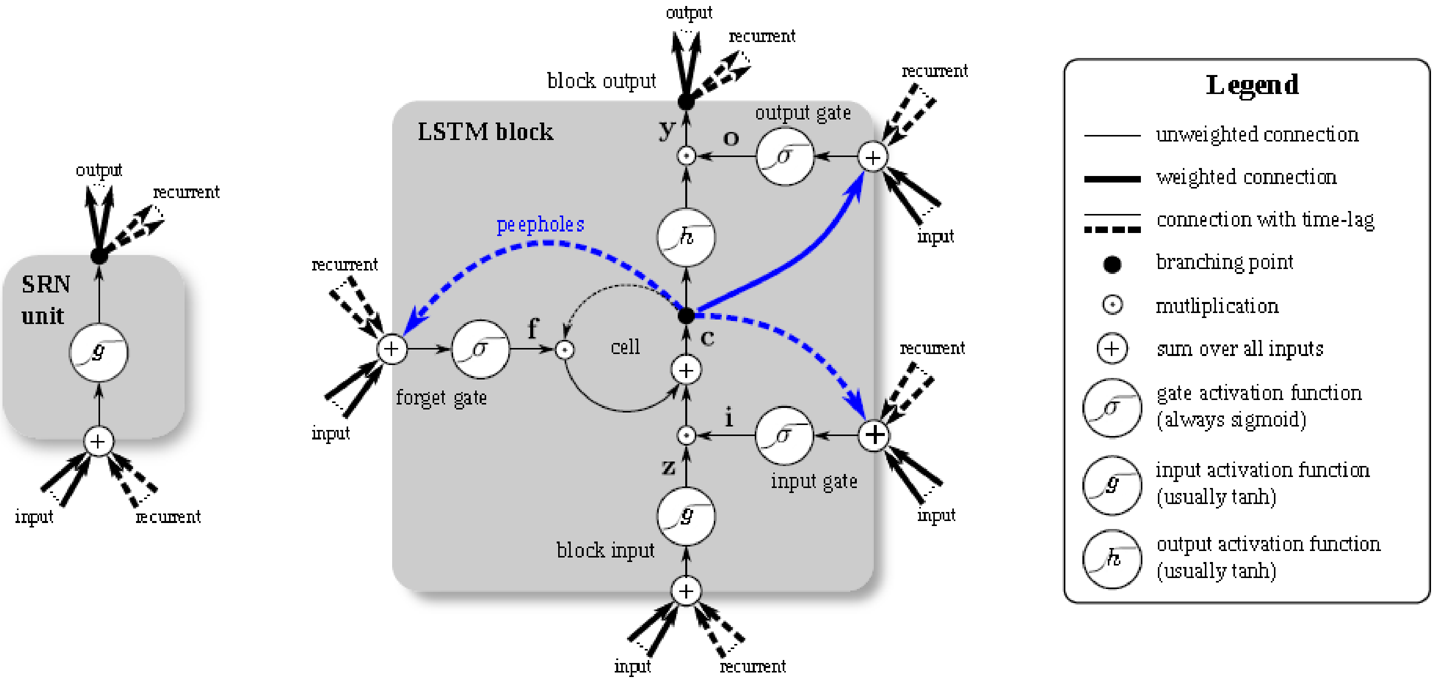


Figure 4.2: **LSTM memory block with one cell.** The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell while the forget gate scales the internal state. The cell input and output activation functions (g and h) are applied at the indicated places.

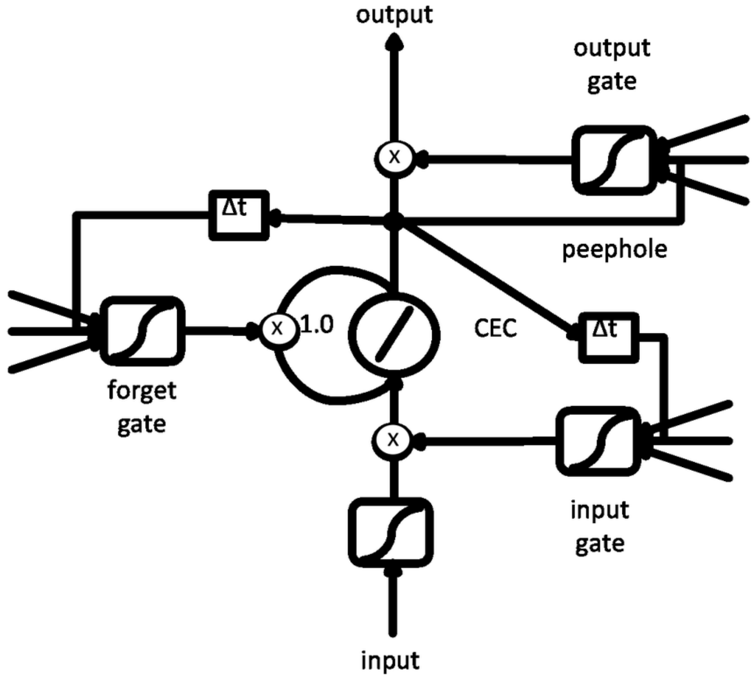




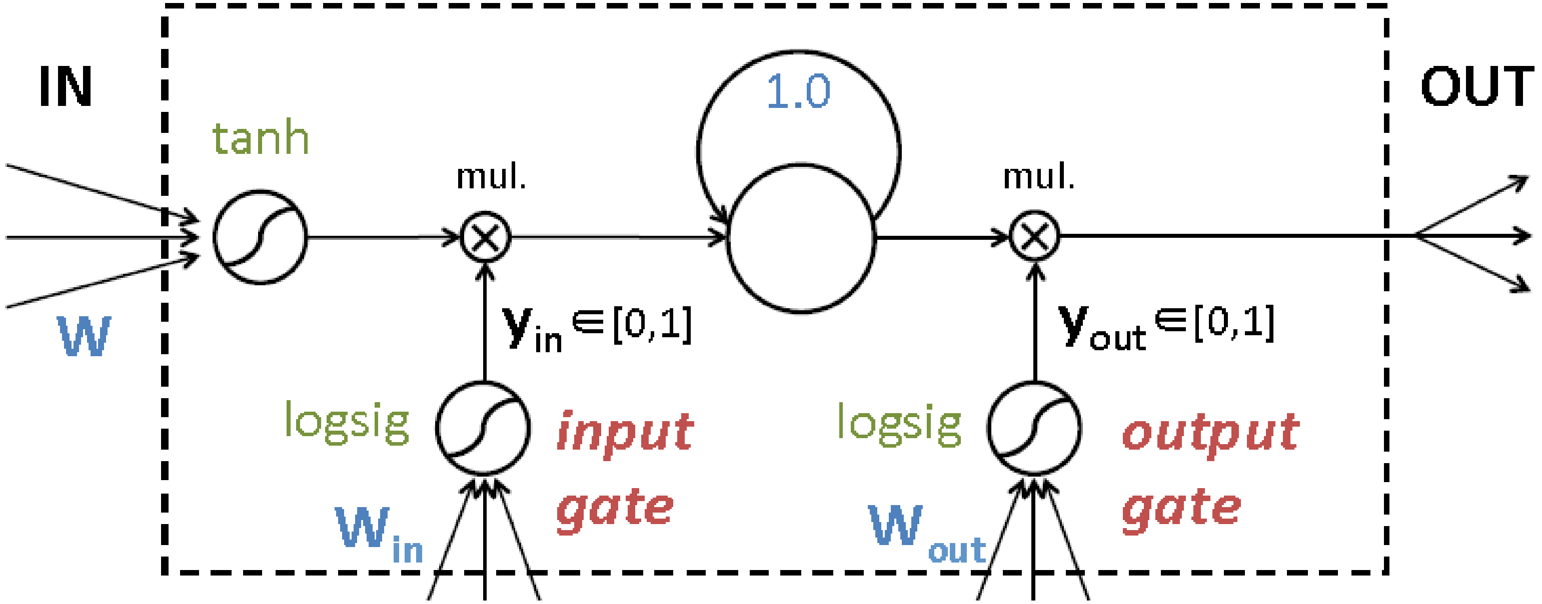
$v_c^{(t)} = s_c^{(t)} \cdot o_c^{(t)}$
$o_c^{(t)}$
$s_c^{(t)} = g_c^{(t)} \cdot i_c^{(t)} + s_c^{(t-1)}$
$i_c^{(t)}$
$g_c^{(t)}$
Edge to next time step
Edge from previous time step (and current input)
weight fixed at 1



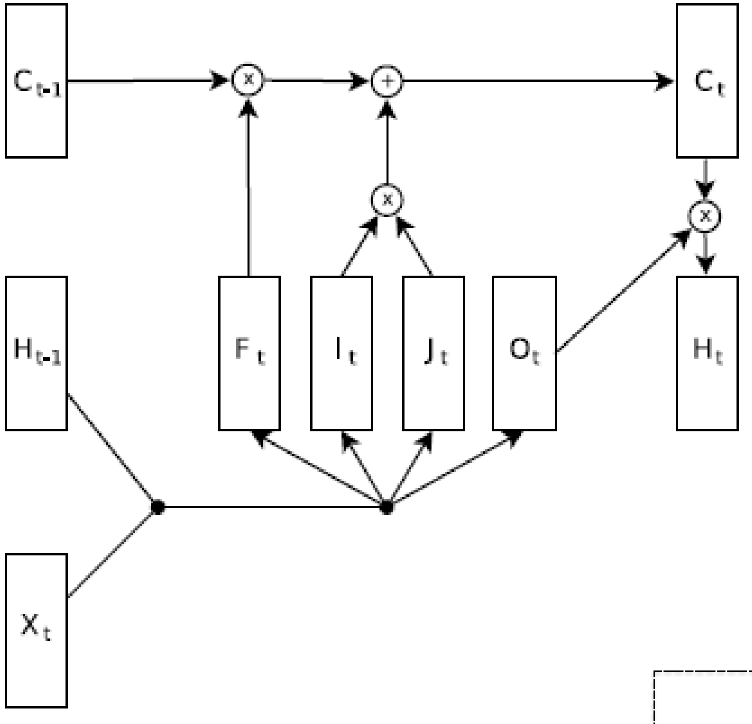
SRN unit
output
recurrent
input
recurrent
LSTM block
block output
output gate
peepholes
forget gate
cell
input gate
block input
input
recurrent
Legend
unweighted connection
weighted connection
connection with time-lag
branching point
mutliplication
sum over all inputs
gate activation function (always sigmoid)
input activation function (usually tanh)
output activation function (usually tanh)



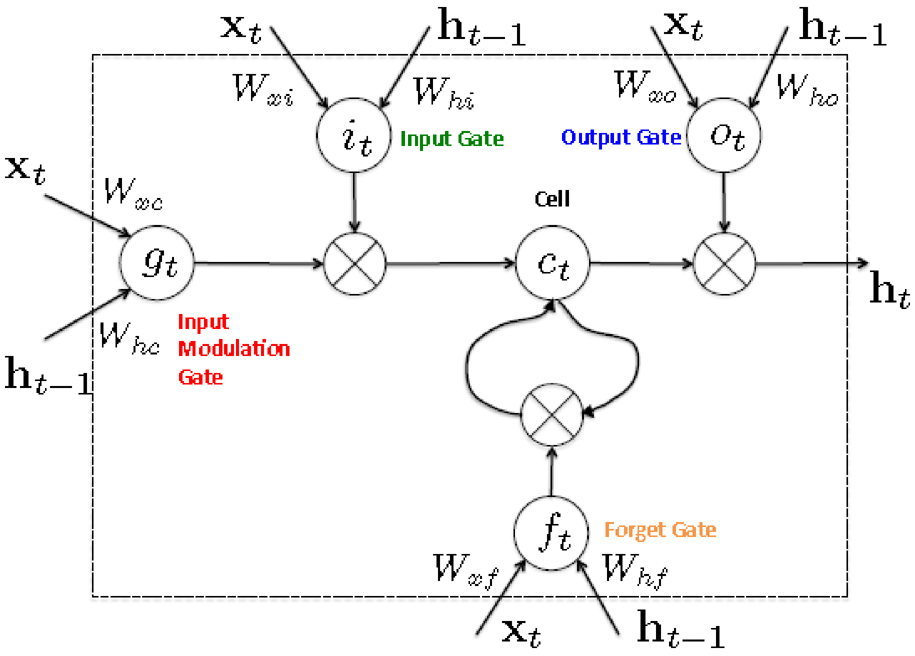
output
output gate
Δt
peephole
1.0
CEC
forget gate
input gate
input



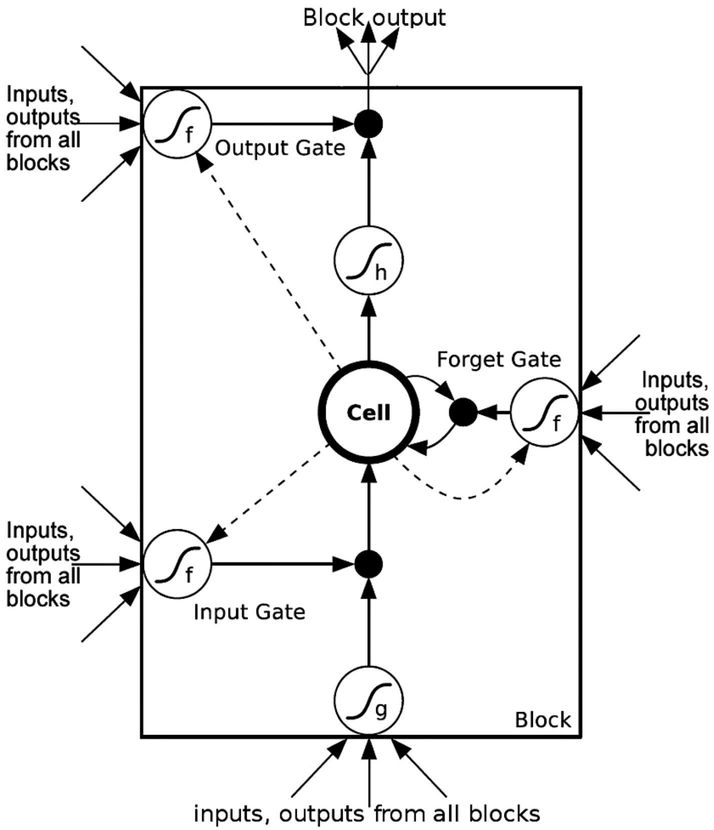
IN
W
tanh
mul.
1.0
mul.
OUT
$y_{in} \in [0,1]$
$y_{out} \in [0,1]$
logsig
input gate
W_{in}
logsig
output gate
W_{out}



C_{t-1}
C_t
H_{t-1}
F_t
I_t
J_t
O_t
H_t
X_t



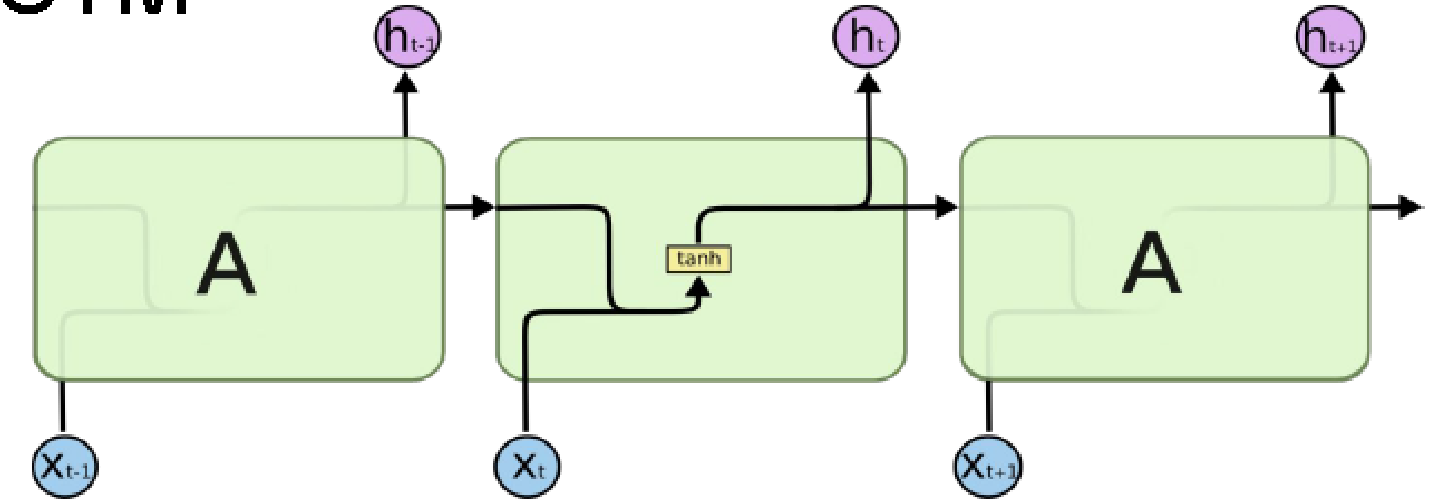
$\mathbf{x}_t$
$\mathbf{h}_{t-1}$
W_{xi}
W_{hi}
i_t
Input Gate
W_{xo}
W_{ho}
Output Gate
o_t
Cell
W_{xc}
g_t
c_t
$\mathbf{h}_t$
W_{hc}
Input Modulation Gate
f_t
Forget Gate
W_{xf}
W_{hf}



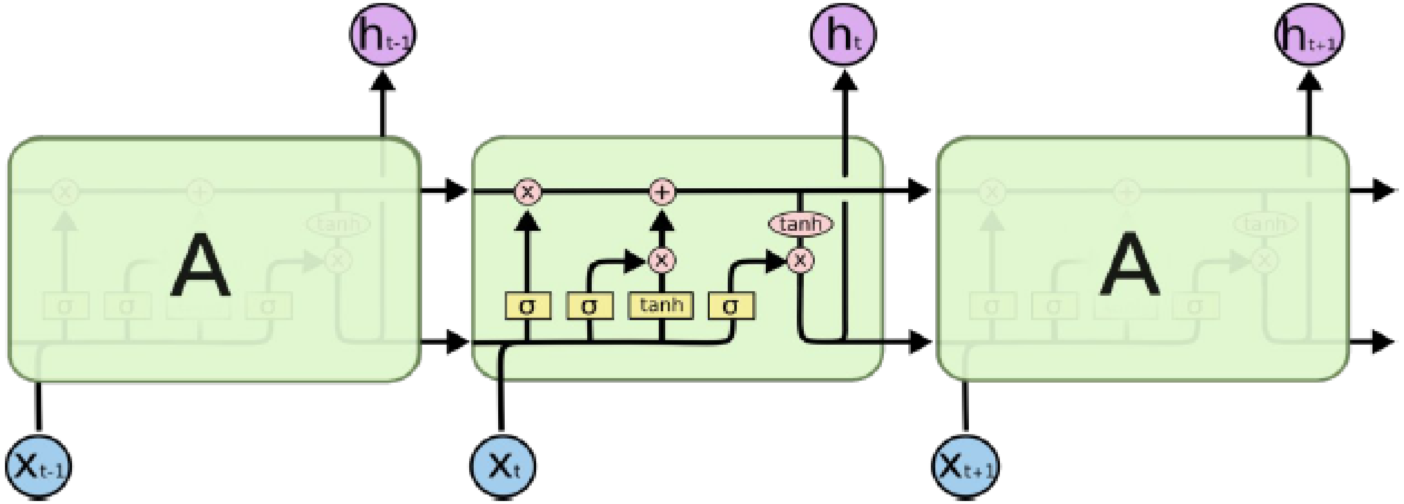
Block output
Inputs, outputs from all blocks
Output Gate
Forget Gate
Cell
Input Gate
Block
inputs, outputs from all blocks

Vanilla RNN vs LSTM

Vanilla RNN cell



LSTM cell



Input gate

Forget gate

Output gate

Hidden state from time t-1

Cell state at time t-1

Cell state at time t

$$i = \sigma(x_t U^i + s_{t-1} W^i)$$
$$f = \sigma(x_t U^f + s_{t-1} W^f)$$
$$o = \sigma(x_t U^o + s_{t-1} W^o)$$
$$g = \ tanh(x_t U^g + s_{t-1} W^g)$$
$$c_t = c_{t-1} \circ f + g \circ i$$
$$s_t = \tanh(c_t) \circ o$$

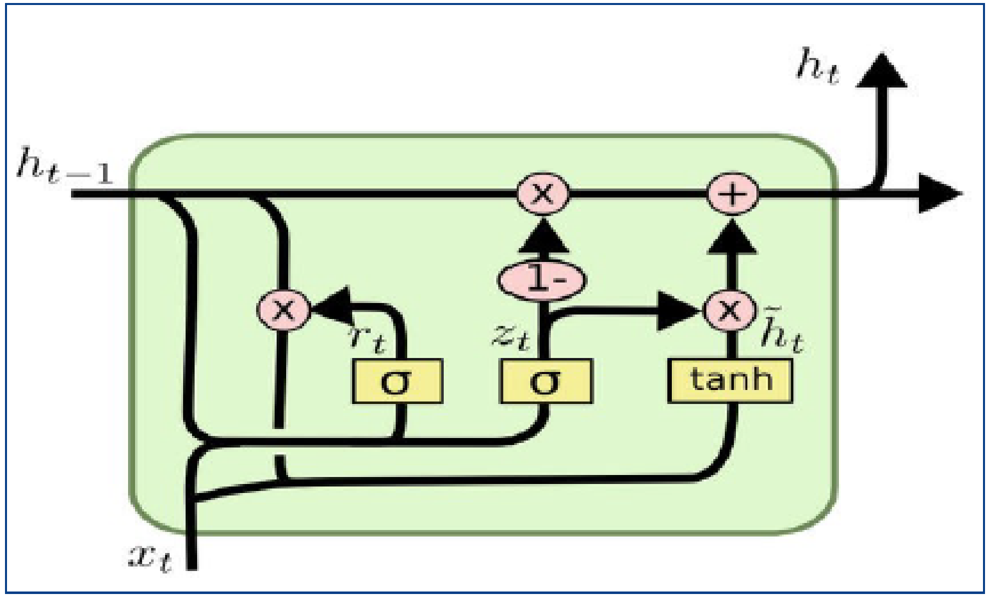
Element-wise Multiplication

Hidden state at time t (to be passed on to next cell at t+1)

- Bias ignored
- Note the activation functions used. Why?

Gated Recurrent Unit (GRU)

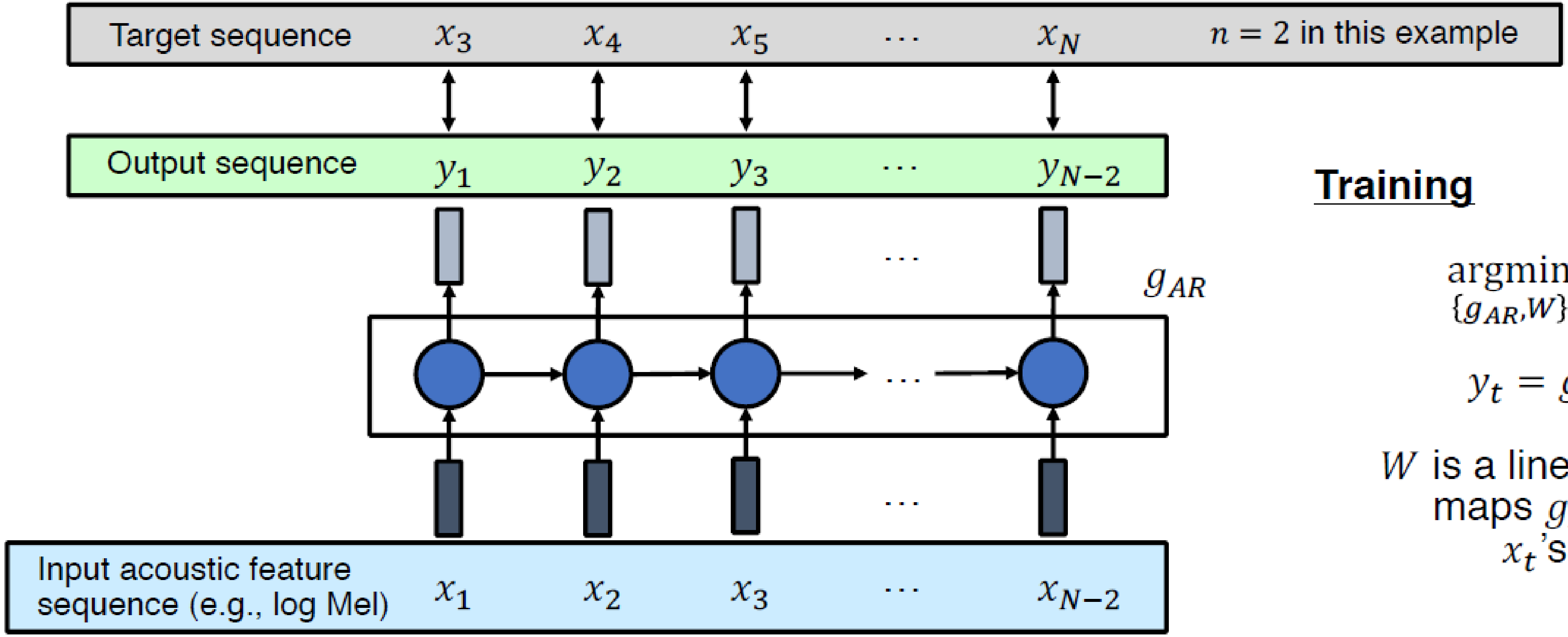
- A dramatic variant of LSTM
  - It combines the forget and input gates into a single update gate
  - It also merges the cell state and hidden state, and makes some other changes
  - The resulting model is simpler than LSTM models
  - Has become increasingly popular



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Autoregressive Predictive Coding (APC)

- Given a previous context $(\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_t})$, APC tries to predict a future audio feature $\boldsymbol{x_{t+n}}$ that is $\boldsymbol{n}$ steps ahead of $\boldsymbol{x_t}$
  - Uses an autoregressive model g_{AR} to summarize history and produce output
  - $n \geq 1$ encourages g_{AR} to infer more global underlying structures of the data rather than simply exploiting local smoothness of speech signals



Training

$$\underset{\{g_{AR}, W\}}{\operatorname{argmin}} \ \sum_{t=1}^{N-n} |x_{t+n} - y_t|,$$

$$y_t = g_{AR}(x_1, \dots, x_t) \cdot W$$

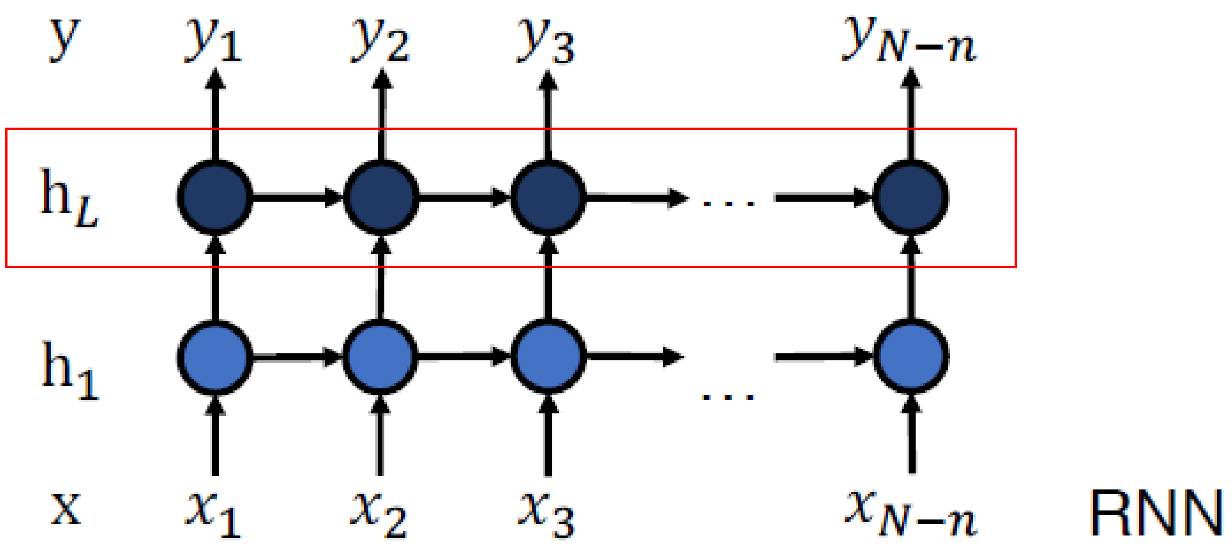
W is a linear transformation that maps g_{AR}'s output back to x_t's dimensionality

Types of autoregressive model g_{AR}

- g_{AR}
  - Input: $\mathbf{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N})$
  - Output: $\mathbf{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_N})$

- L-layer Unidirectional RNN:

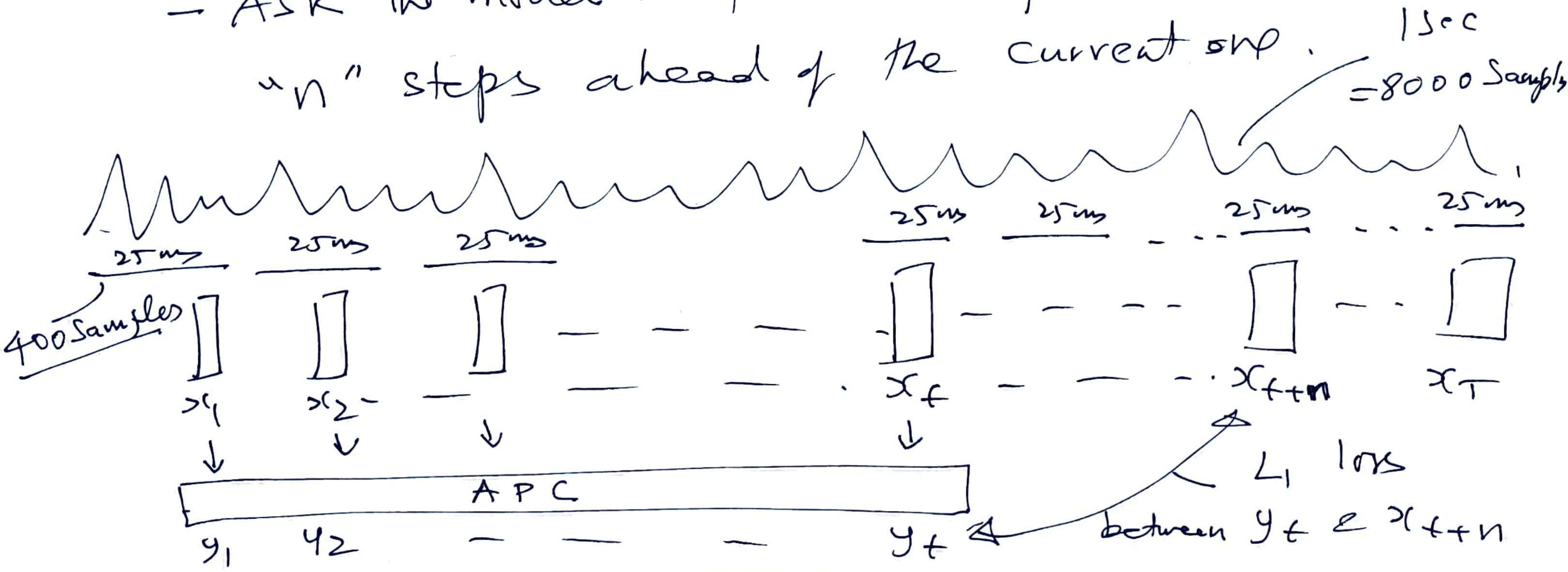
$$\mathrm{h}_0 = \mathrm{x}$$
$$\mathrm{h}_l = \mathrm{RNN}^{(l)}(h_{l-1}), \forall l \in [1, L]$$
$$\mathrm{y} = \mathrm{h}_L \cdot W$$



- Feature extraction: $\mathbf{h}_L$

APC: Auto regressive Predictive Coding

- APCs exploit local smoothness of the speech signal
  - to predict "next" frame.
- To encourage APCs to infer more "global structures"
  - Rather than local information in the signal.
  - Ask the model to predict a frame "n" steps ahead of the current one.



Given an utterance represented as a sequence of acoustic feature vectors

$$x_1 \quad x_2 - - - - - x_t - - - - x_{t+n} - - - x_T$$

RNN/LSTM/GRU – processes $[x_1 \dots x_t]$ to yield an output prediction y_t

$$x_t \in \mathbb{R}^d \quad , \quad y_t \in \mathbb{R}^d$$

History for y_t

$$\boxed{x_1 \quad x_2 - - - - x_t} \quad x_{t+1} - - - x_{t+n} - - - x_T$$

L_1 loss.

$$\downarrow$$

$$\boxed{\text{RNN}}$$

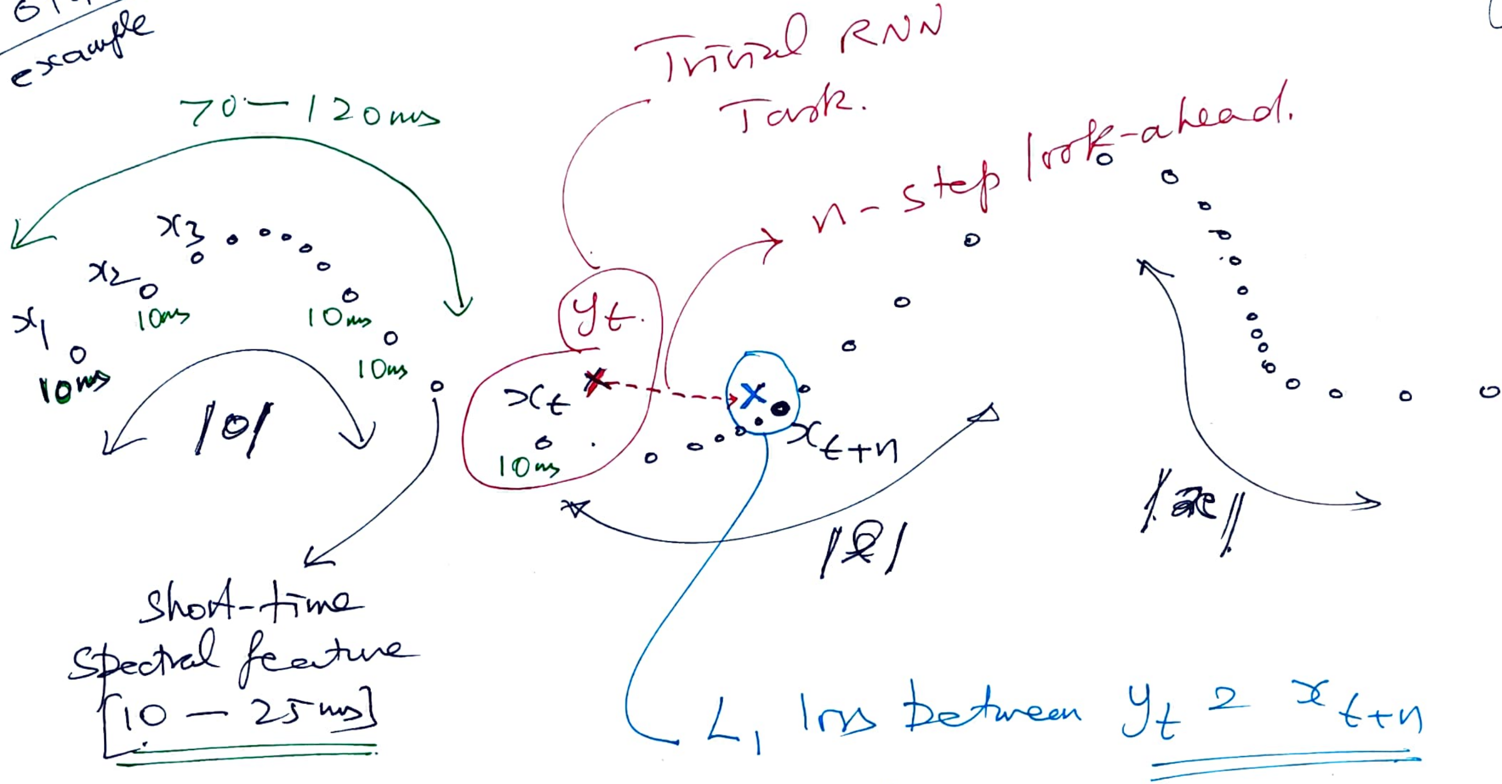
$$\downarrow$$

$$y_1 \quad y_2 - - - - \boxed{y_t}$$

$$L_1 \text{ loss} = \sum_{t=1}^{T-n} \left| x_{t+n} - y_t \right|$$

Model is optimised by minimizing L_1 loss

o/a
example



- Within phone spectral Continuity
  - Articulatory constraints [slow moving articulators]
  - Coarticulation effects [across adjacent phones]

n - hyperparameter

- $n = 1 \rightarrow$ objective ~ LM ~ Too simple a task
- $n \uparrow \rightarrow$ More difficult the task.

$X_{1:T}$: $x_1\ x_2\ \cdots\ x_t\ \cdots\ x_{t+n}\ \cdots\ x_T$

$\big\downarrow f \rightarrow$ RNN / LSTM / GRU

$H_{1:T}$: $h_1\ h_2\ \cdots\ h_t\ \cdots$ h_T

$\big\downarrow g \rightarrow$ Linear Projection
[Regression layer that predicts x_{t+n} as y_{t+n} with h_t

y_{t+n}

$L_t = \| y_{t+n} - x_{t+n} \|_1$

$$H_{1:T} = f(X_{1:T})$$

$$y_{t+n} = g(h_t) = W^T h_t$$

$$L_t = \| y_{t+n} - x_{t+n} \|_1$$

Optimize f & g to minimize L_t over $t = 1, \ldots, T-n$

Table 1: *Comparing APCs with a series of CPC models on phone classification. PERs are reported.*

Method	#(step)			
	2	5	10	20
`cpc-n9all`	51.3	48.8	50.8	54.6
`cpc-n9same`	47.5	48.2	50.0	53.0
`cpc-ctx-n9same`	42.1	46.1	48.8	53.8
`cpc-ctx-exhaust`	42.9	43.1	45.6	49.1
apc (proposed)	36.5	35.6	35.4	37.7

Table 2: *PERs on phone classification. All features are fed to a linear classifier unless otherwise stated. The number of steps to the target #(steps) is not relevant in the first four rows.*

Method	#(step)					
	1	2	3	5	10	20
Mel	50.0					
Mel + MLP-1	43.4					
Mel + MLP-3	41.3					
cpc best	42.1					
apc 1-layer	39.4	36.5	35.4	35.6	35.4	37.7
apc 2-layer	38.5	34.6	35.9	35.7	34.6	38.8
apc 3-layer	37.2	36.7	33.5	36.1	37.1	38.8
apc 4-layer	36.2	34.4	34.5	35.3	36.9	39.6

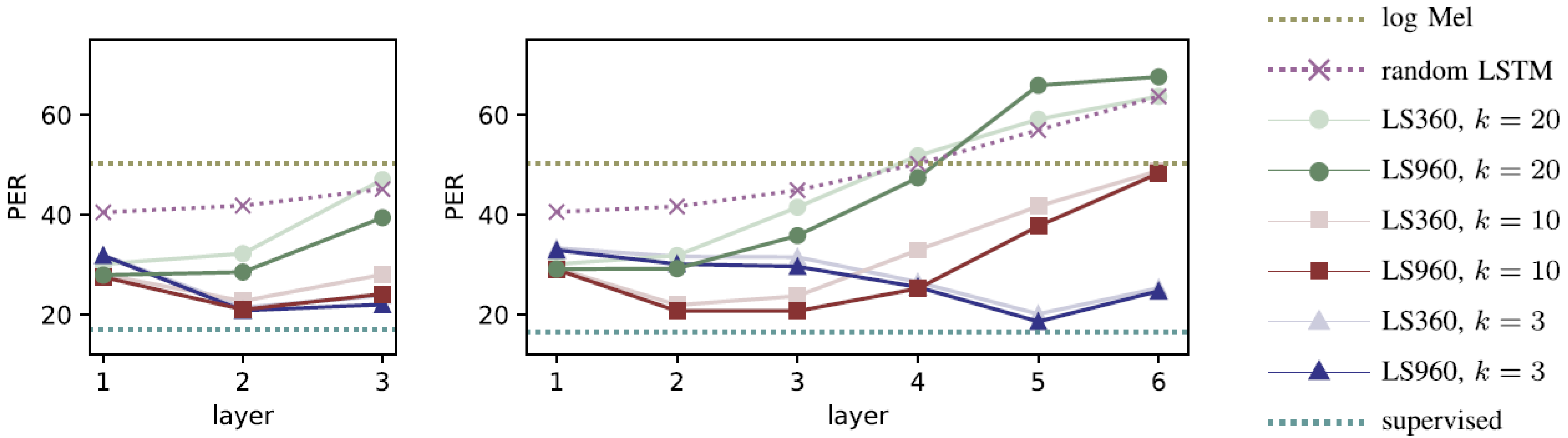


Fig. 1. Phone error rates (PERs) of frame classification on `dev93` with representations produced by 3-layer LSTMs (*left*) and 6-layer LSTMs (*right*). We use LS360 and LS960 to denote the LSTMs trained on the 360-hour subset and the 960 hours combined of LibriSpeech, respectively. We use k to denote the number of time steps into the future in the APC objective.

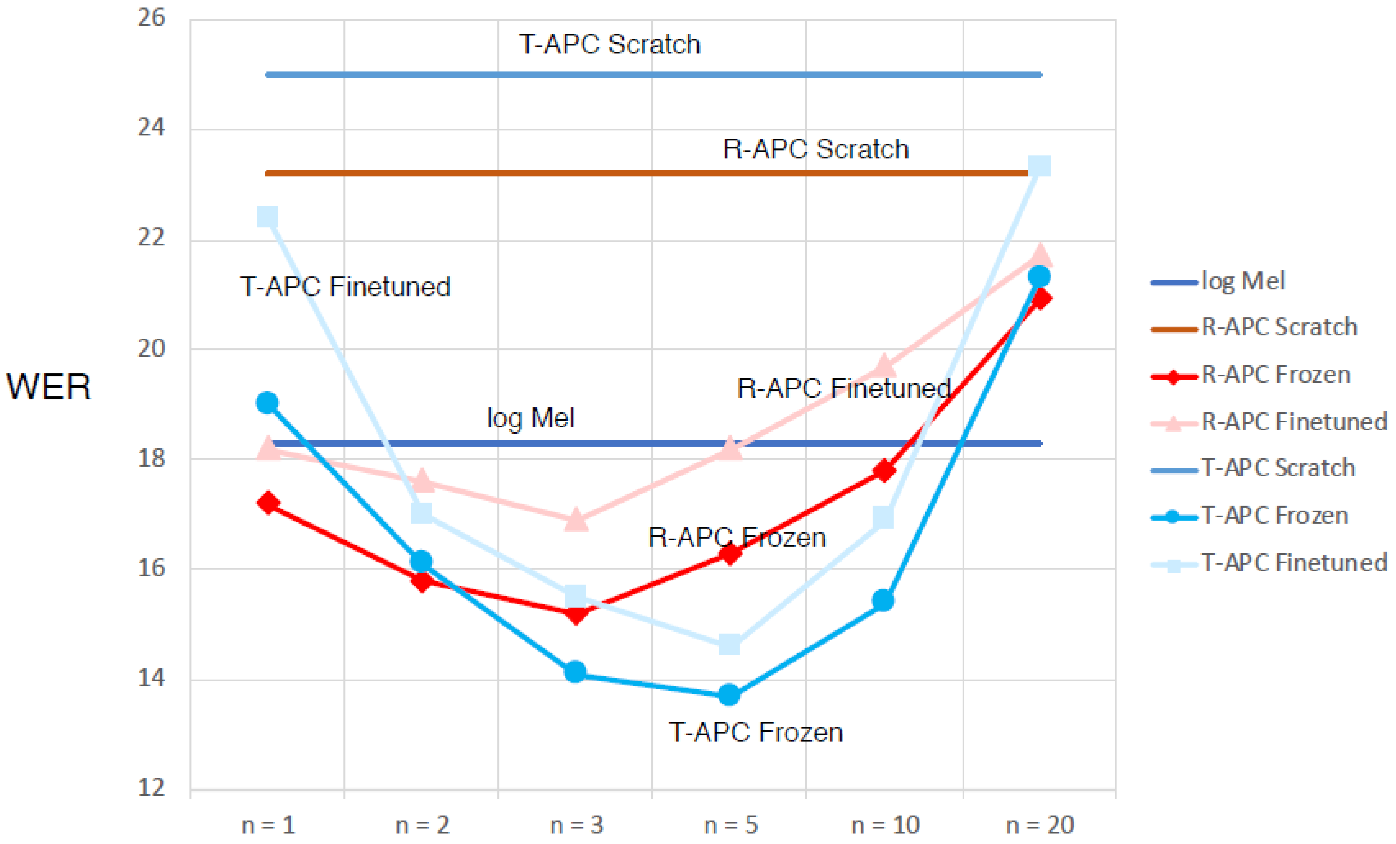
Transfer learning experiments

- Setup: pre-training + fine-tuning
- Pre-training data
  - Speech portion of the LibriSpeech 360 hours subset
  - 921 speakers
  - 80-dimensional log Mel spectrograms as input acoustic features (i.e., $x_t \in \mathbb{R}^{80}$)
  - Use extracted features to replace log Mel as new inputs to downstream models
- Considered downstream tasks
  - Speech recognition
  - Speech translation
  - Speaker identification (skipped in this talk, see paper!)
- Comparing methods
  - Contrastive predictive coding (CPC)
  - Problem-agnostic speech encoder (PASE)

Speech Recognition

- Considered dataset: Wall Street Journal
  - Training: 90% of si284 (~ 72 hours of audio)
  - Validation: 10% of si284
  - Test: dev93
- APC g_{AR}
  - RNNs: 4-layer, 512-dim GRUs
  - Transformers: 4-layer, 512-dim Transformer decoder blocks
- Downstream ASR model
  - Seq2seq with attention [Chorowski et al., 2015]
  - Beam search with beam size = 5
  - No language model rescoring

Choice of $\boldsymbol{n}$, and whether to fine-tune $\boldsymbol{g_{AR}}$



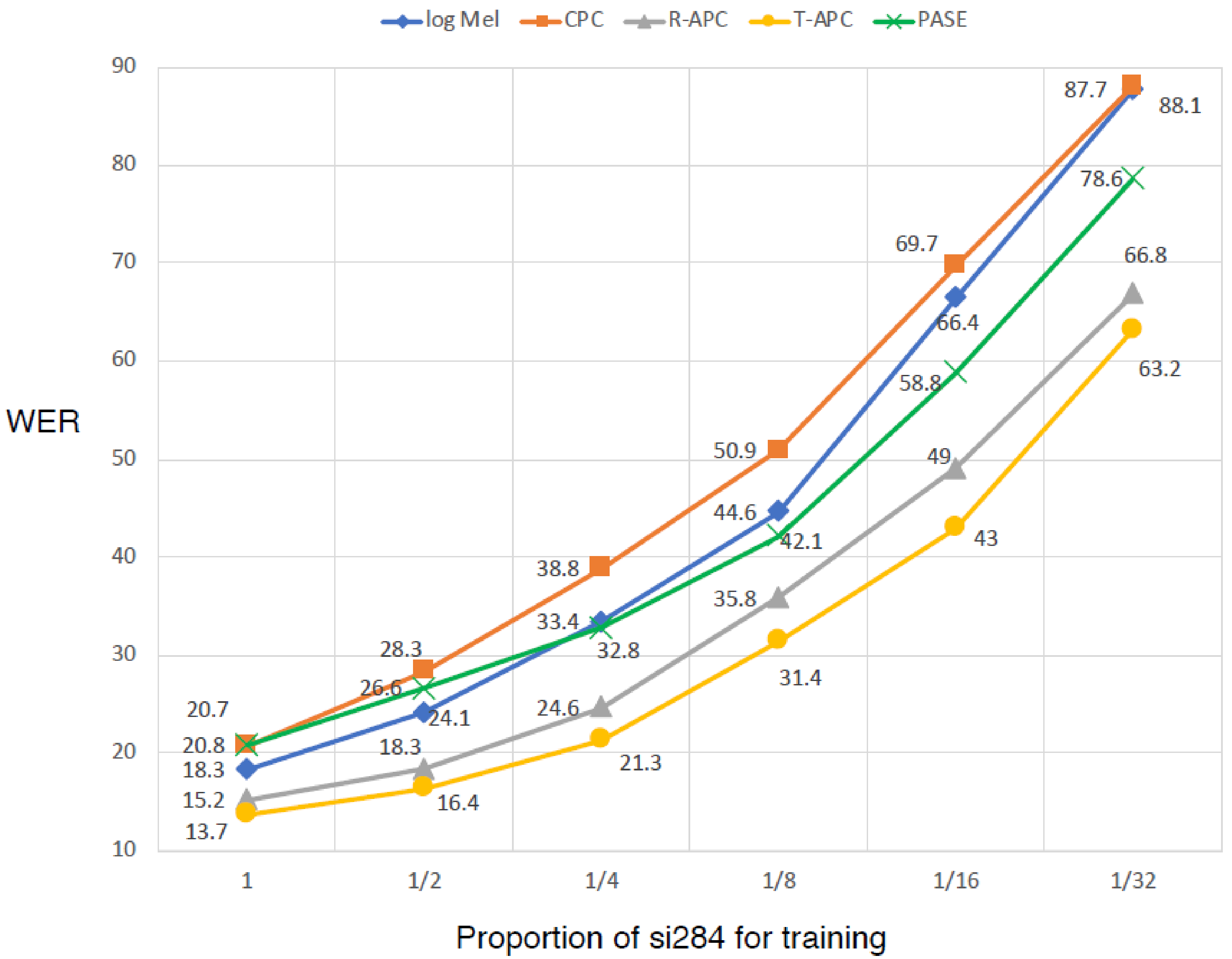
Notations

- R stands for RNN
- T stands for Transformer
- **Scratch**: g_{AR} randomly initialized and concatenate with ASR model
- **Frozen**: keep g_{AR} frozen when training ASR model
- **Finetuned**: fine-tune g_{AR} along with ASR model

Findings

- Sweet spot exists for both Frozen and Finetuned when varying n
- Scratch performance is poor, even worse than log Mel baseline
- APC outperforms log Mel most of the time
- For both R and T, Frozen outperforms Finetuned
- Will use R-APC Frozen with $n = 3$ and T-APC Frozen with $n = 5$ for the rest

APC for reducing the amount of labeled training data

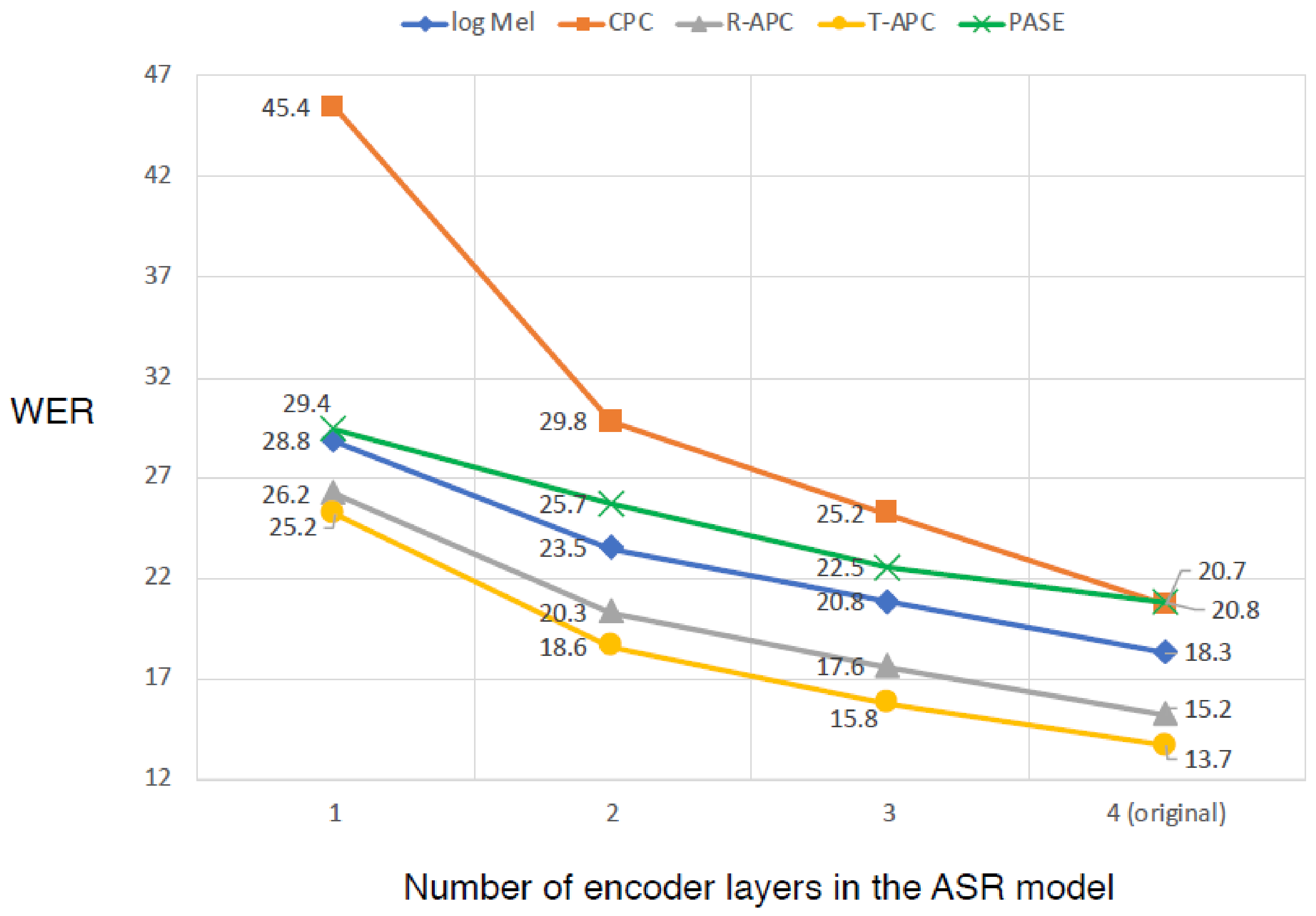


Recap: all feature extractors were pre-trained with 360 hours of LibriSpeech data; we did not fine-tune any feature extractor with the ASR model

Findings

- Full set:
  - 25% and 17% relative improvement for T-APC (13.7) and R-APC (15.2) over log Mel baseline (18.3), respectively
- As we decrease the amount of training data:
  - T-APC (yellow) and R-APC (gray) always outperform other methods
  - Gap between T-APC / R-APC and log Mel (blue) becomes larger
  - Using just half of si284, T-APC (16.4) already outperforms log Mel trained on full set (18.3)
- In the paper we also have the figure where all feature extractors were pre-trained on only 10 hrs of LibriSpeech data. **TLDR**: pre-training still helps even with just 10 hrs of pre-training data

APC for reducing downstream model size



Note: all models trained on full si284

Findings

- T-APC (yellow) and R-APC (gray) always outperform other methods
- T-APC with just 2 layers (18.6) performs similar to log Mel with 4 layers (18.3)

Thank you !!