9. 1. Gated Recurrent Units (GRU)

- **Long products of matrices** can lead to **vanishing** or **exploding** gradients

Memory cell

- Early observation might be highly significant for predicting all future observations
- Need to store vital early information in a **memory cell**

Skipping

- Some tokens might have **no significant information** or **observation**
- Need to skip such tokens in the latent state representation

Resetting

- There could be a **logical break** between parts of sequence
- Need to **reset** internal state representation

9. 1. 1. Gated Hidden State

9. 1. 1. 1. Reset Gate and Update Gate

- Support dedicated mechanisms for when a **hidden state** should be **updated** and **reset**
- Reset Gate: How much of the previous state we might **still want to remember**
- Update Gate: How much of the new state is **just a copy of the old state**
- Use **sigmoid** to transform input values to the interval (0, 1)

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

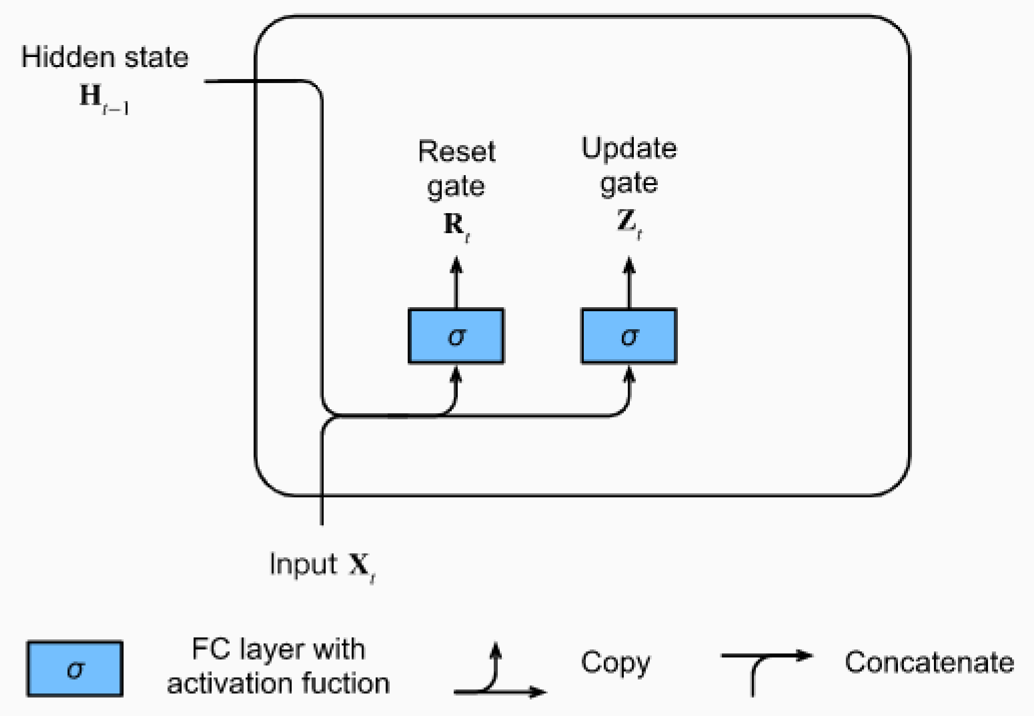

Fig. 9.1.1 Computing the reset gate and the update gate in a GRU model.

9. 1. 1. Gated Hidden State

9. 1. 1. 2. Candidate Hidden State

- Integrate the reset gate, **candidate**, since still need to incorporate the action of update gate
- Whenever the entries in the reset gate are close to 1, recover a vanilla RNN
- Whenever the entries in the reset gate are close to 0, close to MLP with input
- Use **tanh** to ensure values in the candidate hidden state remain in the interval (-1, 1)

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

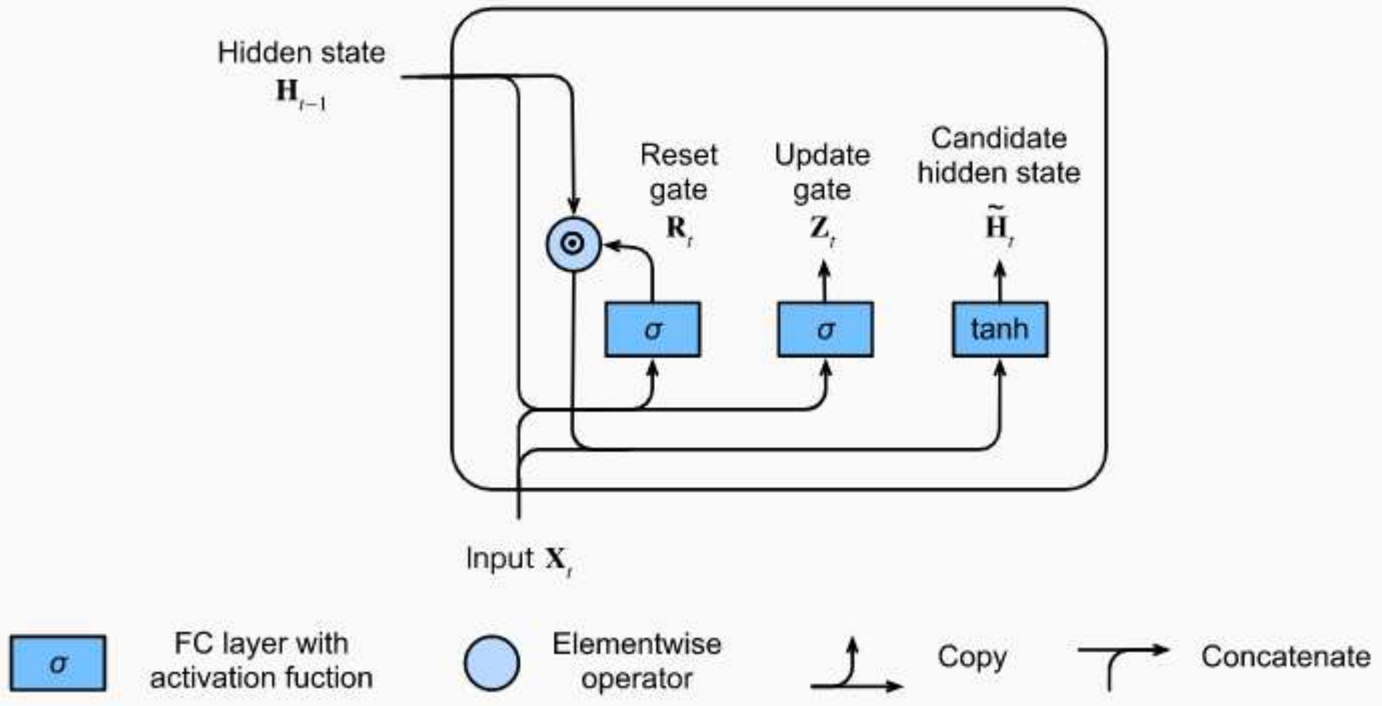

Fig. 9.1.2 Computing the candidate hidden state in a GRU model.

9. 1. 1. Gated Hidden State

9. 1. 1. 3. Hidden State

- Determines the **extent** to which the new hidden state is just the old state and new candidate state
- Whenever update gate is close to 1, simply retain the old state, **current Input X is ignored** skipping time stamp step t in the dependency chain
- Whenever update gate is close to 0, use **candidate latent state**
- This designs can help to avoid vanishing gradients and better capture dependencies for sequences with large time step

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

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t.$$

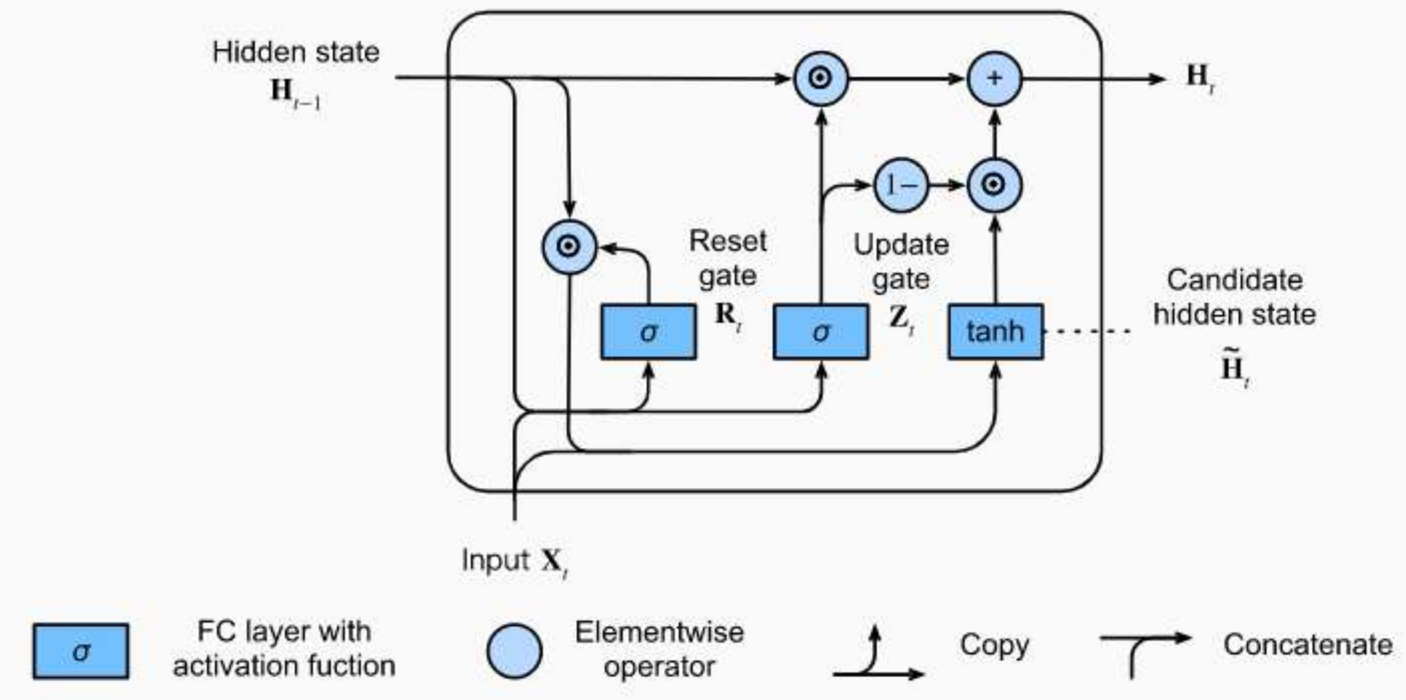

Fig. 9.1.3 Computing the hidden state in a GRU model.

9. 2. Long Short-Term Memory (LSTM)

9. 2. 1. Gated Memory Cell – Input, Forget, Output Gate

- Almost mechanisms are like GRU
- When to remember and when to ignore inputs in the hidden state
- Values of the three gates are in the range of (0, 1)

$$\mathbf{I}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i),$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}_f),$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o),$$

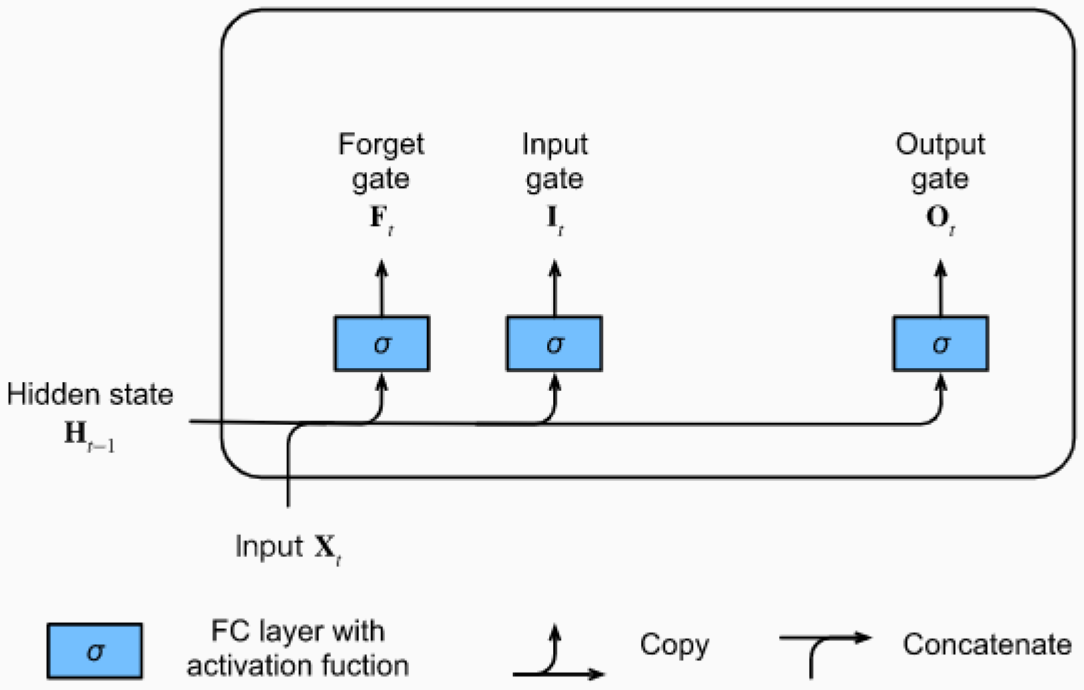

Fig. 9.2.1 Computing the input gate, the forget gate, and the output gate in an LSTM model.

9. 2. Long Short-Term Memory (LSTM)

9. 2. 1. Gated Memory Cell – Candidate Memory Cell

- Use tanh with a value range for (-1, 1)

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

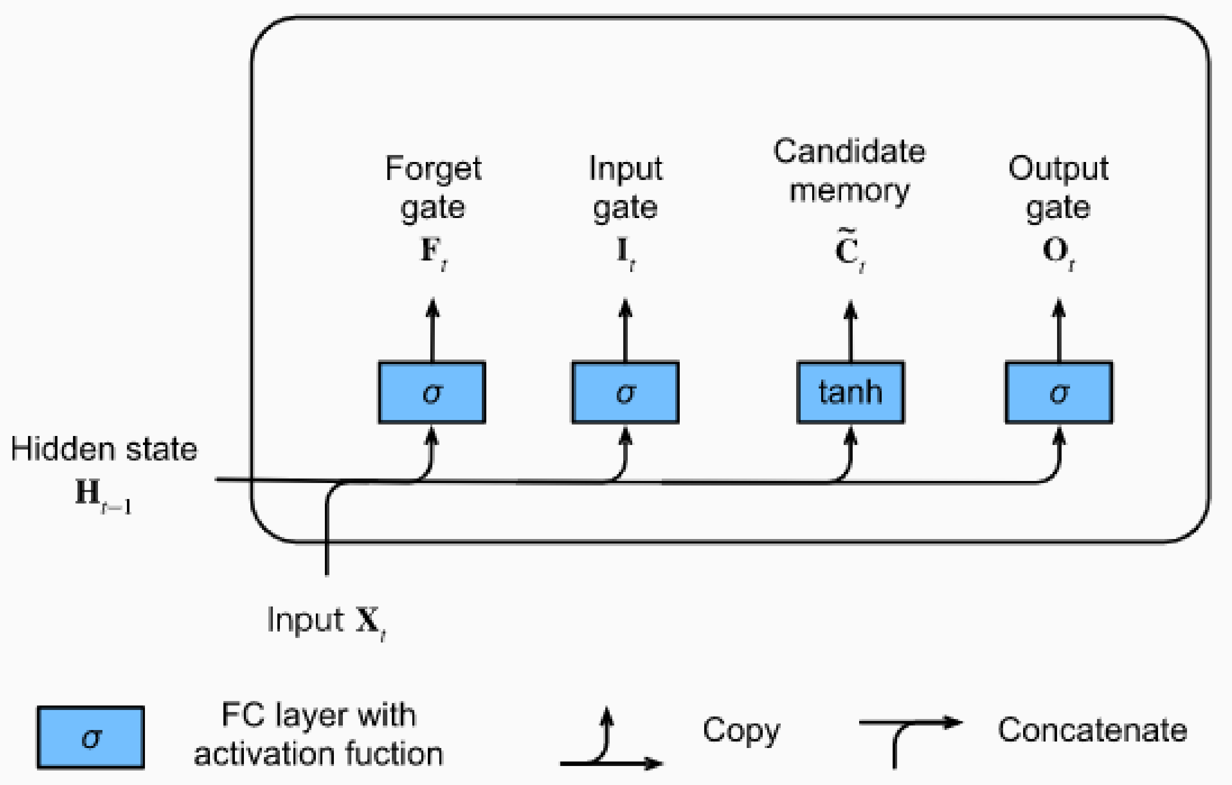

Fig. 9.2.2 Computing the candidate memory cell in an LSTM model.

9. 2. Long Short-Term Memory (LSTM)

9. 2. 1. Gated Memory Cell – Memory Cell

- Input gate I governs how much we take new data via Candidate Cell
- Forget gate F addresses how much of the old memory cell we retain
- If **forget gate** is always approximately **1** and the **input gate** is always approximately **0**, the past memory cells will be saved over time and passed to the current timestamp
- It can avoid vanishing gradient and better capture long range dependencies

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

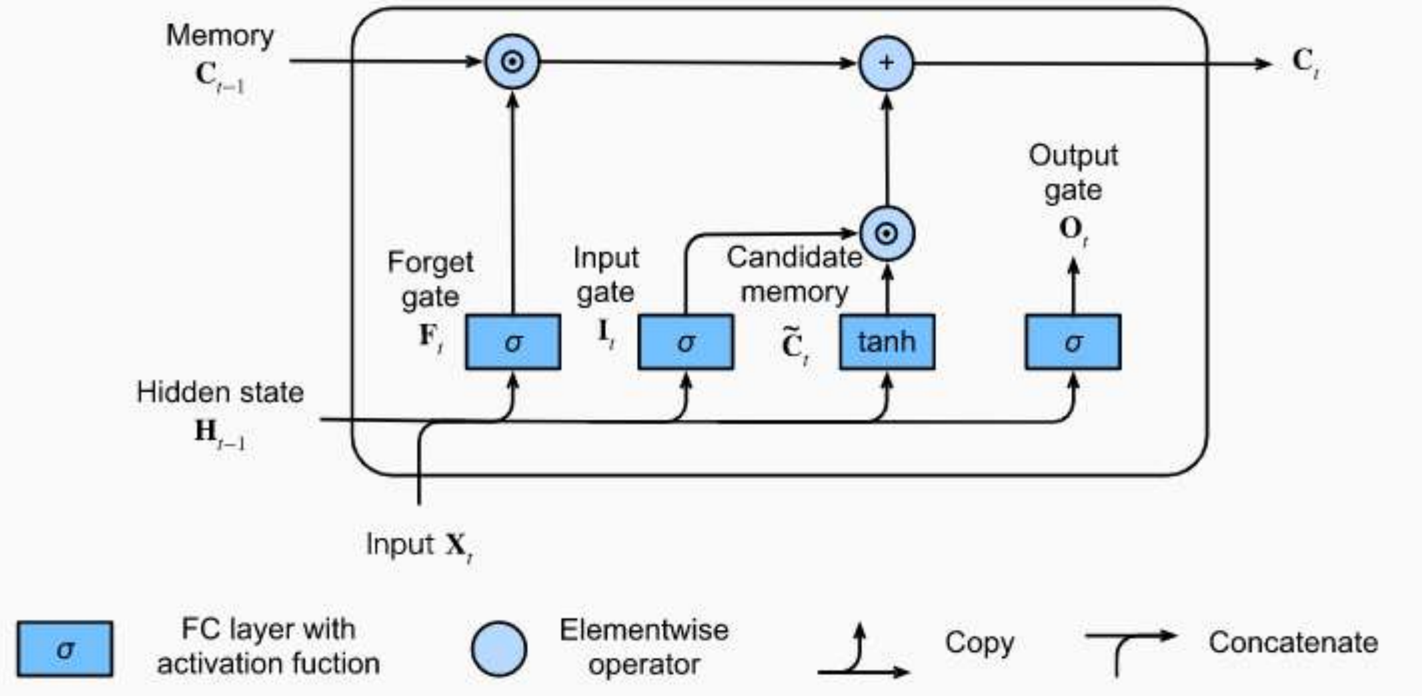

Fig. 9.2.3 Computing the memory cell in an LSTM model.

9. 2. Long Short-Term Memory (LSTM)

9. 2. 1. Gated Memory Cell – Hidden State

- Whenever the output gate approximates 1, effectively pass all memory information
- Whenever the output gate approximates 0, information only retain memory cell and perform no further processing
- LSTMs and GRUs are quite costly due to the long-range dependency of the sequence
- Transformers can be alternatives

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

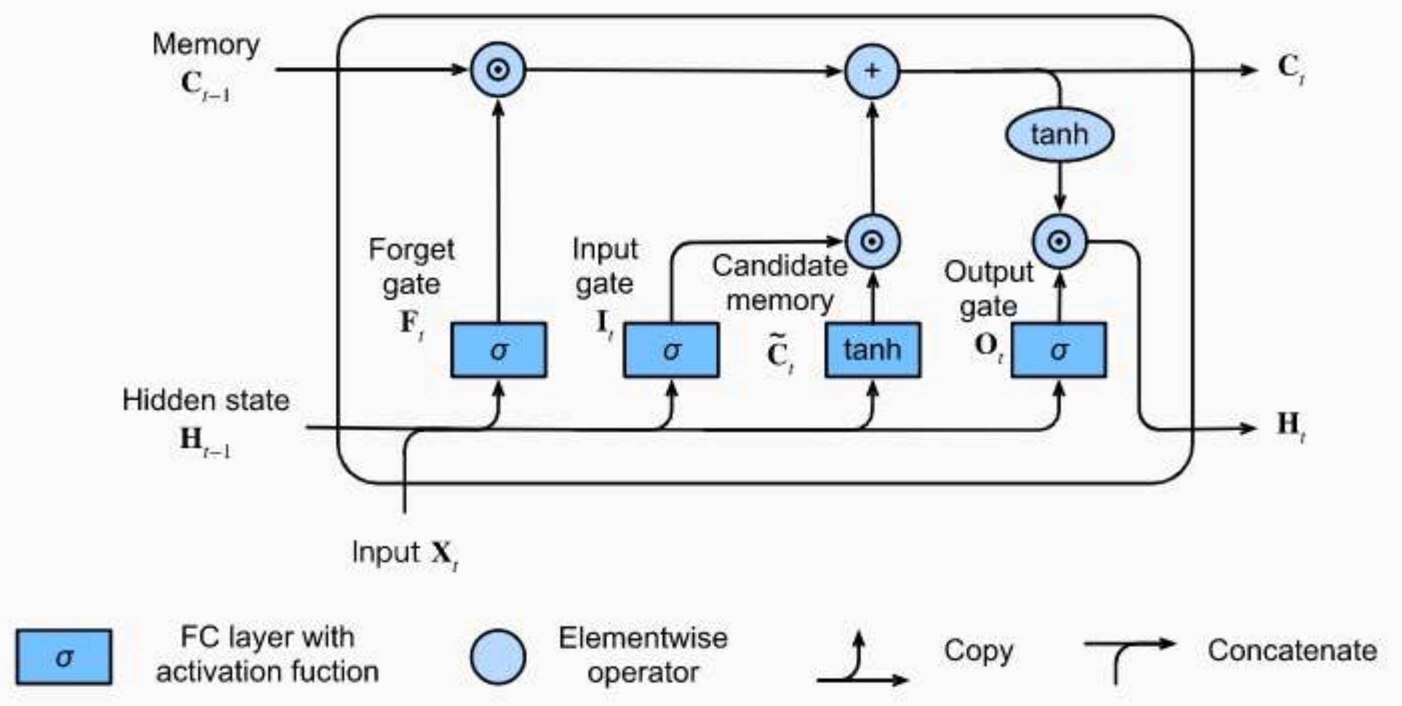

Fig. 9.2.4 Computing the hidden state in an LSTM model.

9. 3. Deep Recurrent Neural Networks

- Up to now, only discussed RNNs with a single unidirectional hidden layer
- With a single layer, it could be challenging to form latent variables
- By stacking multiple layers of RNNs, we could get extra nonlinearity

```
vocab_size, num_hiddens, num_layers = len(vocab), 256, 2
num_inputs = vocab_size
device = d2l.try_gpu()
lstm_layer = nn.LSTM(num_inputs, num_hiddens, num_layers)
model = d2l.RNNModel(lstm_layer, len(vocab))
model = model.to(device)
```

$$\mathbf{H}_t^{(l)} = \phi_l(\mathbf{H}_t^{(l-1)}\mathbf{W}_{xh}^{(l)} + \mathbf{H}_{t-1}^{(l)}\mathbf{W}_{hh}^{(l)} + \mathbf{b}_h^{(l)}),$$

$$\mathbf{O}_t = \mathbf{H}_t^{(L)}\mathbf{W}_{hq} + \mathbf{b}_q,$$

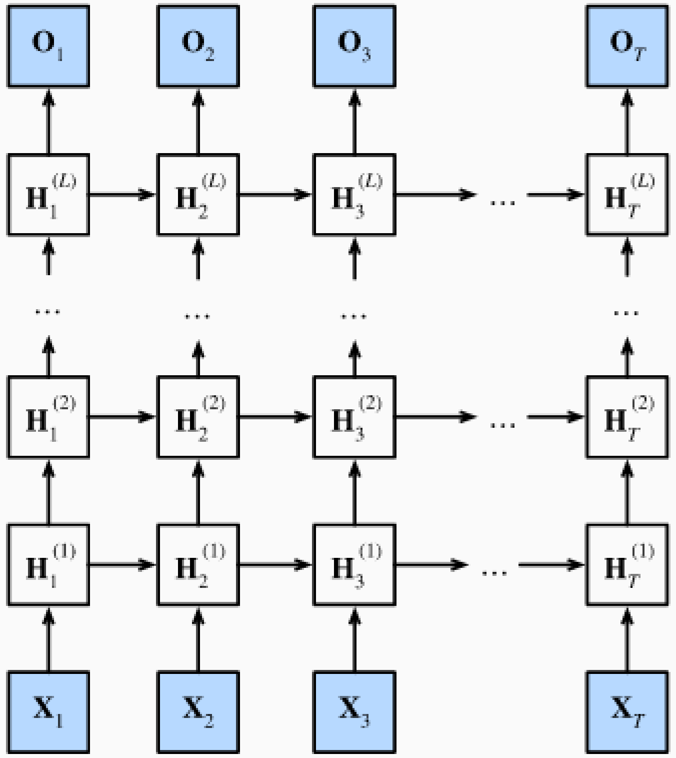

Fig. 9.3.1 Architecture of a deep RNN.

9. 4. Bidirectional Recurrent Neural Networks

- In sequence learning, we assumed that our goal is to model the **next output** given what we have seen
- While this is a typical scenario, not the only one we encounter
  - I am ___
  - I am ___ hungry
  - I am ___ hungry, and I can eat half a pig
- Sometimes longer-range context is vital (e.g. NER, Green refers to Mr. Green or the colr)

9. 4. 1. Dynamic Programming in HMM

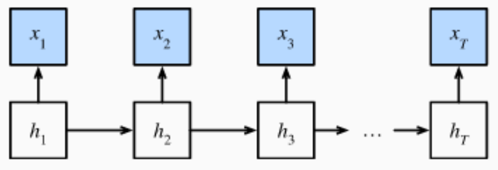

Fig. 9.4.1 A hidden Markov model.

- At any time step t, latent variable h_t that governs our observed emission x_t via $P(x_t|h_t)$
- Any transition, $h_t \rightarrow h_{t+1}$, given by some state transition probability $P(h_{t+1}|h_t)$

$$P(x_1,\ldots,x_T,h_1,\ldots,h_T)=\prod_{t=1}^{T}P(h_t\mid h_{t-1})P(x_t\mid h_t),\ \text{where } P(h_1\mid h_0)=P(h_1).$$

- Forward passing

$$\begin{aligned}
&P(x_1,\ldots,x_T)\\
=&\sum_{h_1,\ldots,h_T}P(x_1,\ldots,x_T,h_1,\ldots,h_T)\\
=&\sum_{h_1,\ldots,h_T}\prod_{t=1}^{T}P(h_t\mid h_{t-1})P(x_t\mid h_t)\\
=&\sum_{h_2,\ldots,h_T}\underbrace{\left[\sum_{h_1}P(h_1)P(x_1\mid h_1)P(h_2\mid h_1)\right]}_{\pi_2(h_2)\stackrel{\text{def}}{=}}P(x_2\mid h_2)\prod_{t=3}^{T}P(h_t\mid h_{t-1})P(x_t\mid h_t)\\
=&\sum_{h_3,\ldots,h_T}\underbrace{\left[\sum_{h_2}\pi_2(h_2)P(x_2\mid h_2)P(h_3\mid h_2)\right]}_{\pi_3(h_3)\stackrel{\text{def}}{=}}P(x_3\mid h_3)\prod_{t=4}^{T}P(h_t\mid h_{t-1})P(x_t\mid h_t)\\
=&\ldots\\
=&\sum_{h_T}\pi_T(h_T)P(x_T\mid h_T).
\end{aligned}$$

$$\pi_{t+1}(h_{t+1})=\sum_{h_t}\pi_t(h_t)P(x_t\mid h_t)P(h_{t+1}\mid h_t).$$

- Backward passing

$$\begin{aligned}
&P(x_1,\ldots,x_T)\\
=&\sum_{h_1,\ldots,h_T}P(x_1,\ldots,x_T,h_1,\ldots,h_T)\\
=&\sum_{h_1,\ldots,h_T}\prod_{t=1}^{T-1}P(h_t\mid h_{t-1})P(x_t\mid h_t)\cdot P(h_T\mid h_{T-1})P(x_T\mid h_T)\\
=&\sum_{h_1,\ldots,h_{T-1}}\prod_{t=1}^{T-1}P(h_t\mid h_{t-1})P(x_t\mid h_t)\cdot\underbrace{\left[\sum_{h_T}P(h_T\mid h_{T-1})P(x_T\mid h_T)\right]}_{\rho_{T-1}(h_{T-1})\stackrel{\text{def}}{=}}\\
=&\sum_{h_1,\ldots,h_{T-2}}\prod_{t=1}^{T-2}P(h_t\mid h_{t-1})P(x_t\mid h_t)\cdot\underbrace{\left[\sum_{h_{T-1}}P(h_{T-1}\mid h_{T-2})P(x_{T-1}\mid h_{T-1})\rho_{T-1}(h_{T-1})\right]}_{\rho_{T-2}(h_{T-2})\stackrel{\text{def}}{=}}\\
=&\ldots\\
=&\sum_{h_1}P(h_1)P(x_1\mid h_1)\rho_1(h_1).
\end{aligned}$$

$$\rho_{t-1}(h_{t-1})=\sum_{h_t}P(h_t\mid h_{t-1})P(x_t\mid h_t)\rho_t(h_t),$$

9. 4. 2. Bidirectional Model

$$\overrightarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(f)} + \overrightarrow{\mathbf{H}}_{t-1} \mathbf{W}_{hh}^{(f)} + \mathbf{b}_h^{(f)}),$$
$$\overleftarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(b)} + \overleftarrow{\mathbf{H}}_{t+1} \mathbf{W}_{hh}^{(b)} + \mathbf{b}_h^{(b)}),$$

- Key features of bidirectional model use information from both future and past observations to predict the current one
- Use bidirectional RNN naively can compute poor accuracy
  - During training we have pas and future
  - During test, we only have past data
- Also extremely slow to compute forward propagation requires both forward and backward recursions
- In practice, bidirectional layers are used very narrow set of apps
  - Filling in missing words
  - Annotating tokens (e.g. NER)
  - Encoding sequences wholesale as a step-in pipeline (machine translation)

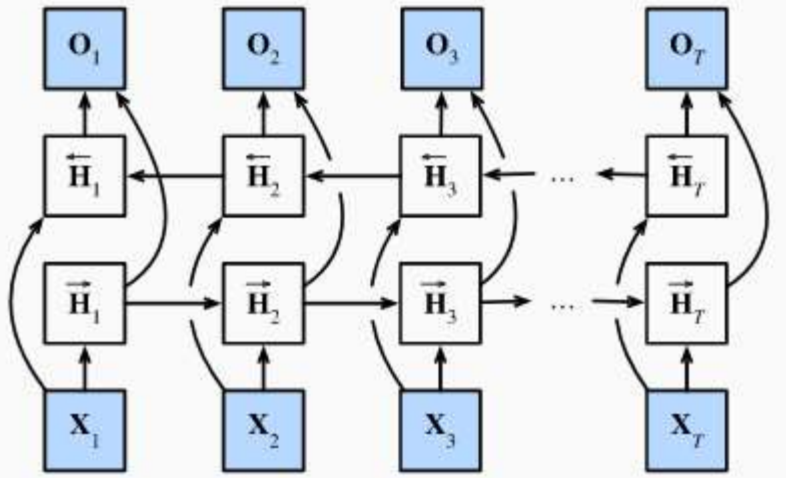

Fig. 9.4.2 Architecture of a bidirectional RNN.

9. 4. 2. Bidirectional Model

- Key features of bidirectional model use information from both future and past observations to predict the current one
- Use bidirectional RNN naively can compute poor accuracy
  - During training we have pas and future
  - During test, we only have past data

```
from d2l import torch as d2l
import torch
from torch import nn

# Load data
batch_size, num_steps, device = 32, 35, d2l.try_gpu()
train_iter, vocab = d2l.load_data_time_machine(batch_size, num_steps)
# Define the bidirectional LSTM model by setting `bidirectional=True`
vocab_size, num_hiddens, num_layers = len(vocab), 256, 2
num_inputs = vocab_size
lstm_layer = nn.LSTM(num_inputs, num_hiddens, num_layers, bidirectional=True)
model = d2l.RNNModel(lstm_layer, len(vocab))
model = model.to(device)
# Train the model
num_epochs, lr = 500, 1
d2l.train_ch8(model, train_iter, vocab, lr, num_epochs, device)
```

```
perplexity 1.1, 117322.8 tokens/sec on cuda:0
time travellererererererererererererererererererererererererer
travellererererererererererererererererererererererererererer
```

```
perplexity 1.0, 197213.3 tokens/sec on cuda:0
time travelleryou can show black is white by argument said filby
traveller with a slight accession ofcheerfulness really thi
```