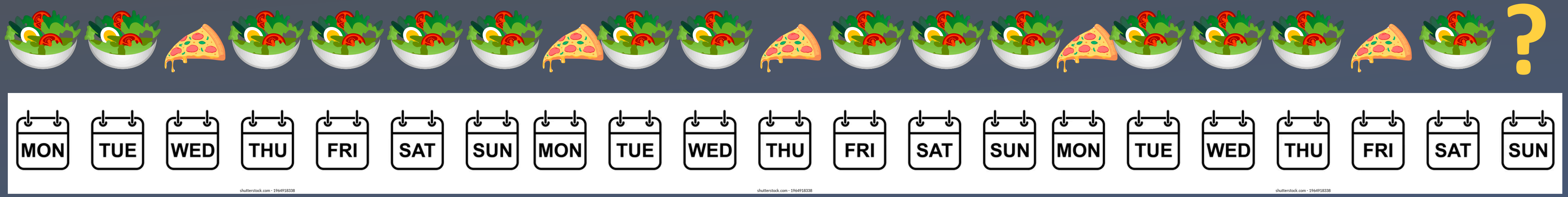
Lecture 9: Recurrent Neural Networks (RNNs)

COMPSCI/DATA 182: Deep Learning

09/26/2024

In **Sequences**, we Trust



- *What is the entree for today ?*
  - What do I factor from history ?
    - How much history ?
    - What tidbits do I need to retain ?
    - What tidbits can I forget about ?
    - Can knowledge of the future help me, as well ?

Latent Autoregressive Models

$$P(x_1, x_2, \ldots, x_T),$$

$$\boxed{P(x_1, \ldots, x_T) = P(x_1) \prod_{t=2}^{T} P(x_t \mid x_{t-1}).}$$

$$P(x_1, \ldots, x_T) = P(x_1) \prod_{t=2}^{T} P(x_t \mid x_{t-1}, \ldots, x_1).$$

- Leveraging sequences

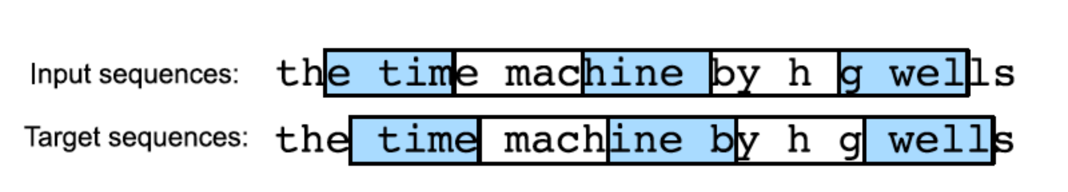
TEXT: The Goldmine

- Text is among the most common forms of sequence data encountered in deep learning
  - Common choices for token: characters, words, and word pieces.
- Language models estimate the joint probability of a text sequence
  - For long sequences, ngrams provide a convenient model by truncating the dependence
- Language models can be scaled up with increased data size, model size, and amount in training compute
- PERPLEXITY

$$\text{Perplexity} = \exp(\text{Cross-Entropy Loss})$$

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_{1:i-1})\right)$$

- Large language models can perform desired tasks by predicting output text given input text instructions.



Sequences, in Deep Learning

$$\mathbf{H} = \phi(\mathbf{X}\mathbf{W}_{xh} + \mathbf{b}_h).$$

$$\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q,$$

Hidden layer

H = Hidden layer output
X = Input
W = Hidden layer parameters
ϕ = activation function
b = bias parameter
O = output
Batch size n, and d inputs
t = time (stamp)

$$\mathbf{H}_t = \phi(\mathbf{X}_t\mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h).$$

$$\mathbf{O}_t = \mathbf{H}_t\mathbf{W}_{hq} + \mathbf{b}_q.$$

Output layer

- **Hidden Units**

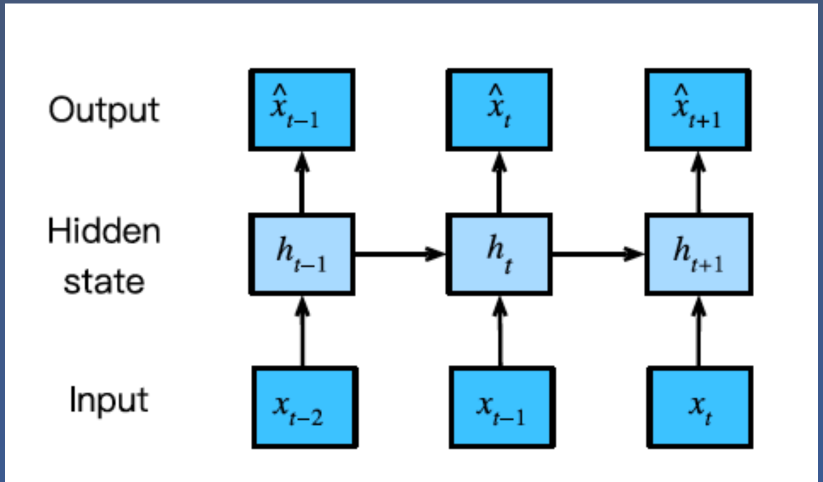
Hidden Units

$$\mathbf{H} = \phi(\mathbf{X}\mathbf{W}_{\mathrm{xh}} + \mathbf{b}_{\mathrm{h}}).$$

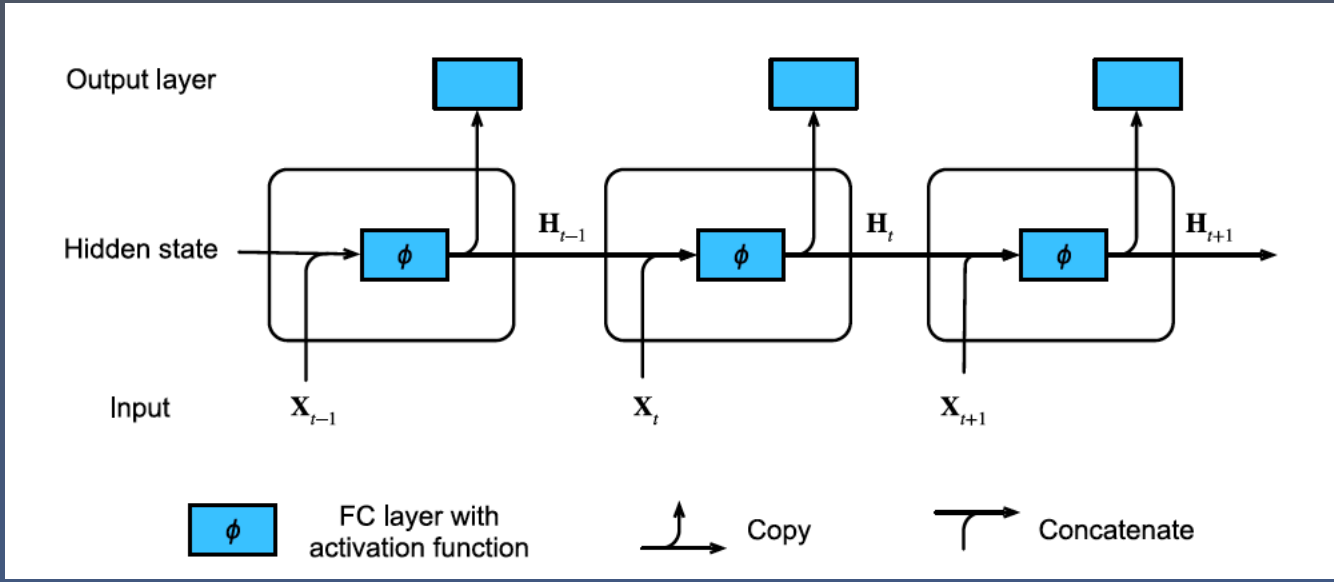
$$\mathbf{O} = \mathbf{H}\mathbf{W}_{\mathrm{hq}} + \mathbf{b}_{\mathrm{q}},$$

$$\mathbf{H}_t = \phi(\mathbf{X}_t\mathbf{W}_{\mathrm{xh}} + \mathbf{H}_{t-1}\mathbf{W}_{\mathrm{hh}} + \mathbf{b}_{\mathrm{h}}).$$

$$\mathbf{O}_t = \mathbf{H}_t\mathbf{W}_{\mathrm{hq}} + \mathbf{b}_{\mathrm{q}}.$$

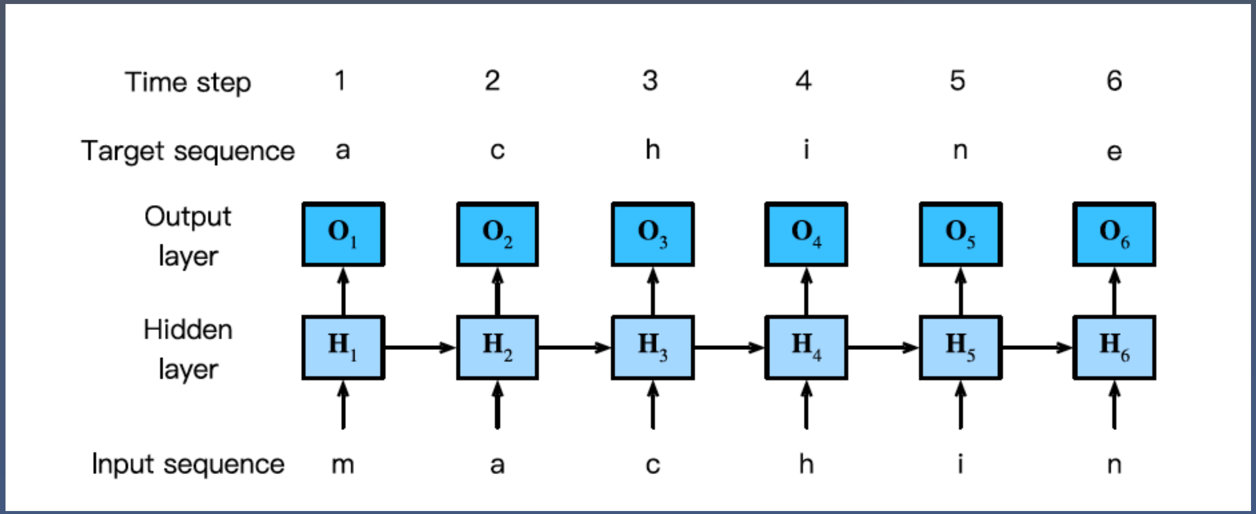


Recurrence



- **Recurrent** computation
- Even at *different* time steps, RNNs always use these *same* model parameters
  - the parametrization cost of an RNN does **not** grow as the number of time steps increases

Character sequence (model)



Gradient Clipping

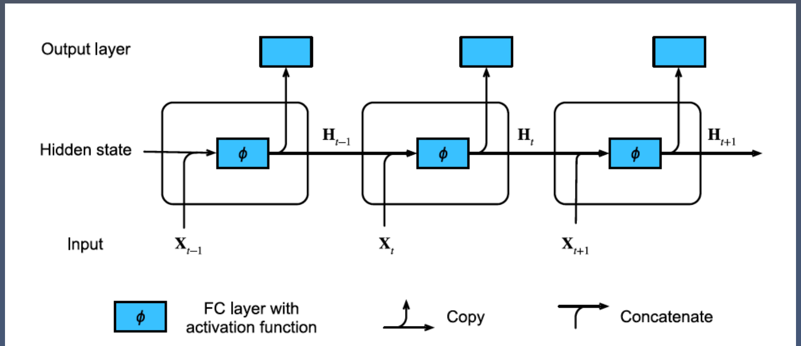
$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|$$

$$|f(\mathbf{x}) - f(\mathbf{x} - \eta\mathbf{g})| \leq L\eta\|\mathbf{g}\|.$$

$$\mathbf{g} \leftarrow \min\left(1, \frac{\theta}{\|\mathbf{g}\|}\right)\mathbf{g}.$$

- Time steps = hidden layers
- More time steps —> more (hidden) layers
  - Vanishing & Exploding gradients

Loss, in RNNs



$$\begin{aligned} h_t &= f(x_t, h_{t-1}, w_h), \\ o_t &= g(h_t, w_o), \end{aligned}$$

$$L(x_1, \ldots, x_T, y_1, \ldots, y_T, w_h, w_o) = \frac{1}{T} \sum_{t=1}^{T} l(y_t, o_t).$$

- Unrolling across time steps
- Sum the gradients in unrolled (can be rather long)

Backpropagation through **time**

$$L(x_1, \ldots, x_T, y_1, \ldots, y_T, w_h, w_o) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, o_t).$$

$$h_t = f(x_t, h_{t-1}, w_h),$$
$$o_t = g(h_t, w_o),$$

$$\frac{d}{dx} f(g(x)) = \frac{df}{dg} \cdot \frac{dg}{dx}$$

$$\begin{aligned}\frac{\partial L}{\partial w_h} &= \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l(y_t, o_t)}{\partial w_h} \\ &= \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l(y_t, o_t)}{\partial o_t} \frac{\partial g(h_t, w_o)}{\partial h_t} \frac{\partial h_t}{\partial w_h}.\end{aligned}$$

$$a_0 = 0 \quad a_t = b_t + c_t a_{t-1}$$

$$a_t = b_t + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} c_j\right) b_i$$

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial w_h}$$

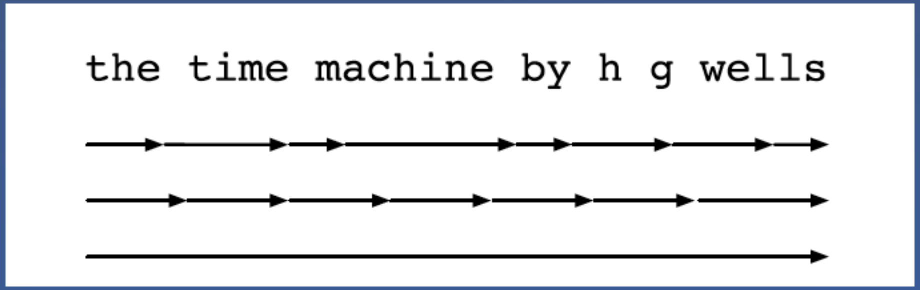
$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_h)}{\partial h_{j-1}}\right) \frac{\partial f(x_i, h_{i-1}, w_h)}{\partial w_h}.$$

- Gradient can blow up

Truncation

$\partial h_{t-\tau} / \partial w_{\mathrm{h}}$

- Terminate the sequence
    - Fixed, Random
- *Short term* dependencies
    - A good thing !
- Truncation (regular or randomized):
    - Computational convenience
    - Numerical stability

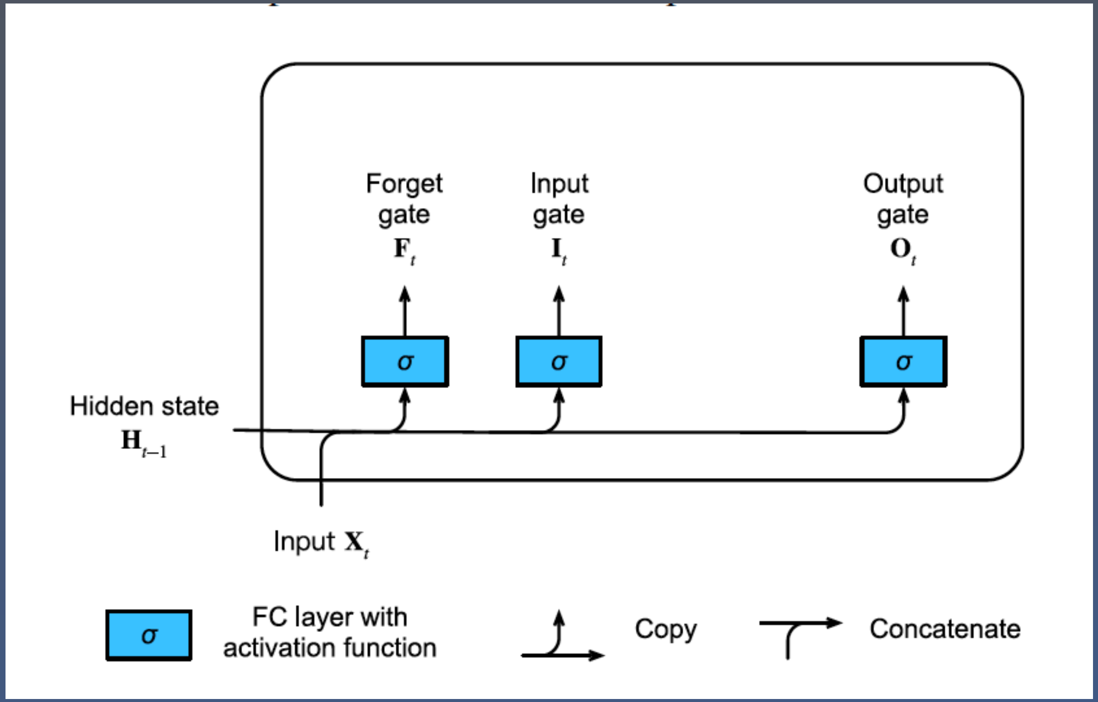


What about **vanishing** gradients ?

Long Short Term Memory: **LSTM**

- Regular recurrent node —> **Memory Cell**
- **Gates**:
    - Input gate
    - Forget gate
    - Output gate

Gates



$$
\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xi}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{hi}} + \mathbf{b}_{\mathrm{i}}),\\
\mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xf}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{hf}} + \mathbf{b}_{\mathrm{f}}),\\
\mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xo}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{ho}} + \mathbf{b}_{\mathrm{o}}),
\end{aligned}
$$

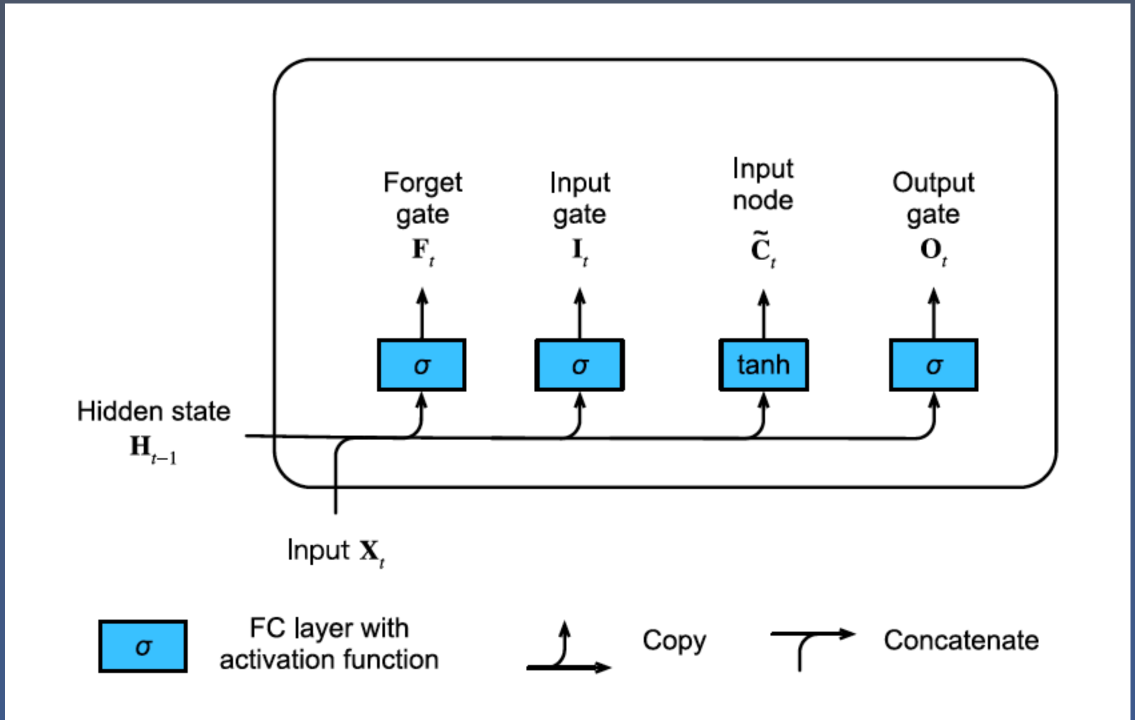
Memory Cell Update

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t\mathbf{W}_{\mathrm{xc}} + \mathbf{H}_{t-1}\mathbf{W}_{\mathrm{hc}} + \mathbf{b}_{\mathrm{c}}),$$

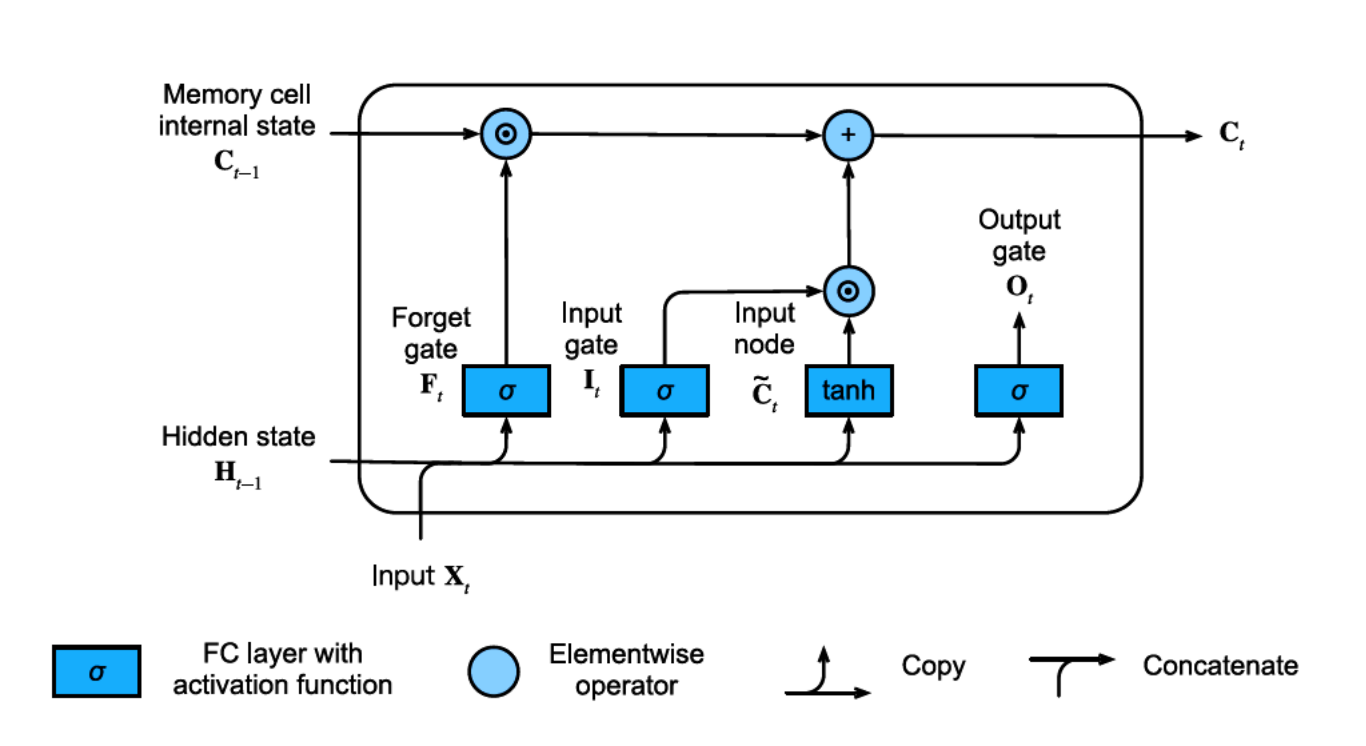
- Input node

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t.$$

- Memory cell state, update



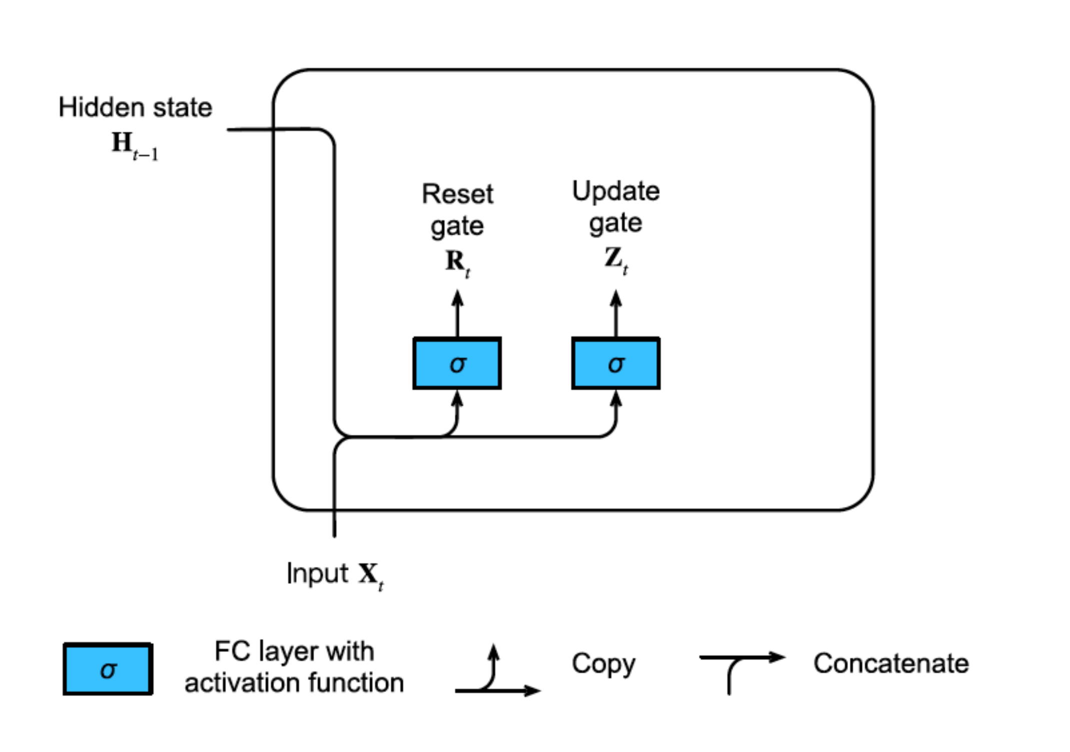
Hidden State Update



$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$

- Output gate: 1 vs 0 ?

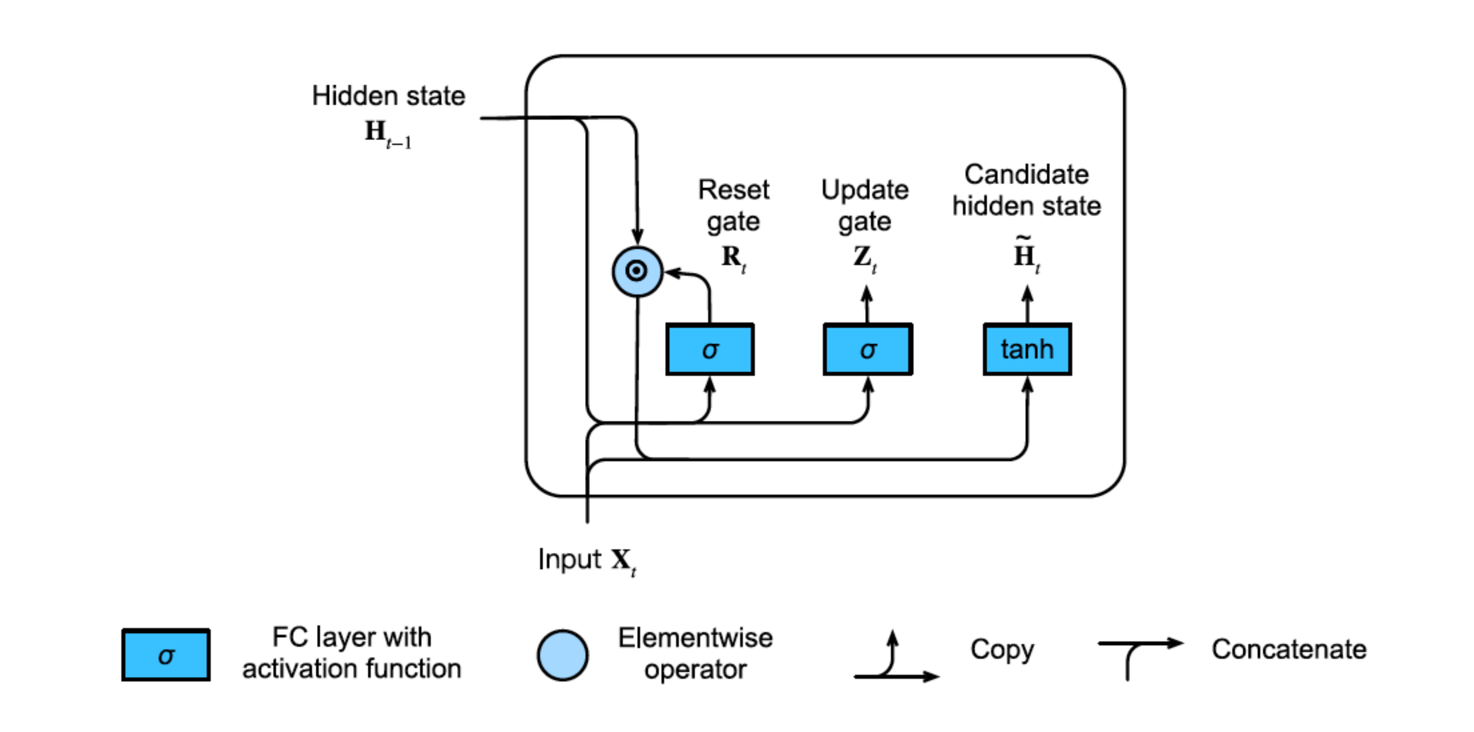
Gated Recurrence Unit GRU



$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xr}} + \mathbf{H}_{t-1}\mathbf{W}_{\mathrm{hr}} + \mathbf{b}_{\mathrm{r}}),$$
$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xz}} + \mathbf{H}_{t-1}\mathbf{W}_{\mathrm{hz}} + \mathbf{b}_{\mathrm{z}}),$$

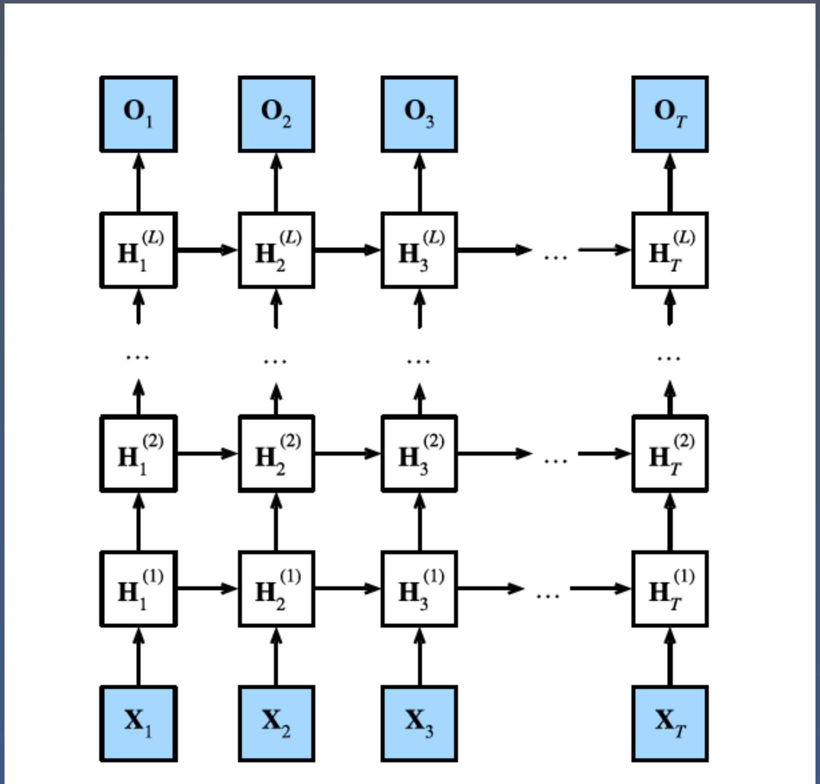
- Update gate : Input + Forget gates (of LSTM) combined
- Reset gate

GRU : Hidden State



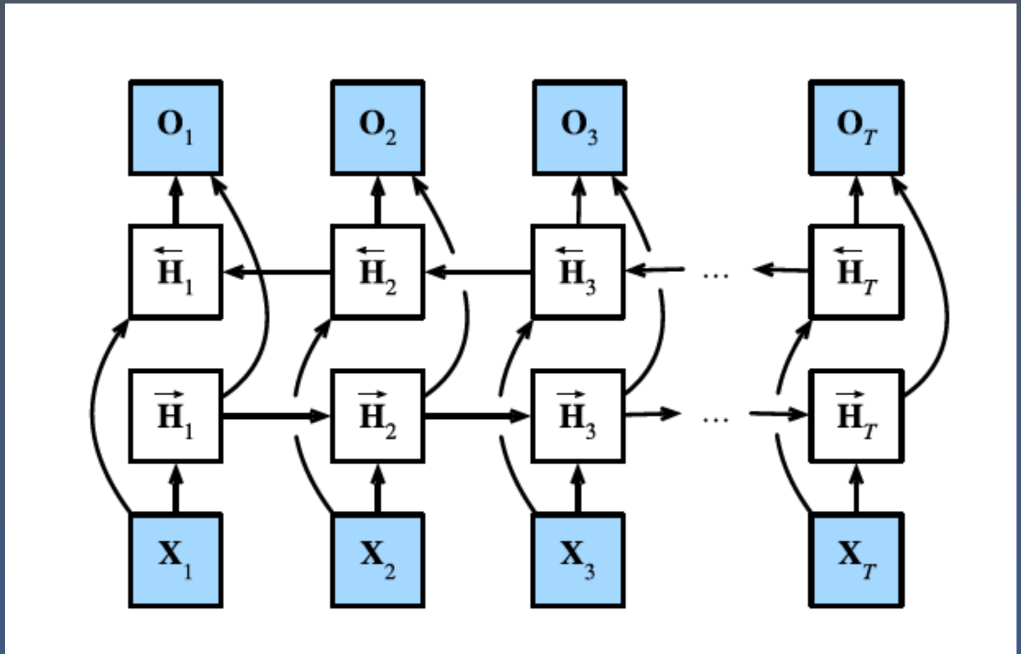
$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h),$$

Deep RNNs



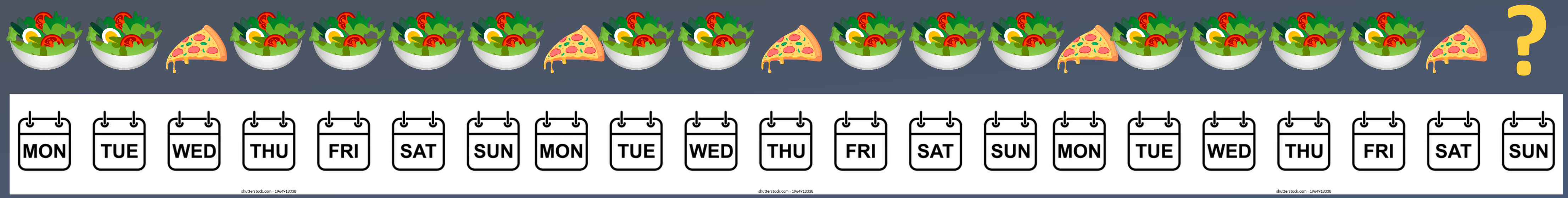
O_1
O_2
O_3
O_T
$H_1^{(L)}$
$H_2^{(L)}$
$H_3^{(L)}$
$H_T^{(L)}$
$H_1^{(2)}$
$H_2^{(2)}$
$H_3^{(2)}$
$H_T^{(2)}$
$H_1^{(1)}$
$H_2^{(1)}$
$H_3^{(1)}$
$H_T^{(1)}$
X_1
X_2
X_3
X_T

Bidirectional RNNs



- Go both forward and backward in the sequence

Summary



The prediction, is based on understanding *"the psychology of the individual" (- Jeeves)*, which in turn is best facilitated by **Recurrent Neural Networks (RNNs)**. We maintain a detailed, day-by-day eating record of the individual, stored as a **Sequence** of time-stamped entries in **Hidden Units**. To keep things manageable, we periodically **Truncate** this sequence, either at fixed intervals or randomly. The individual's clear pattern of eating habits is stored in **Long Short-Term Memory (LSTM)**, where the **Input Gate** carefully preserves the meals from the last 2 days, and the **Forget Gate** ensures nothing beyond those two days is remembered. This recent history is used to predict today's lunch through the Output Gate. A **GRU** simplifies the process, reducing memory to "just the last two days." The **Bidirectional RNN** anticipates tomorrow's pizza, accounting for both past meals and future plans.