Data Science for Assistive Health Technologies

Dr. Muhammad Adeel Nisar

Assistant Professor – Department of IT,
Faculty of Computing and Information Technology,
University of the Punjab, Lahore

Contents

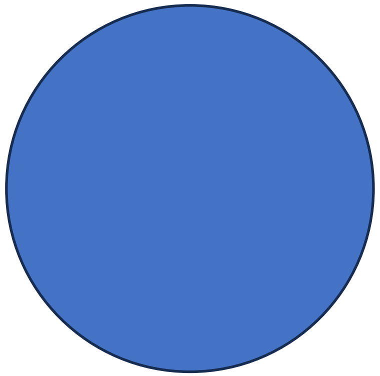
Acknowledgement

- The slides material is acquired from the lectures/publications of
- Andrew Ng, Stanford University,
- Chris Olah, Google Brain,
- Fei-Fei Li, Stanford University,
- Geoffrey Hinton, Google & University of Toronto,
- Alexandar and Ava Amini, MIT
- Hongyi Zhu and Hsinchun Chen, University of Arizona
- Hung-yi Lee, National Taiwan University,
- Ian Goodfellow, Google Brain,
- Yann LeCun, New York University,
- Yoshua Bengio, Universite de Montreal
- Colah’s Blog, “Understanding LSTM Networks,” 2015
- I. Witten, et al., “Data Mining,” 2017
- Frederic Li, Adeel Nisar, Kimiaki Shirahama, University of Siegen

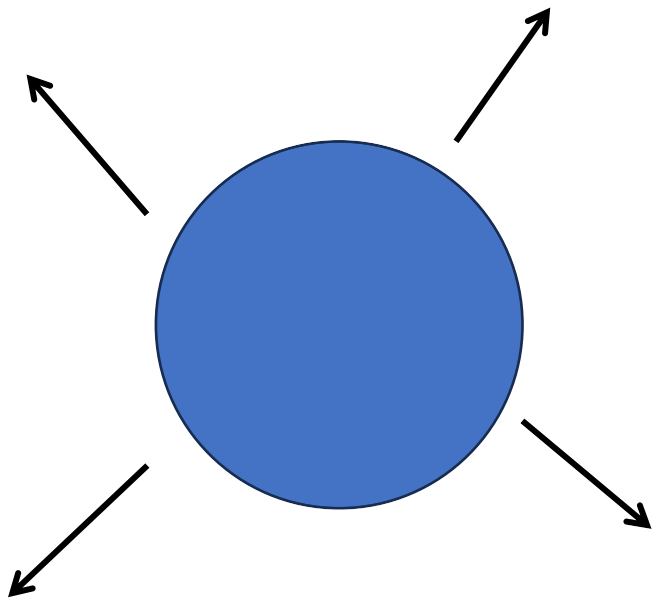
Human Brain and Sequential Data

- Human brain deals with information streams. Most data is obtained, processed, and generated sequentially.
  - E.g., listening: soundwaves → vocabularies/sentences
  - E.g., action: brain signals/instructions → sequential muscle movements
- Human thoughts have persistence; humans don't start their thinking from scratch every second.
  - As you read this sentence, you understand each word based on your prior knowledge.

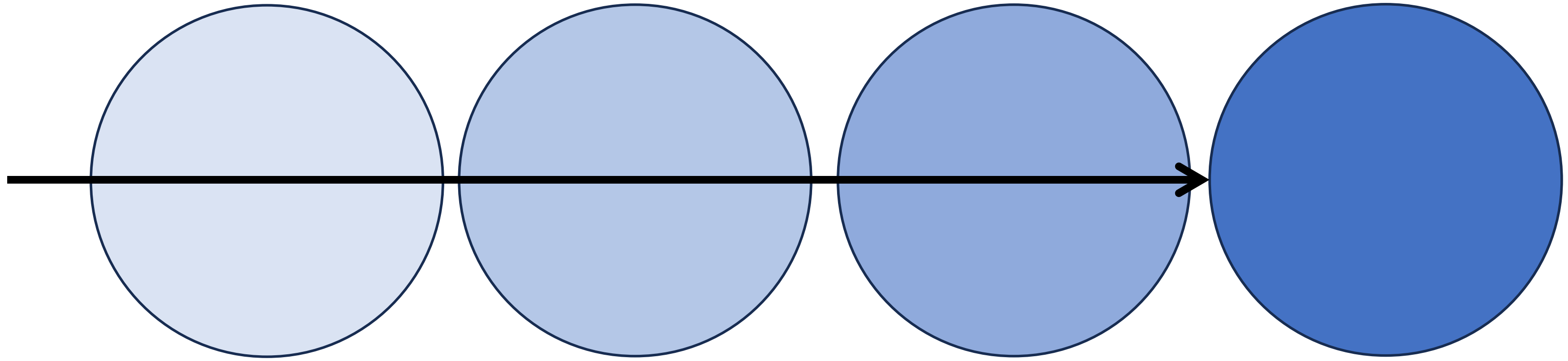
Sequence Prediction



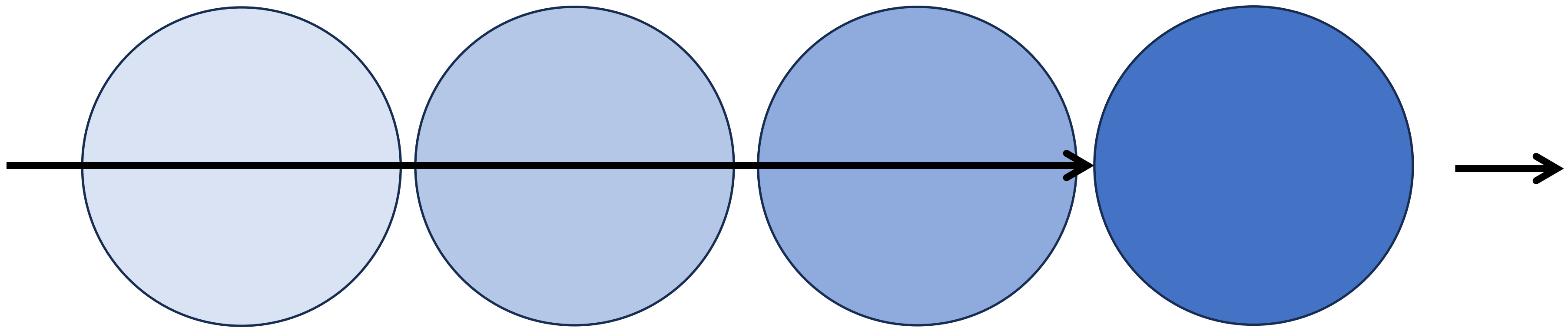
Sequence Prediction



Sequence Prediction



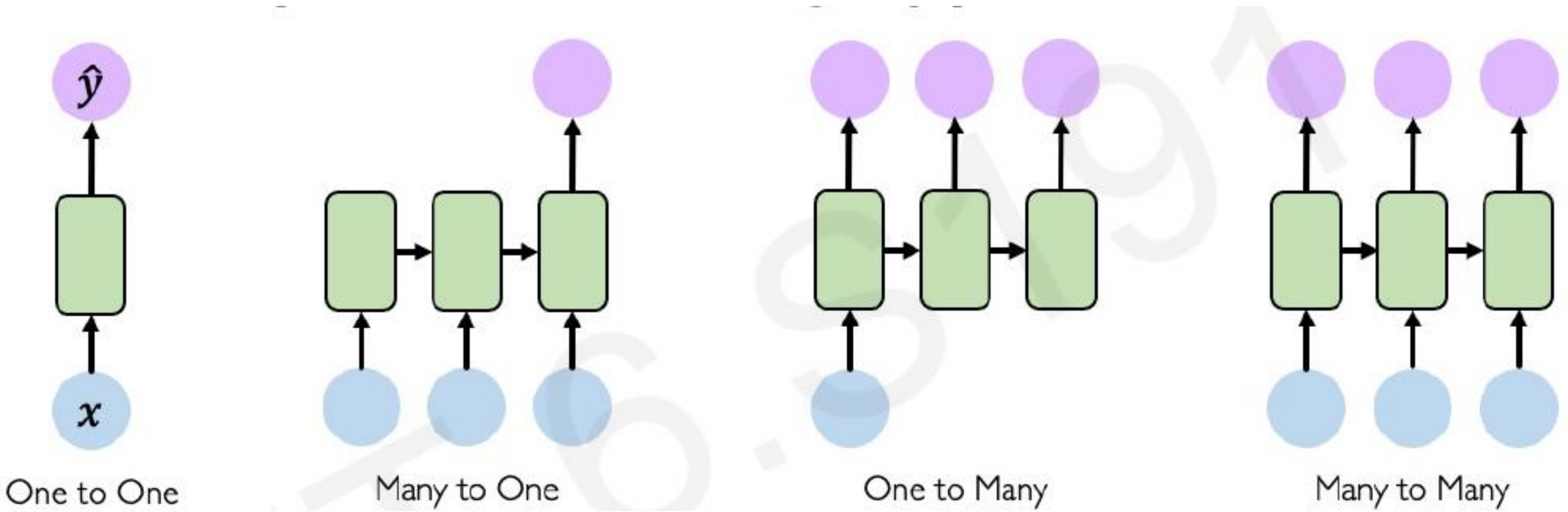
Sequence Prediction



Real-life Sequence Learning Applications

- Long-term Activity Recognition (e.g., Activities of Daily Living)
- Flow Detection
- Sleep Stage Classification
- Parkinson Patient’s Tremor Detection
- Pain Monitoring
- Speech Recognition
- Text Translation
- Stock Price Prediction

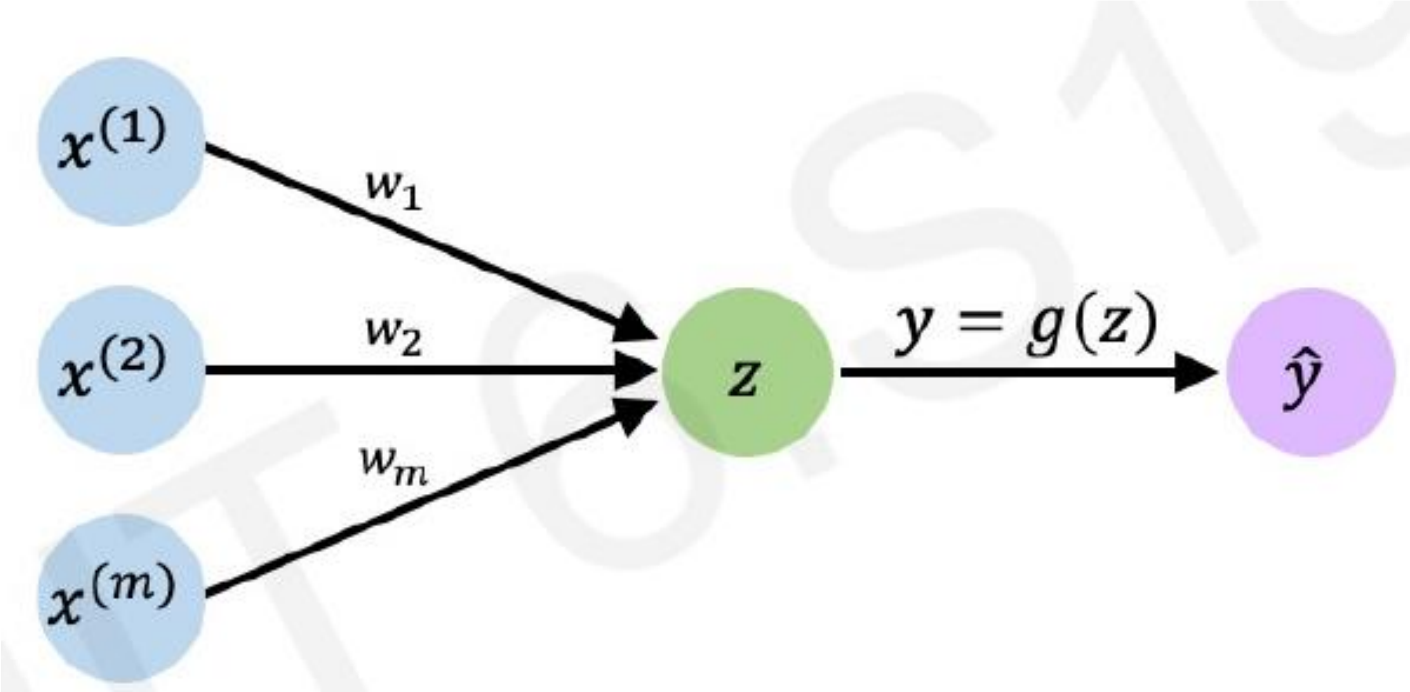
Sequence Modeling Applications



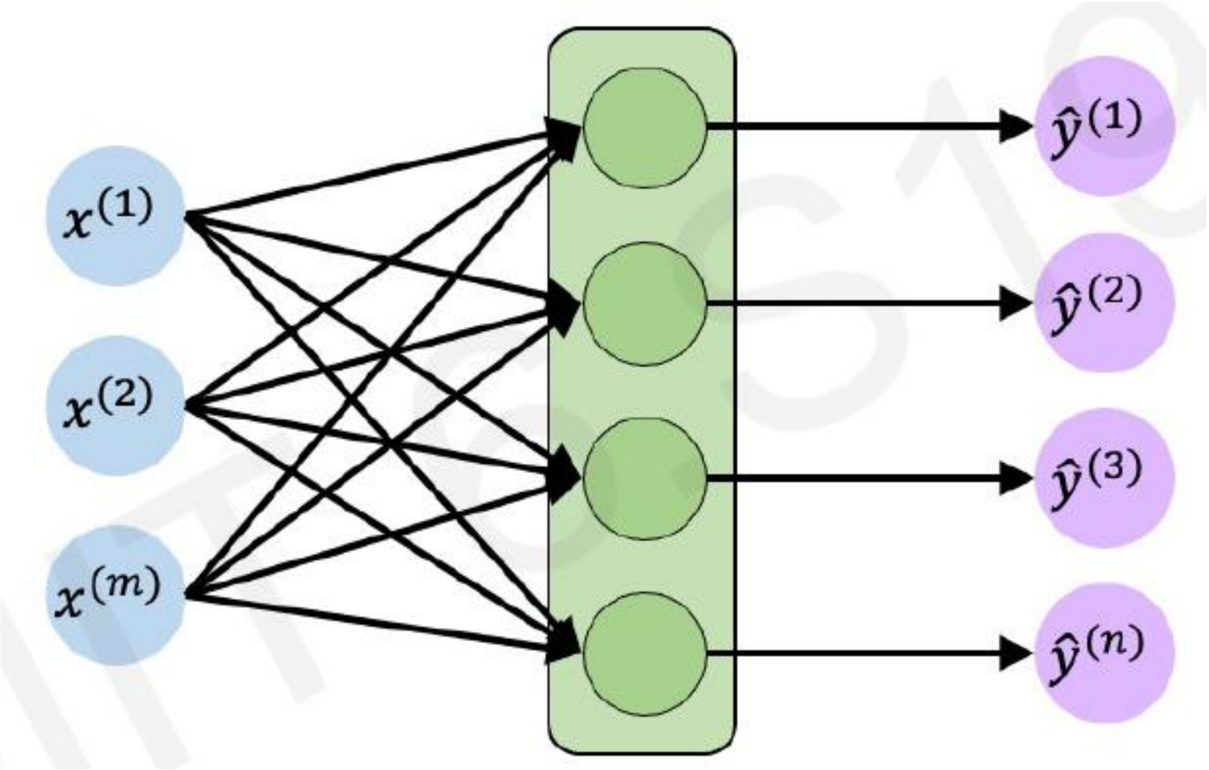
Recurrent Neural Network

- The applications of standard Artificial Neural Networks (and also Convolutional Networks) are limited due to:
  - They only accepted a fixed-size vector as input (e.g., an image) and produce a fixed-size vector as output (e.g., probabilities of different classes).
  - These models use a fixed amount of computational steps (e.g. the number of layers in the model).
- Recurrent Neural Networks (RNNs) are a family of neural networks introduced to **learn sequential data**.
  - Inspired by the temporal-dependent and persistent human thoughts

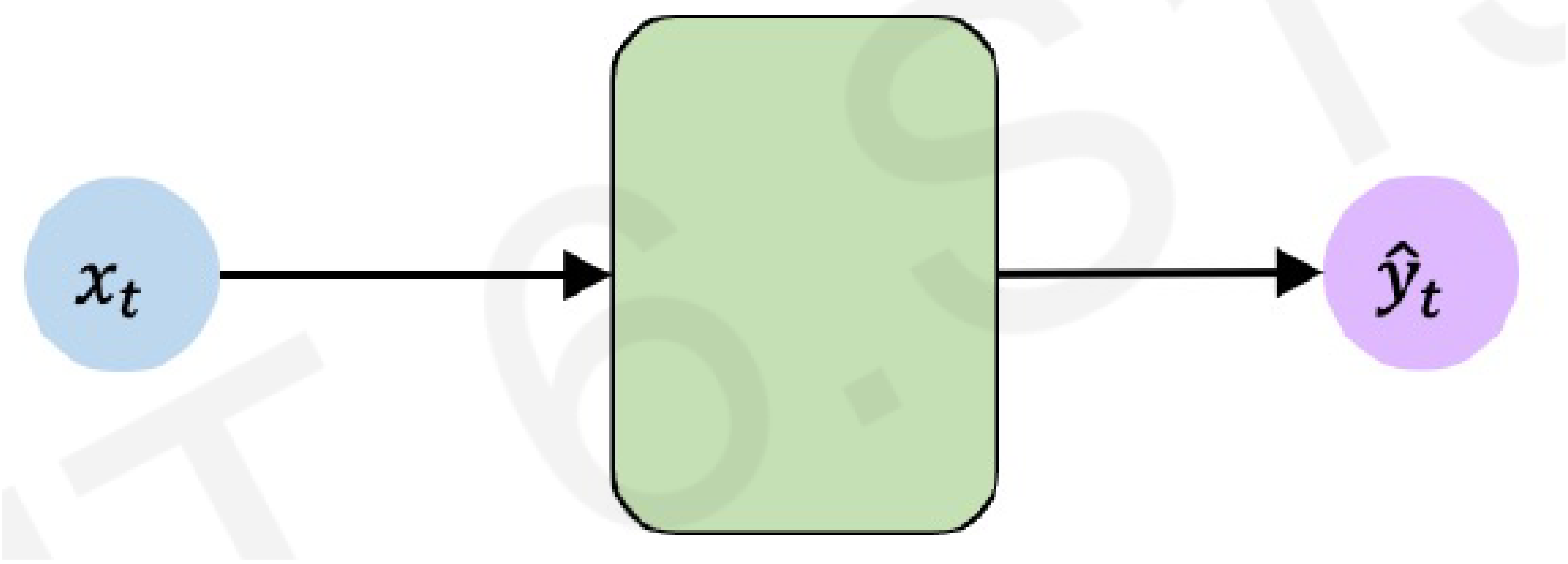
Perceptron Revisited



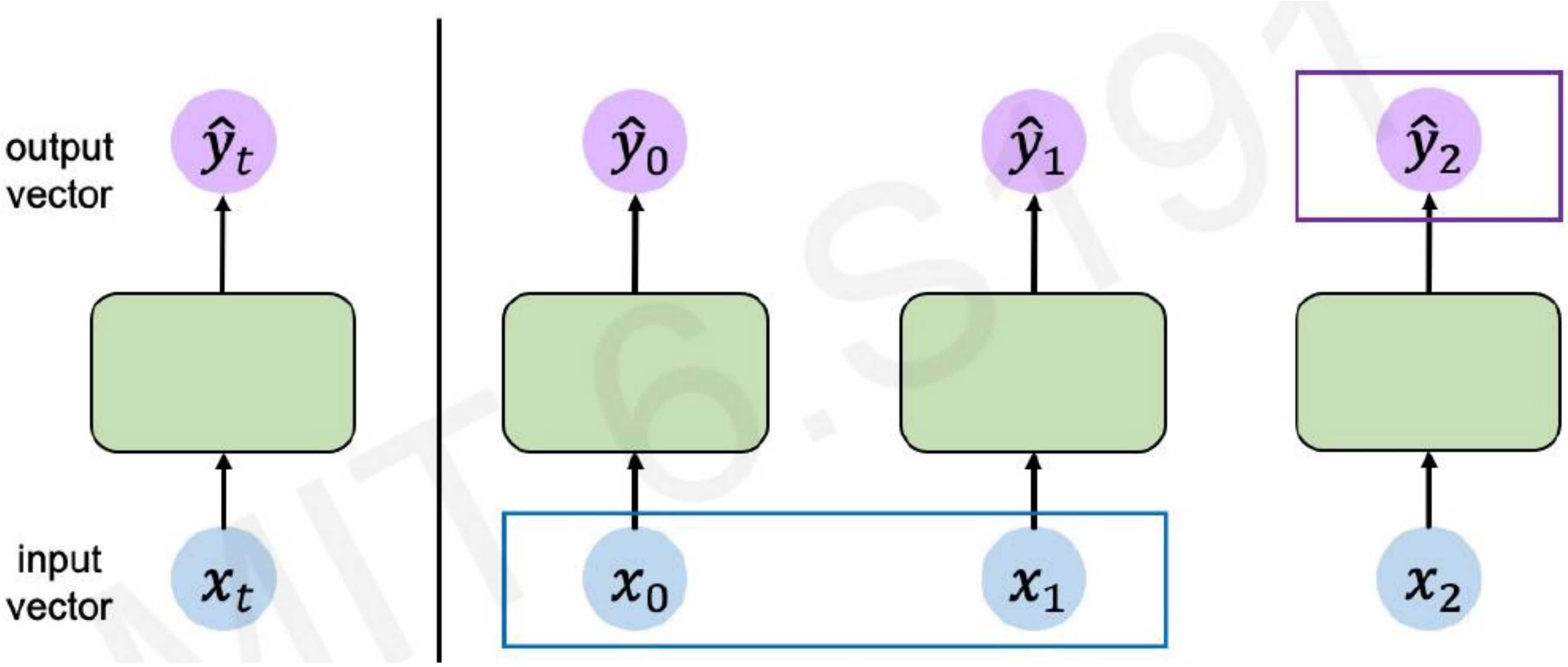
Feedforward Network Revisited



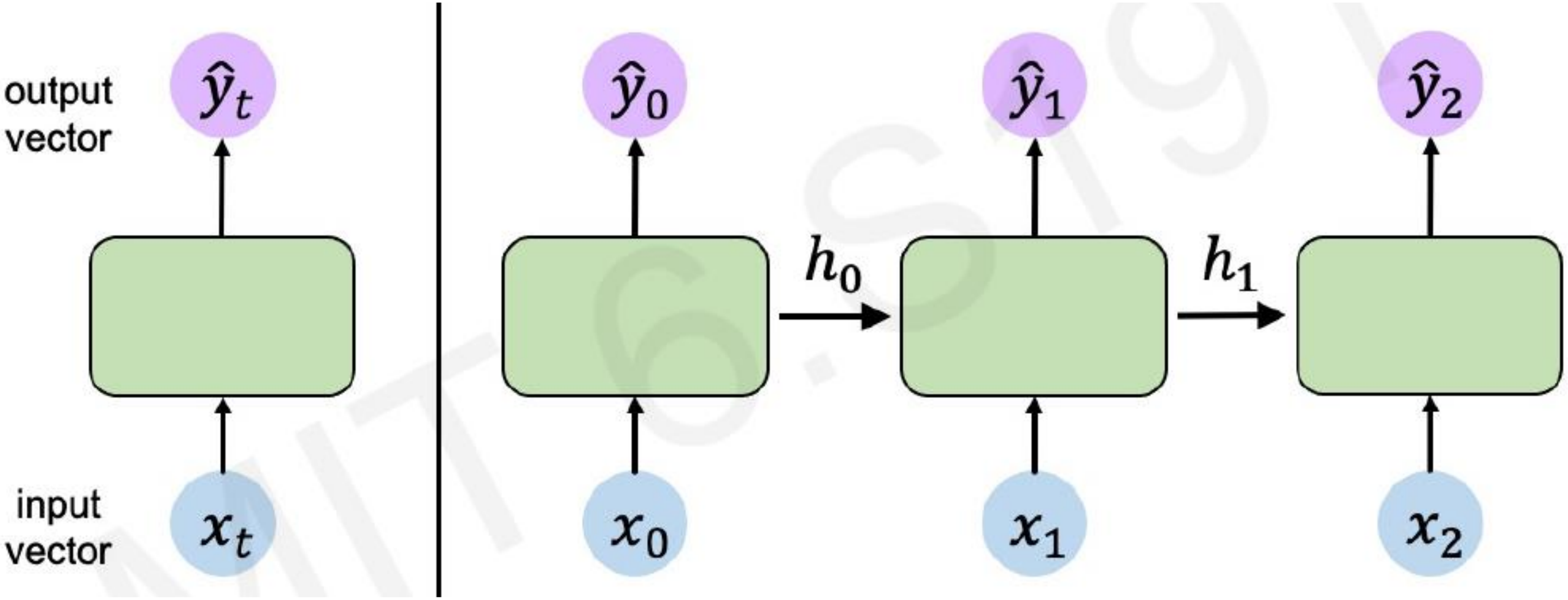
Feedforward Network Revisited



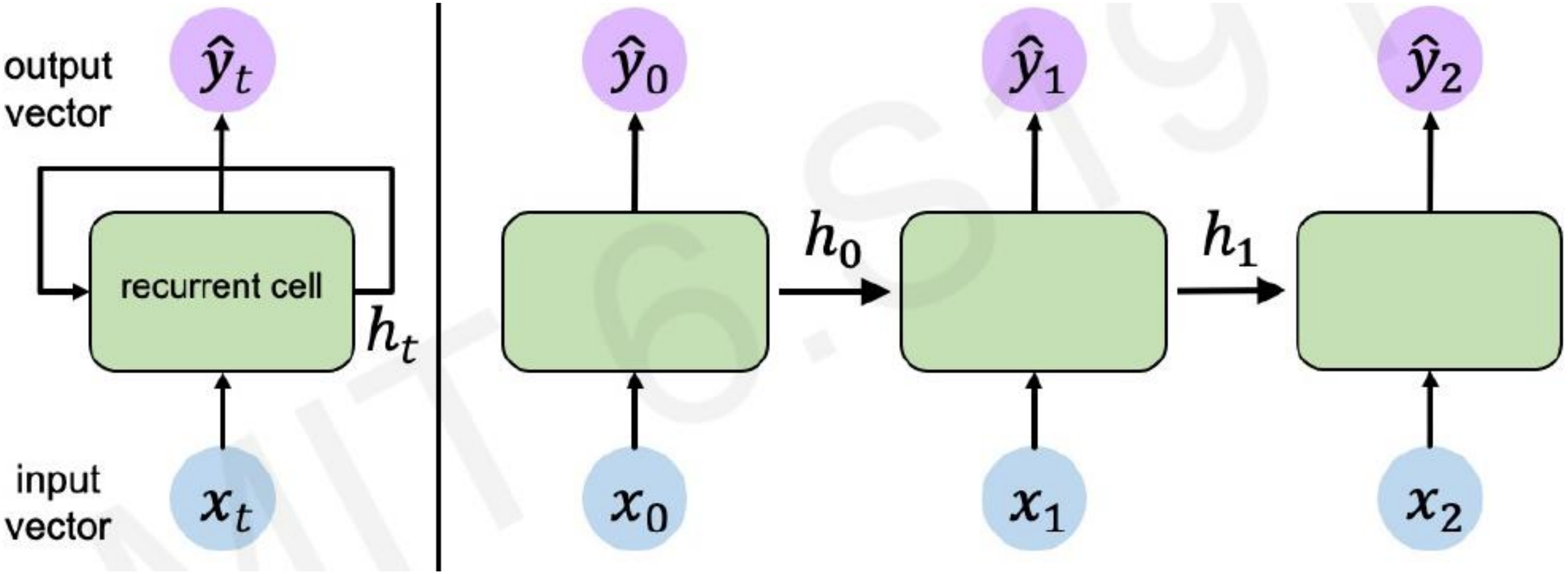
Multiple Feedforward Networks



Recurrent Neural Networks

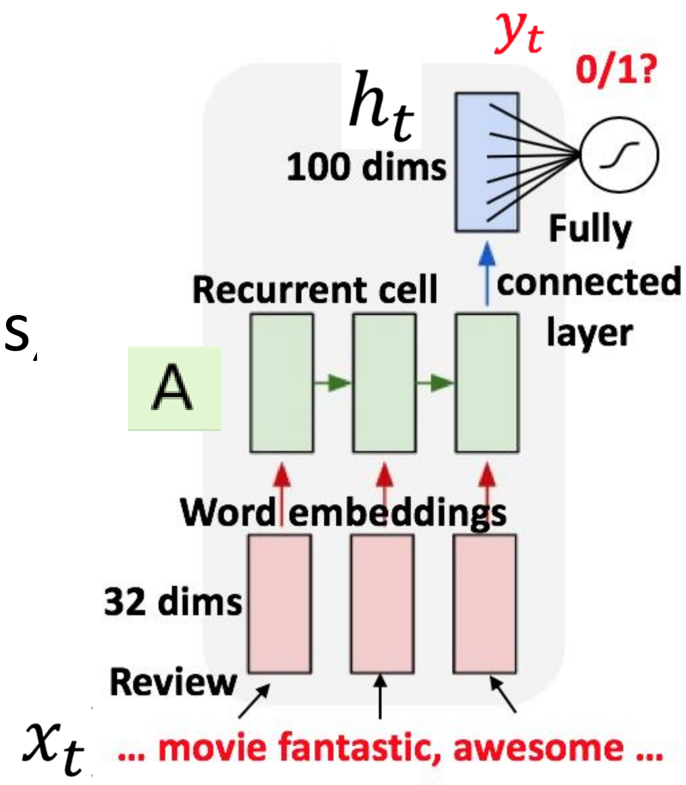


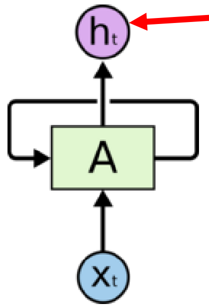
Recurrent Neural Networks



Recurrent Neural Networks

- Recurrent Neural Networks are networks with loops, allowing information to persist.





Output is to predict a vector h_t, where $output\ y_t = \varphi(h_t)$ at some time steps (t)

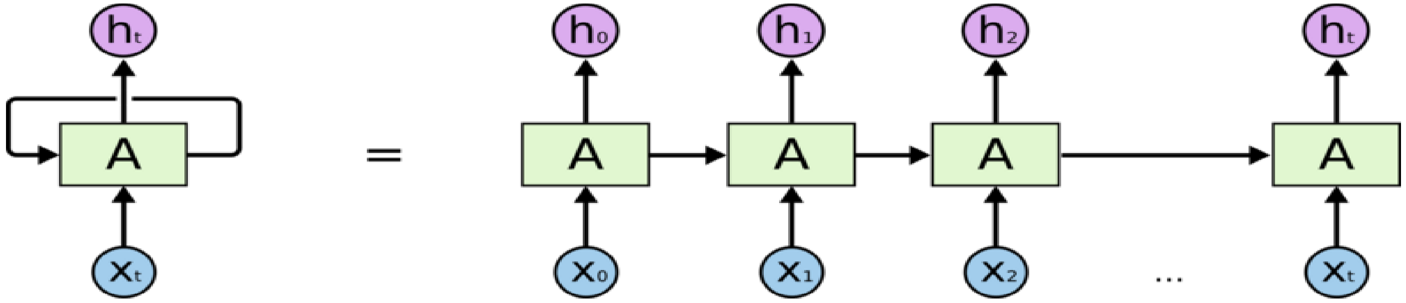
Recurrent Neural Networks have loops.

In the above diagram, a chunk of neural network, $A = f_W$, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next.

$$\overset{\text{new state}}{\boxed{h_t}} = \underset{\text{function with parameter } W}{\boxed{f_W}}\Big(\overset{\text{old state}}{\boxed{h_{t-1}}}, \underset{\text{Input vector at some time step}}{\boxed{x_t}}\Big)$$

Recurrent Neural Networks

- Unrolling RNN



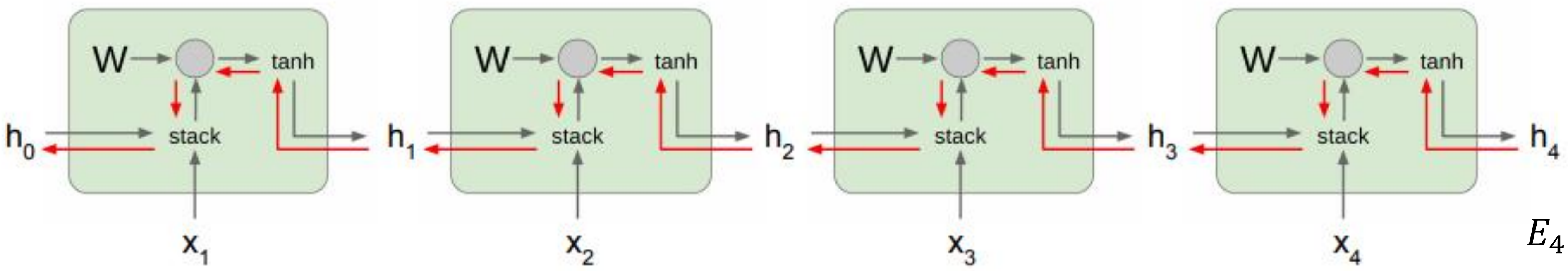
An unrolled recurrent neural network.

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. The diagram above shows what happens if we **unroll the loop**.

Recurrent Neural Networks

- The recurrent structure of RNNs enables the following characteristics:
  - Specialized for processing a sequence of values $x^{(1)}, \dots, x^{(\tau)}$
    - Each value $x^{(i)}$ is processed with the **same network *A* that preserves past information**
  - Can scale to much **longer sequences** than would be practical for networks without a recurrent structure
    - Reusing network ***A*** reduces the required amount of parameters in the network
  - Can process **variable-length sequences**
    - The network complexity does not vary when the input length change
  - However, vanilla RNNs suffer from the training difficulty due to **exploding and vanishing gradients**.

Exploding and Vanishing Gradients



$E_4 = J(y_4, \widehat{y_4})$

In vanilla RNNs, computing this gradient involves many factors of W_{hh} (and repeated $\tanh$)*. If we decompose the singular values of the gradient multiplication matrix,

- Largest singular value > 1 → **Exploding gradients**
  - Slight error in the late time steps causes drastic updates in the early time steps → Unstable learning
- Largest singular value < 1 → **Vanishing gradients**
  - Gradients passed to the early time steps is close to 0. → Uninformed correction

* Refer to Bengio et al. (1994) or Goodfellow et al. (2016) for a complete derivation

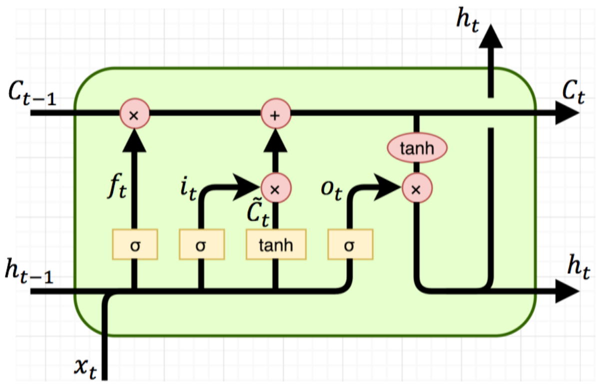
Networks with Memory

- Vanilla RNN operates in a "multiplicative" way (repeated tanh).

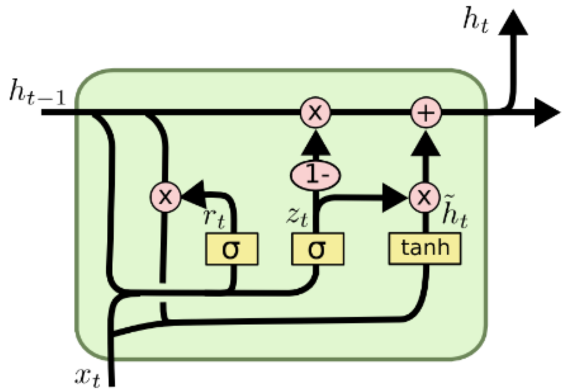
- Two recurrent cell designs were proposed and widely adopted:
    - **Long Short-Term Memory (LSTM)** (Hochreiter and Schmidhuber, 1997)
    - Gated Recurrent Unit (GRU) (Cho et al. 2014)

- Both designs process information in an "additive" way with gates to control information flow.
    - **Sigmoid gate outputs numbers between 0 and 1, describing how much of each component should be let through.**

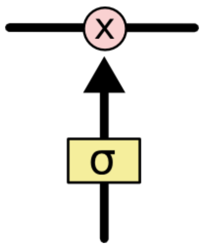
E.g. $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right) = Sigmoid\,(W_f x_t + U_t h_{t-1} + b_f)$



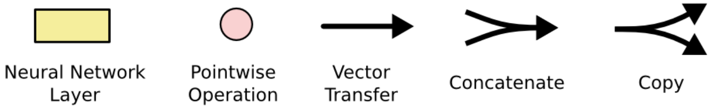
Standard LSTM Cell



GRU Cell



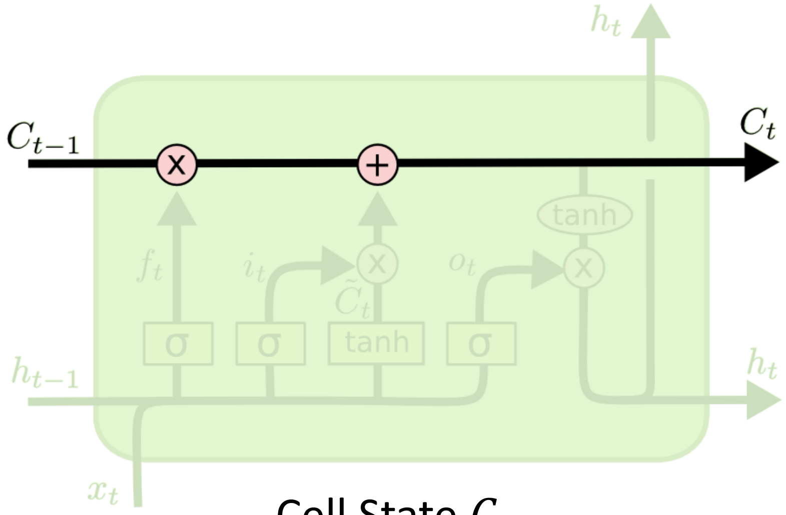
A Sigmoid Gate



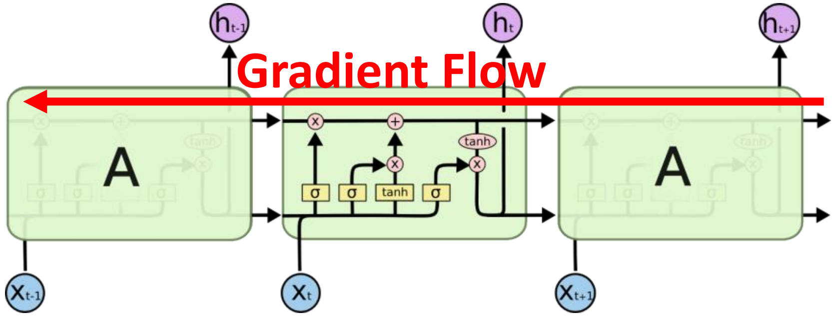
Long Short-Term Memory (LSTM)

- The key to LSTMs is the **cell state**.
  - **Stores information of the past → long-term memory**
  - **Passes along time steps with minor linear interactions → “additive”**
  - Results in an **uninterrupted gradient flow** → errors in the past pertain and impact learning in the future

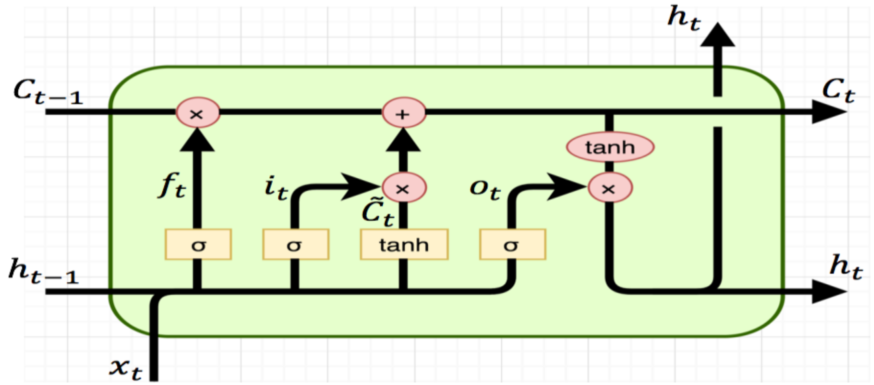
- The LSTM cell manipulates input information with three gates.
  - **Input gate** → controls the intake of new information
  - **Forget gate** → determines what part of the cell state to be updated
  - **Output gate** → determines what part of the cell state to output



Cell State C_t

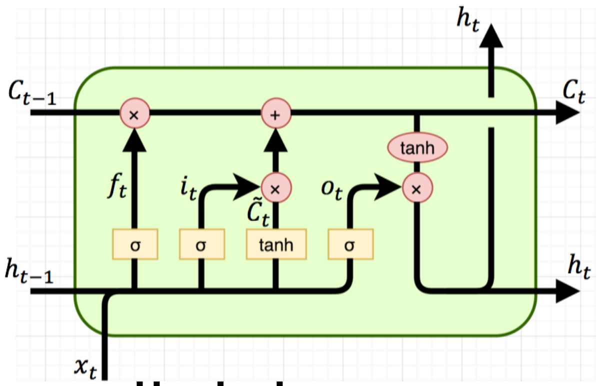


LSTM: Components & Flow

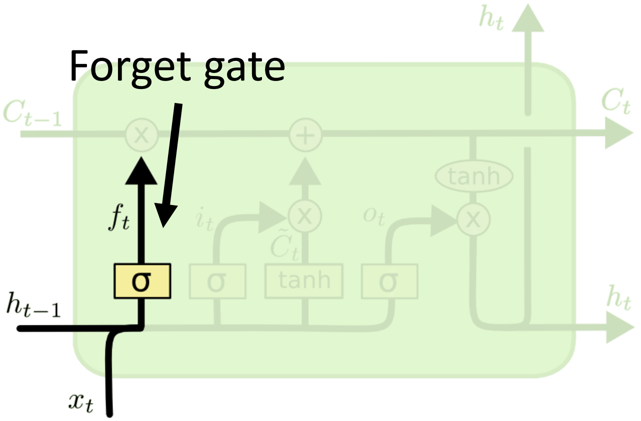


- LSM unit output $$h_t = o_t * \tanh\left(C_t\right)$$
- ***Output gate*** units $$o_t = \sigma\left(W_o\ \left[h_{t-1}, x_t\right]\ +\ b_o\right)$$
- Transformed memory cell contents $$\tanh\left(C_t\right)$$
- Gated update to memory cell units $$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
- ***Forget gate*** units $$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right]\ +\ b_f\right)$$
- ***Input gate*** units $$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right]\ +\ b_i\right)$$
- Potential *input* to memory cell $$\tilde{C}_t = \tanh(W_C \cdot \left[h_{t-1}, x_t\right]\ +\ b_C)$$

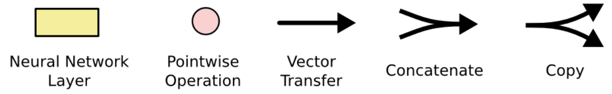
Step-by-step LSTM Walk Through



- **Step 1**: Decide what information to throw away from the cell state (memory) ➔ $f_t * C_{t-1}$
  - The output of the previous state h_{t-1} and the new information x_t jointly determine what to forget
    - h_{t-1} contains selected features from the memory C_{t-1}
  - Forget gate f_t ranges between $[0, 1]$

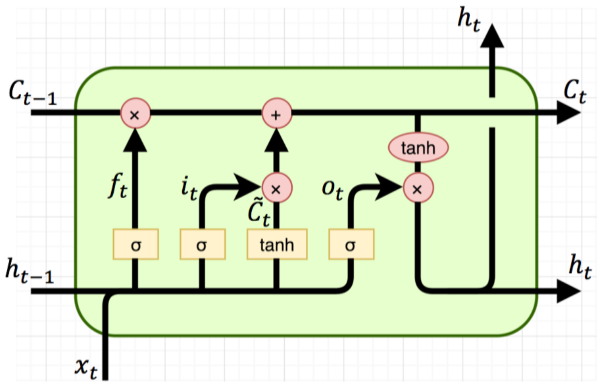


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$



Text processing example:
Cell state may include the <u>gender of the current subject</u> (h_{t-1}). When the model observes a <u>new subject</u> (x_t), it may <u>want to forget</u> ($f_t \rightarrow 0$) the old subject in the <u>memory</u> (C_{t-1}).

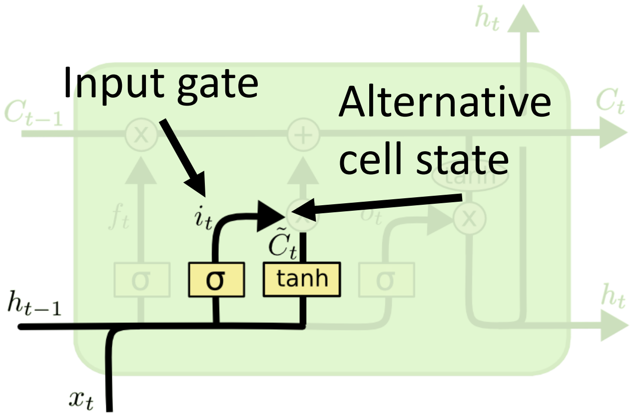
Step-by-step LSTM Walk Through



- **Step 2**: Prepare the updates for the cell state

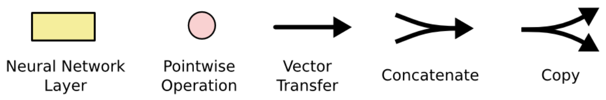
  from input ➔ $i_t * \tilde{C}_t$

  - An alternative cell state $\widetilde{C}_t$ is created from the new information x_t with the guidance of h_{t-1}.
  - Input gate i_t ranges between $[0, 1]$



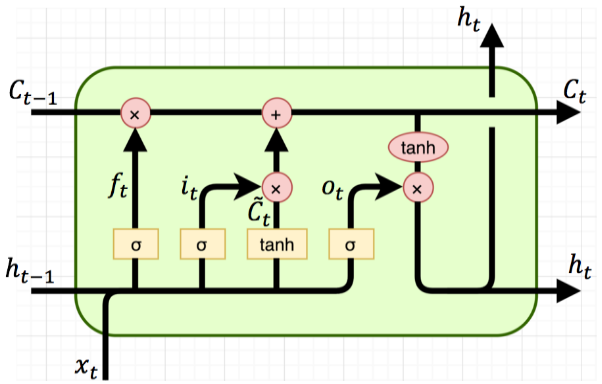
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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



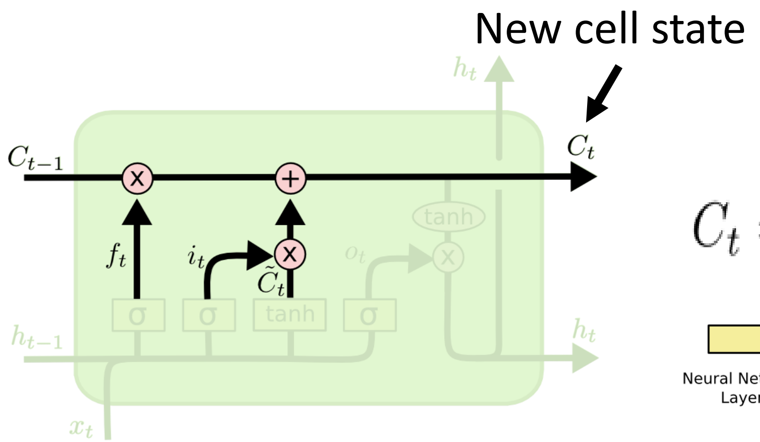
Example:
The model may <u>want to add</u> ($i_t \to 1$) the <u>gender of new subject</u> ($\widetilde{C}_t$) to the cell state to replace the old one it is forgetting.

Step-by-step LSTM Walk Through

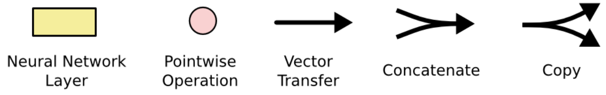


- **Step 3**: Update the cell state ➔ $f_t * C_{t-1} + i_t * \tilde{C}_t$
  - The new cell state C_t is comprised of information from the past $f_t * C_{t-1}$ and valuable new information $i_t * \tilde{C}_t$

  - $*$ denotes elementwise multiplication

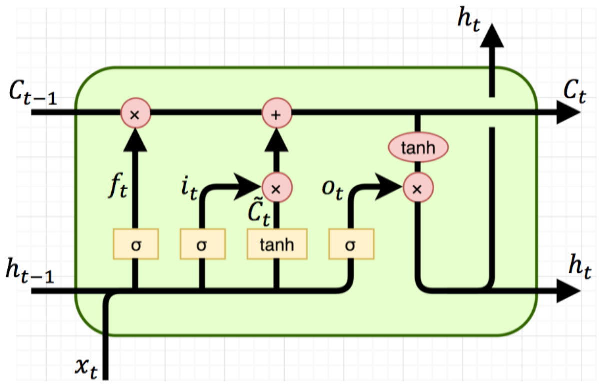


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

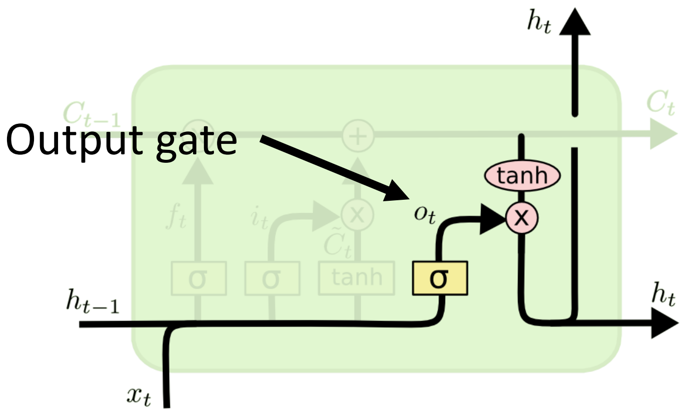


Example:
The model <u>drops the old gender information</u> ($f_t * C_{t-1}$) and <u>adds new gender information</u> ($i_t * \tilde{C}_t$) to form the <u>new cell state</u> (C_t).

Step-by-step LSTM Walk Through

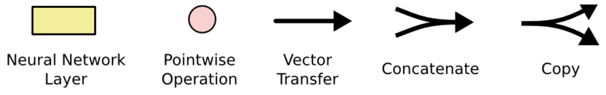


- **Step 4**: Decide the filtered output from the new cell state ➔ $o_t * \tanh(C_t)$
  - tanh function filters the new cell state to characterize stored information
    - Significant information in C_t → ± 1
    - Minor details → 0
  - Output gate o_t ranges between $[0, 1]$
  - h_t serves as a control signal for the next time step



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

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

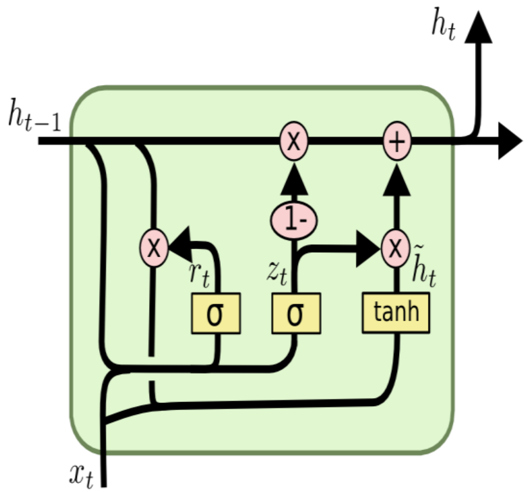


Example:
Since the model just saw a new subject (x_t), it might <u>want to output</u> ($o_t \rightarrow 1$) information relevant to a verb ($\tanh(C_t)$), e.g., singular/plural, in case a verb comes next.

Gated Recurrent Unit (GRU)

- GRU is a variation of LSTM that also adopts the gated design.
- Differences:
  - GRU uses an **update gate** $\boldsymbol{z}$ to substitute the input and forget gates i_t and f_t
  - Combined the cell state C_t and hidden state h_t in LSTM as a single cell state h_t

- GRU obtains similar performance compared to LSTM with fewer parameters and faster convergence. (Cho et al. 2014)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

Update gate: controls the composition of the new state

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Reset gate: determines how much old information is needed in the alternative state $\widetilde{h_t}$

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

Alternative state: contains new information

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

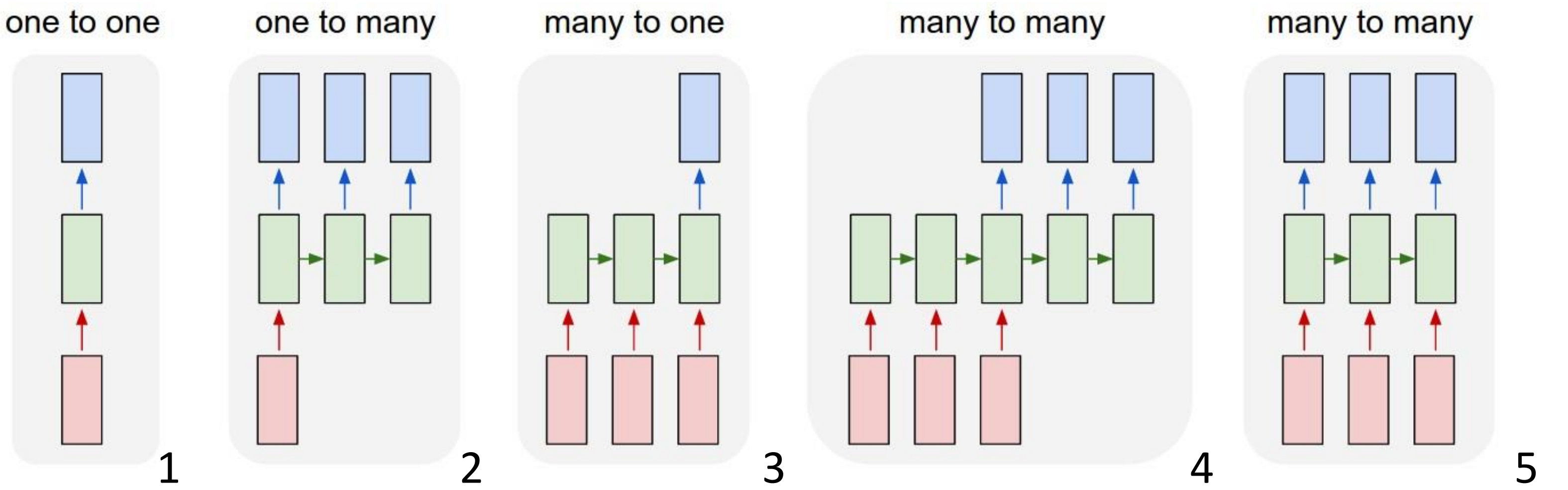
New state: replace selected old information with new information in the new state

Sequence Learning Architectures

- Learning on RNN is more robust when the vanishing/exploding gradient problem is resolved.
  - RNNs can now be applied to different Sequence Learning tasks.

- Recurrent NN architecture is flexible to operate over various sequences of vectors.
  - Sequence in the input, the output, or in the most general case both
  - Architecture with one or more RNN layers

Sequence Learning with One RNN Layer



- Each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

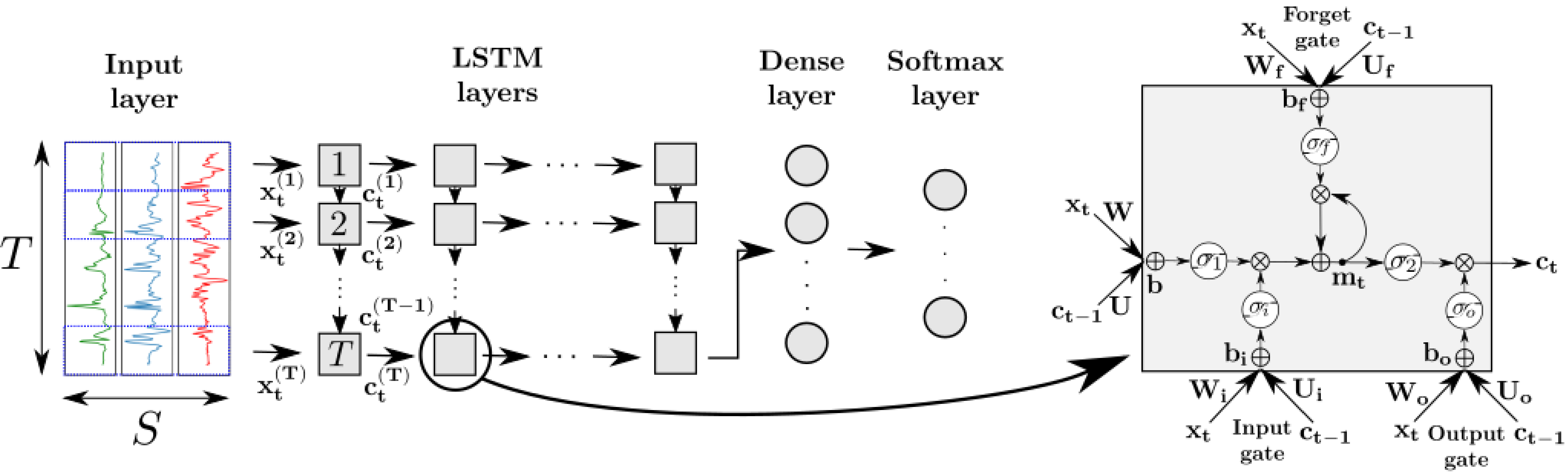
(1) Standard NN mode without recurrent structure (e.g. **image classification, one label for one image**).
(2) Sequence output (e.g. **image captioning, takes an image and outputs a sentence of words**).
(3) Sequence input (e.g. **sentiment analysis, a sentence is classified as expressing positive or negative sentiment**).
(4) Sequence input and sequence output (e.g. **machine translation, a sentence in English is translated into a sentence in French**).
(5) Synced sequence input and output (e.g. **video classification, label each frame of the video**).

Applications of RNN in Assistive Health Technologies

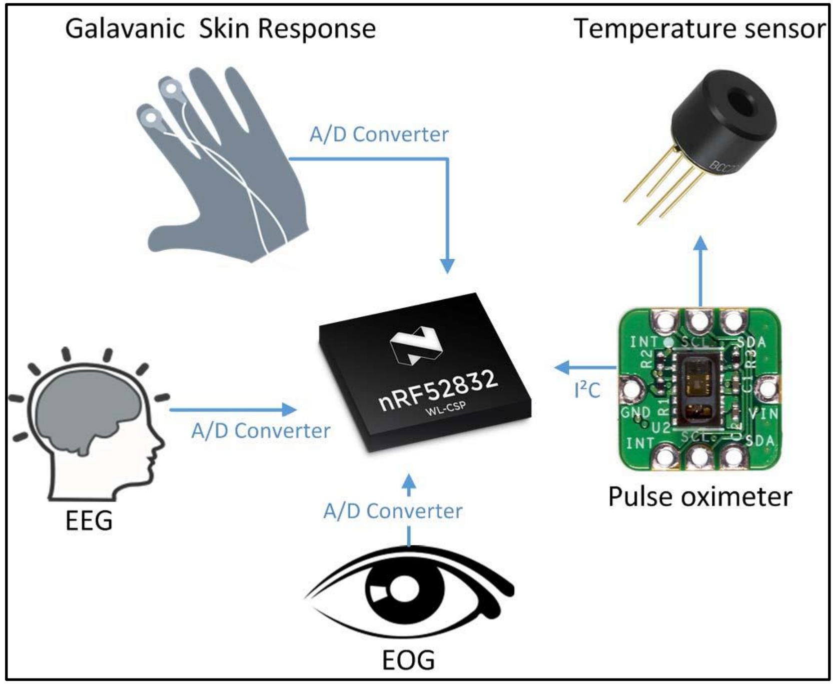
By using timeseries sensor data for the following domains:

- Emotion Recognition
- Pain Recognition
- Activity Recognition
- Sleep Stage Classification
- Parkinson Patient’s Tremor Detection

Applications of RNN in Assistive Health Technologies



Sensor-Based Emotion Recognition



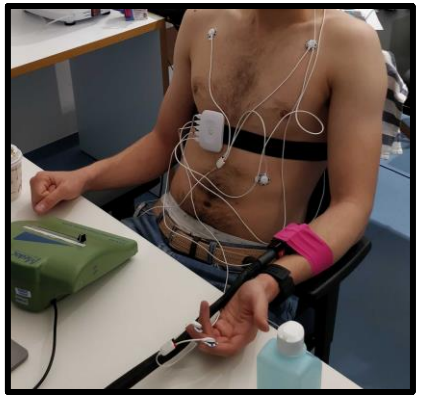
AF1 (%)	Subject-dependent	Subject-independent
HCF	**91.49**	**28.85**
MLP	32.60	26.74
CNN	34.95	15.99
LSTM	35.27	27.83

Sensor headband (BVP, PPG, GSR, EEG, EOG)

Happiness, frustration, boredom, other (4 classes)

A. Grünewald, **F. Li**, H. Kampling, D. Krönert et al., *Biomedical Data Acquisition and Processing to Recognize Emotions for Affective Learning*, Proc. of IEEE BIBE, 2018

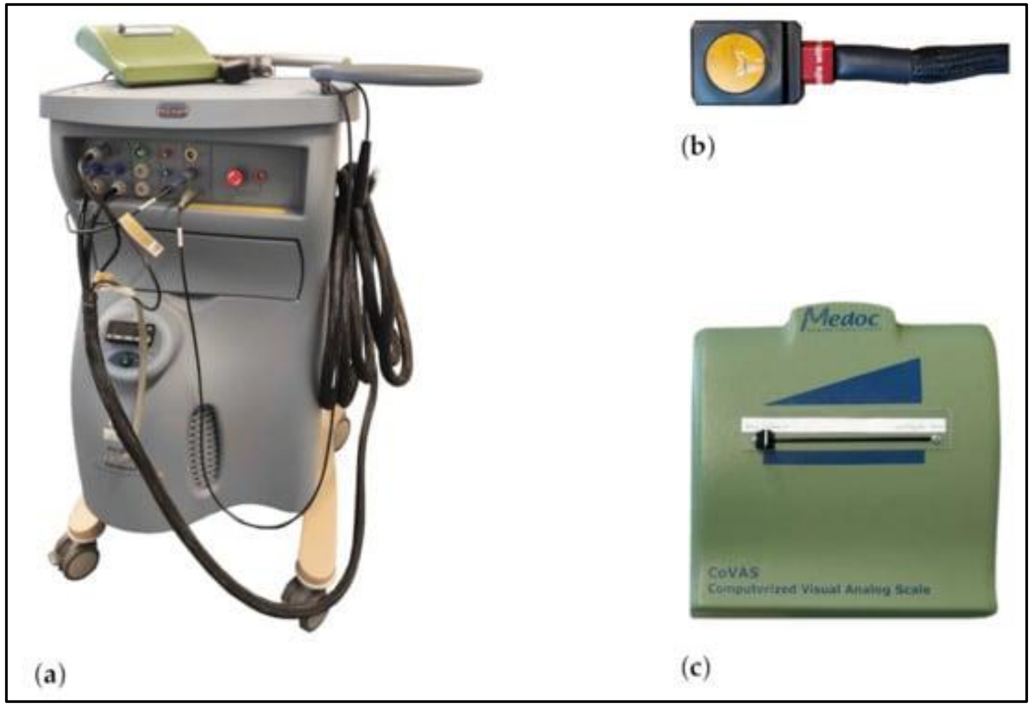
Sensor-Based Pain Monitoring



- Smartwatch (GSR, T°, BVP)
- Multisensor platform (GSR, EMG, ECG, Respiration)

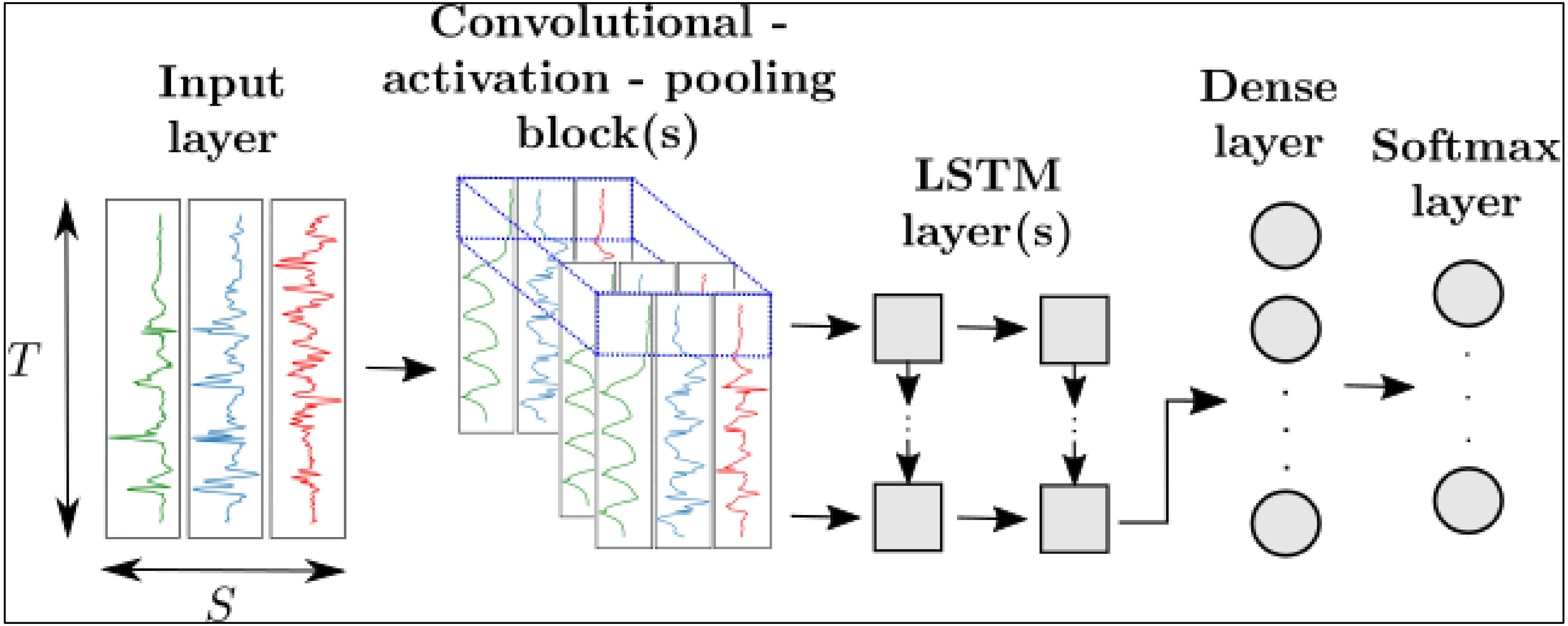
Pain, no pain (2 classes)

DL Approach	Pain Recognition	
	Acc (%)	AF1 (%)
MLP	84.01	83.58
LSTM	83.48	83.12



P. J. Gouverneur, **F. Li,** T. M. Szikszay, W. M. Adamczyk, K. Luedtke, M. Grzegorzek, *Classification of Heat-induced Pain using Physiological Signals*, Proc. of ITIB 2020

Sensor-Based Activity Recognition



<u>F. Li</u>, K. Shirahama, M. A. Nisar, L. Köping, M. Grzegorzek, *Comparison of Feature Learning Methods for Human Activity Recognition using Wearable Sensors*, Sensors (MDPI), 2018

Results

OPPORTUNITY dataset [1]

- 7 IMUs (3D acceleration, angular velocity, magnetometer) and 12 3D accelerometers placed all over the body
- 17 activities of daily life and one NULL class (18 classes)

Method	Accuracy	Weighted F1-Score	Average F1-Score
HCF	89.96	89.53	63.76
CBH	89.66	88.99	62.27
CBS	90.22	89.88	67.50
MLP	91.11	90.86	68.17
CNN	90.58	90.19	65.26
LSTM	91.29	91.16	69.71
Hybrid	**91.76**	**91.56**	**70.86**
AE	87.80	87.60	55.62

[1] R. Chavarriaga et al., *The Opportunity Challenge: a Benchmark Database for On-body Sensor-based Activity Recognition*, Pattern Recognition Letters (Elsevier), 2013

UniMIB SHAR dataset [2]

- Smartphone (3D acceleration)
- 9 activities of daily life and 7 falling movements (16 classes)

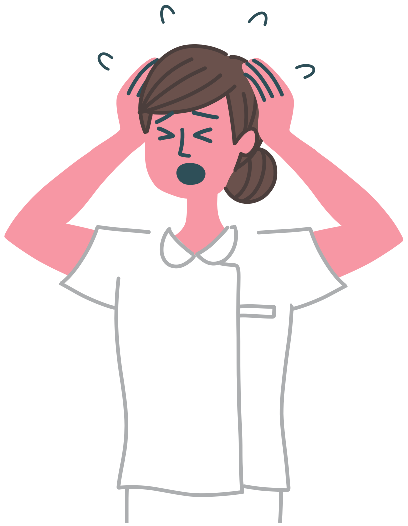
Method	Accuracy	Weighted F1-score	Average F1-score
Baseline [2]	54.70	—	—
HCF	32.01	22.85	13.78
CBH	75.21	74.13	60.01
CBS	**77.03**	**75.93**	63.23
MLP	71.62	70.81	59.97
CNN	74.97	74.29	**64.65**
LSTM	71.47	70.82	59.32
Hybrid	74.63	73.65	64.47
AE	65.67	64.84	55.04

[2] D. Micucci et al., *UniMIB SHAR: a New Dataset for Human Activity Recognition using Acceleration Data from Smartphone*, Applied Sciences (MDPI), 2016

Aging Society and Health Assessment

- Elderly population is expected to double by 2050[1]
- Activities of Daily Living (ADLs) as a health assessment tool[2]

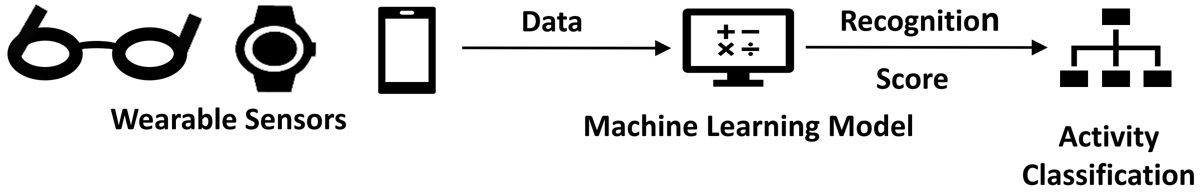
- Continuous monitoring of ADLs



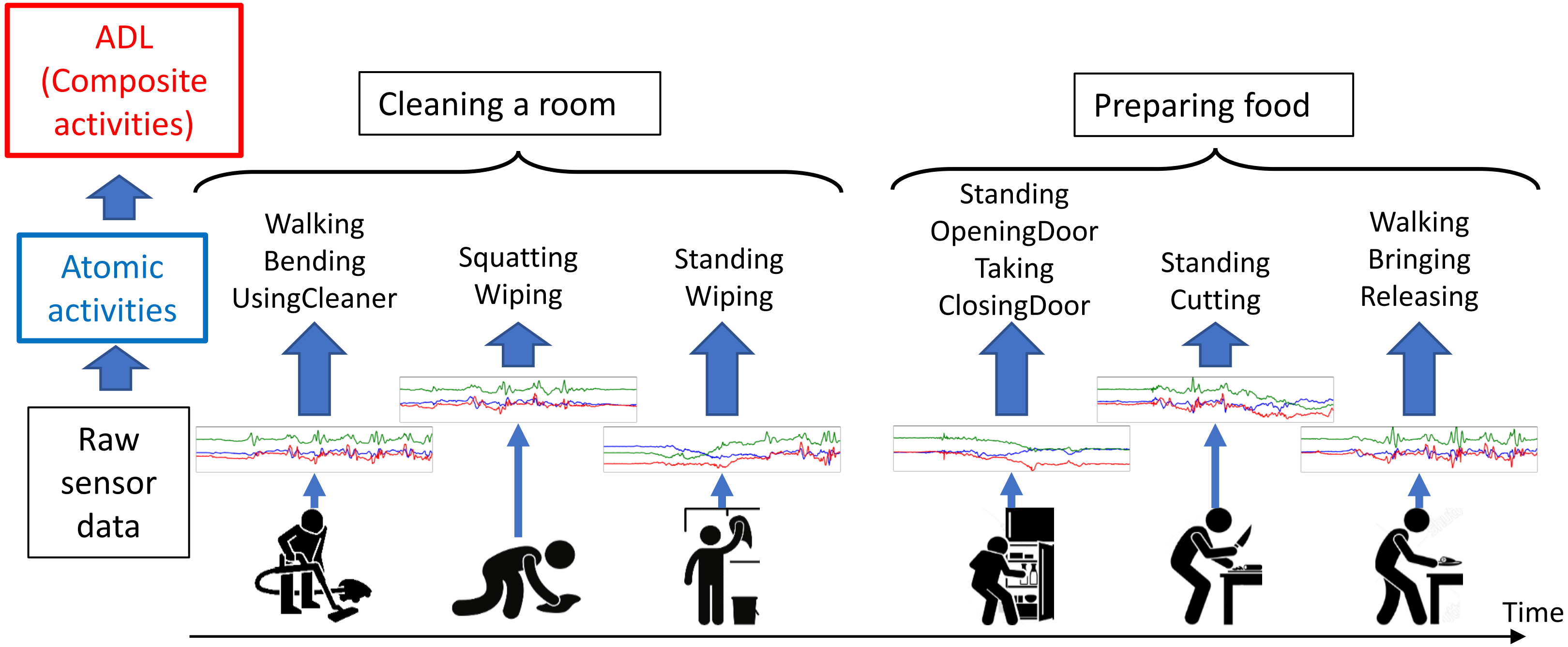
[1] https://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2017_Highlights.pdf
[2] Graf, Carla L. "The Lawton Instrumental Activities of Daily Living Scale." *AJN, American Journal of Nursing* 108 (2008): 52–62.

Human Activity Recognition (HAR) System

- **Sensor-based HAR**



- **Acquisition of CogAge Datasets[1]**

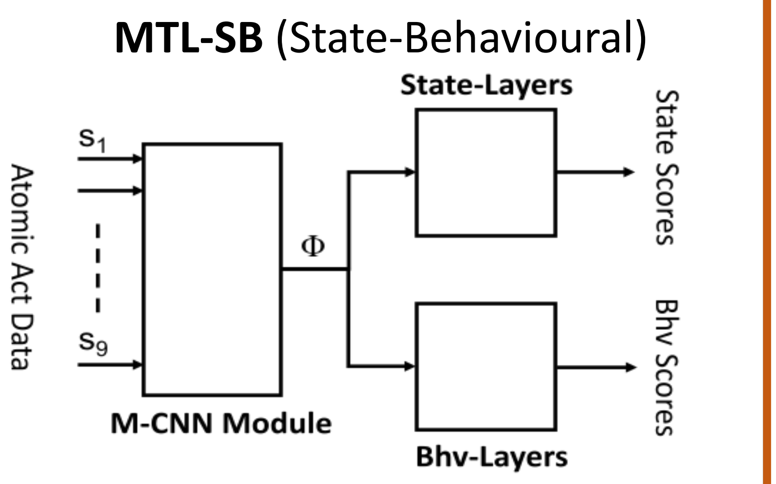


[1] Cognitive Village, Bundesministerium für Bildung und Forschung (BMBF) Project, FKZ 16SV7311

Hierarchical Multitask Learning

- Objective
  - Preserve the temporal evolution
  - Augment the dataset by learning all three types of activities
- Issues with existing approaches
  - Built for the activities with same temporal scales
- Proposed approach
  - Multitask model for different temporal scales
  - Time-distributed CNN-LSTM modules
  - Compared to baseline with single task learning (STL) models

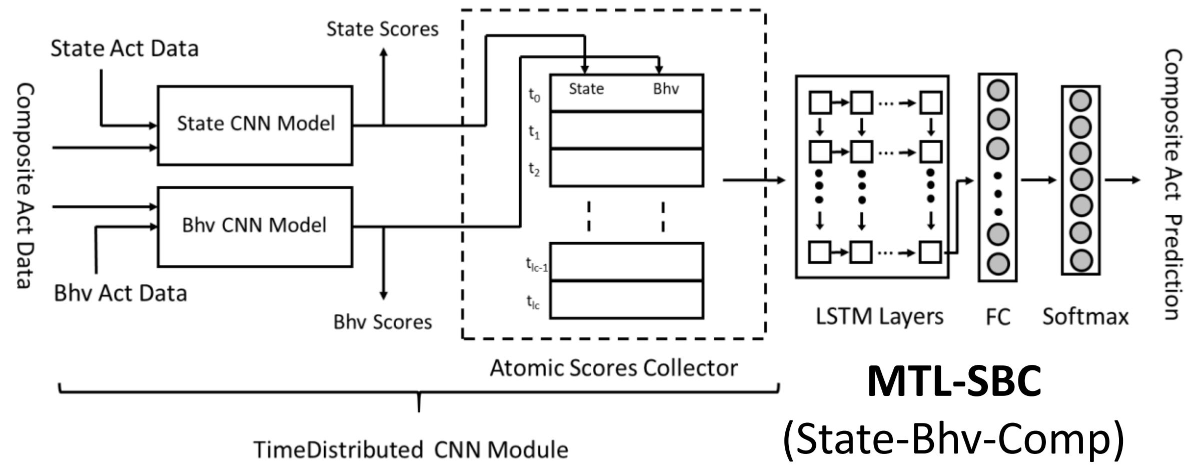
MTL Models



$$L(f_i) = -\frac{1}{U}\sum_{u=1}^{U} \boldsymbol{y}_i^{(u)^T} \log(\boldsymbol{q}_i^{(u)})$$

Where $i \in \{S, B, C\}$

$$L(f_{\mathrm{MTL-SB}}) = w_S \,.\, L(f_S) + w_B \,.\, L(f_B)$$



$$L(f_{\mathrm{MTL-SBC}}) = w_S \,.\, L(f_S) + w_B \,.\, L(f_B) + w_C \,.\, L(f_C)$$

MTL Models – Results and Issues

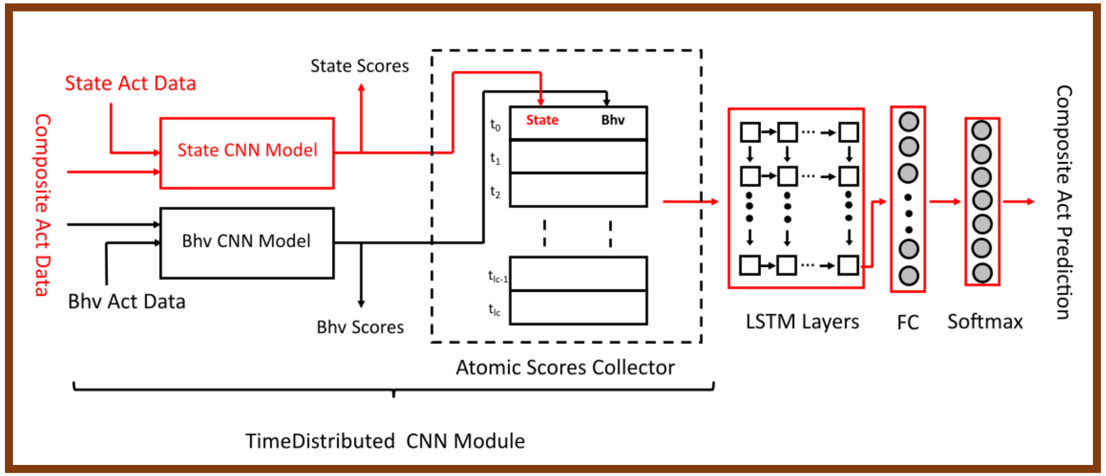
Models		State	Behavioural	Composite
Atomic	**Composite**	**Accuracy (%)**		
M-CNN	RP+MP+AP	92.4	71.8	88.5
MTL-SB		77.4	72.0	
MTL-SBC		74.4	72.2	**92.9**

Confusion Matrix – State Activities – MTL-SB						
Activities	Bending	Lying	Sitting	Squatting	Standing	Walking
Bending	**69**	0	0	0	0	0
Lying	6	**63**	0	0	0	0
Sitting	**43**	2	**19**	3	0	0
Squatting	17	0	2	**48**	0	0
Standing	**43**	0	0	2	**27**	0
Walking	5	0	0	0	0	**65**

MTL-StateComposite-BehaviouralComposite

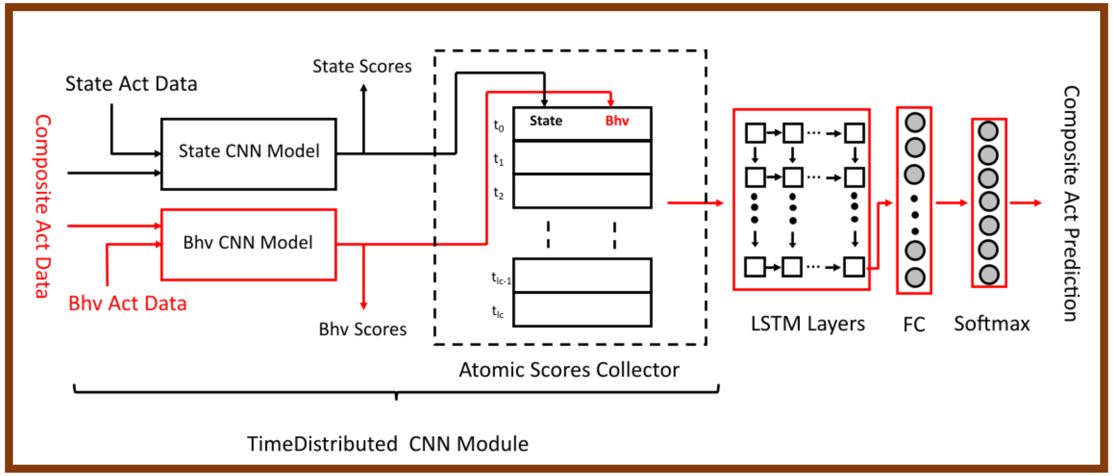
MTL-SC-BC

$$L(f_{SC}) = w_S \,.\, L(f_S) + w_C \,.\, L(f_C)$$



MTL-SC-BC

$$L(f_{BC}) = w_B \,.\, L(f_B) + w_C \,.\, L(f_C)$$



Final Results – All Methods

Recognition of State, Behavioural and Composite Activities							
Method		State		Behavioural		Composite	
Atomic	**Comp.**	**AF1**	**Acc**	**AF1**	**Acc**	**AF1**	**Acc**
CB	RP+MP+AP	88.2	88.6	67.9	68.2	88.0	88.5
M-CNN	RP+MP+AP	92.3	92.4	71.7	71.8	87.4	87.9
MTL-SBC		73.3	74.4	71.7	72.2	92.3	92.9
MTL-SC-BC		**95.1**	**95.2**	**73.4**	**73.9**	**93.8**	**94.0**

More Application of Assistive Health Technologies

- Emotion Recognition
- Anxiety and Stress Detection
- Dataset
  - Sensors
    - Audio
    - Visual
  - Data
    - RAVDESS
      - https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio
    - TESS
      - https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess
    - Stress Detection
      - https://figshare.com/articles/dataset/Anxiety_Dataset_2022/19875217