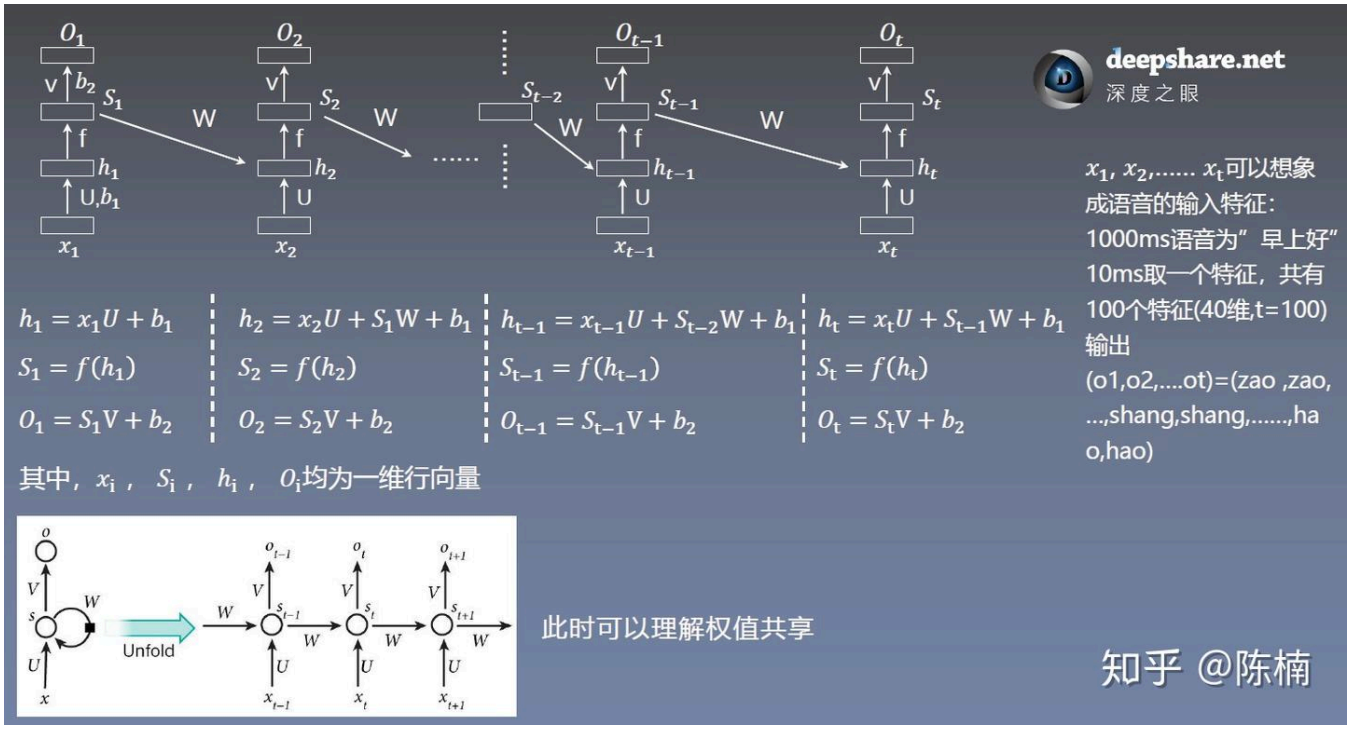
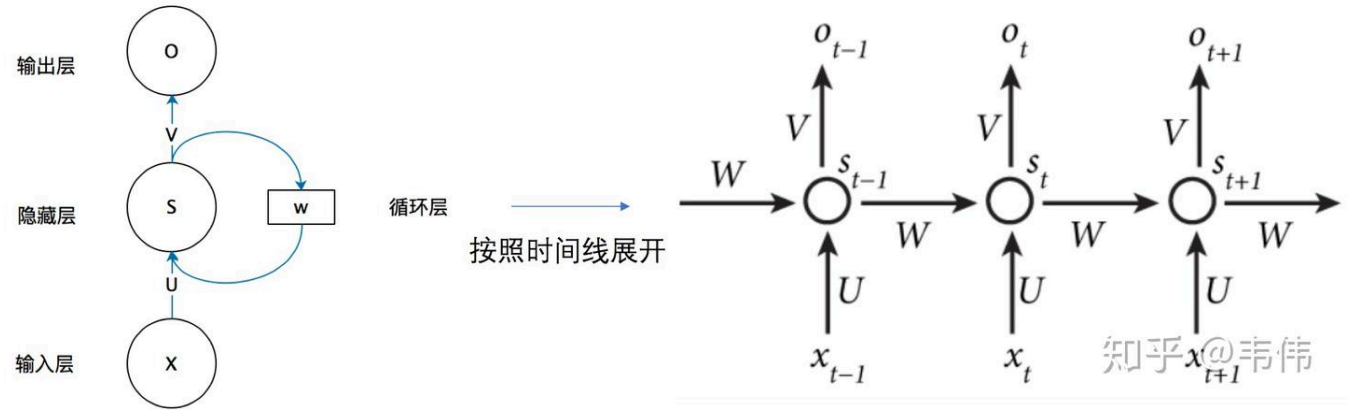
RNN,LSTM,GRU计算方式

RNN计算方式

`torch.nn.RNN(input_size, hidden_size, num_layers=1, nonlinearity='tanh', bias=True, batch_first=False, dropout=0.0, bidirectional=False, device=None, dtype=None)`





$$h_t = \tanh(x_t W_{ih}^T + b_{ih} + h_{t-1} W_{hh}^T + b_{hh})$$

```
# Efficient implementation equivalent to the following with bidirectional=False
def forward(x, h_0=None):
    if batch_first:
        x = x.transpose(0, 1)
    seq_len, batch_size, _ = x.size()
    if h_0 is None:
        h_0 = torch.zeros(num_layers, batch_size, hidden_size)
    h_t_minus_1 = h_0
    h_t = h_0
    output = []
    for t in range(seq_len):
```

```
        for layer in range(num_layers):
            h_t[layer] = torch.tanh(
                x[t] @ weight_ih[layer].T
                + bias_ih[layer]
                + h_t_minus_1[layer] @ weight_hh[layer].T
                + bias_hh[layer]
            )
        output.append(h_t[-1])
        h_t_minus_1 = h_t
    output = torch.stack(output)
    if batch_first:
        output = output.transpose(0, 1)
    return output, h_t
```

- 参数：
  ·input_size-输入X中预期特征的数量
  ·hidden size-隐藏状态h中的特征数量
  ·num_layers-循环层数。例如，设置num_layers:=2表示将两个RNN堆叠在一起形成一个堆叠RNW,第二个RNN接收第一个RNN的输出并计算最终结果。默认为：1
  ·nonlinearity-使用的非线性函数。可以是"tanh或"relu'。默认为："tanh
  ·bias-如果为False,则该层不使用偏置权重bh和bhh。默认为：True
  ·batch_first-如果为True,则输入和输出张量提供为(batch,seq,feature),而不是(seg,batch,feature)。请注意，这不适用于隐藏状态或单元状态。有关详细信息，请参阅下面的输入/输出部分。默认为：False
  ·dropout-如果非零，则在除最后一层外的每个RNN层的输出上引入Dropout层，dropout概率等于dropout。默认为：0
  ·bidirectional-如果为True,则成为双向RNN。默认为：False

输入：input,,hx
·input:形状为(L,Hin)的张量（用于无批次输入），形状为(L,N,Hn)当batch_first=:False时，或形状为(N,L,Him)当batch_first=True时，包含输入序列的特征。输入也可以是填充的可变长度序列。有关详细信息，请参阅torch.nn.utils.rnn,pack padded sequence()或torch.nn.utils.rnn.pack_sequence()

·hx:形状为(D*num_layers,Hout)的张量（用于无批次输入），或形状为(D*num layers,.N,Hout)的张量（用于输入序列批次）包含初始隐藏状态。如果未提供，则默认为零。

其中

$$
\begin{aligned}
N &= \text{batch size} \\
L &= \text{sequence length} \\
D &= 2 \text{ if bidirectional=True otherwise } 1 \\
H_{in} &= \text{input size} \\
H_{out} &= \text{hidden size}
\end{aligned}
$$

输出：output,hn

output:形状为(L,D*Hot)的张量（用于无批次输入），或形状为(L,N,D*Hout)当batch_first:=False时，或形状为(W,L,D*Hout)当batch_first=True时，包含从RNN最后一层在每个t的输出特征ht)。如果输入是torch.nn.utils.rnn.PackedSequence,则输出也将是打包序列。

hn:形状为(D*num _layers,Hot)的张量（用于无批次输入），或形状为D*num_layers,N,Hout)的张量（用于批次中的每个元素）包含最终隐藏状态。

变量：

weight ih[k]-第k层的可学习输入-隐藏权重，对于k=O,形状为hidden size,input size)。否则，形状(hidden size,num_directions hidden_size)

weight_hh[k-第k层的可学习隐藏-隐藏权重，形状为hidden_size,.hidden_size)

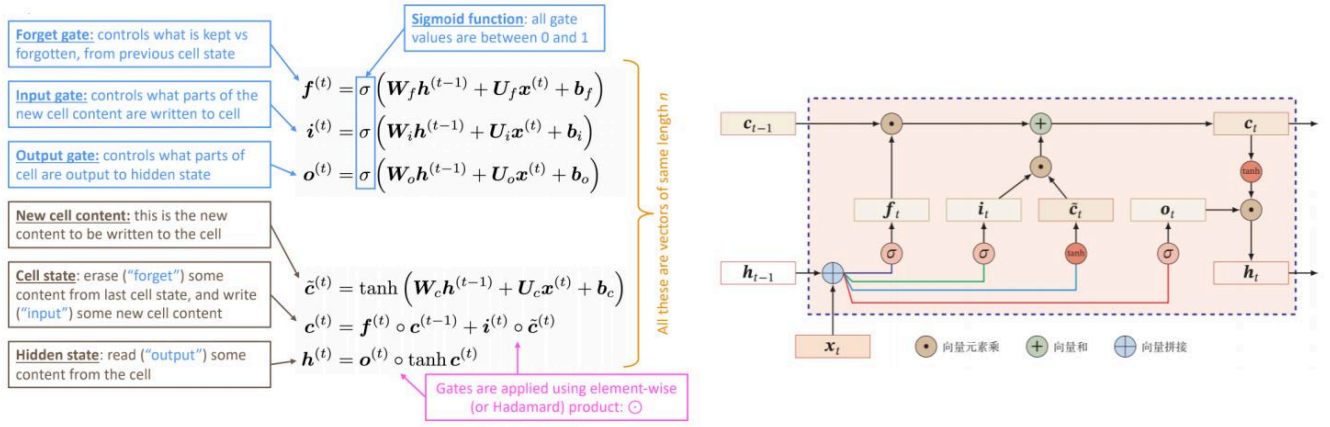
bias ih l[k-第k层的可学习输入-隐藏偏置，形状为hidden_size)

bias_hh[k-第k层的可学习隐藏-隐藏偏置，形状为hidden_size)

```
rnn = nn.RNN(10, 20, 2)
input = torch.randn(5, 3, 10)
h0 = torch.randn(2, 3, 20)
output, hn = rnn(input, h0)
```

LSTM计算方式

`torch.nn.LSTM(input_size, hidden_size, num_layers=1, bias=True, batch_first=False, dropout=0.0, bidirectional=False, proj_size=0, device=None, dtype=None)`



$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

在时间步 t，ht 表示隐藏状态，Ct 表示细胞状态，xt 表示时间步 t 的输入，ht−1 表示上一时间步（t−1）的隐藏状态，或在初始时刻 t=0 时的初始隐藏状态。it、ft、gt、ot 分别表示输入门、遗忘门、细胞门和输出门。σ 是 sigmoid 激活函数，⊙ 表示哈达玛积（逐元素相乘）。

在多层 LSTM 中，第 l 层的输入是前一层隐藏状态乘以一个 dropout 掩码，其中每个元素是一个伯努利随机变量，以概率 p 为 0（即 dropout 概率为 p）。

如果指定了 proj_size > 0，则将使用带有投影的 LSTM（LSTM with projections），这会对 LSTM 单元做如下修改：

第一，ht 的维度将从 hidden_size 改变为 proj_size（相应地，权重矩阵 Whi 的维度也会改变）。

第二，每一层输出的隐藏状态将乘以一个可学习的投影矩阵：ht = Whr · ht，其中 Whr 是投影矩阵。

需要注意的是，由于这一修改，LSTM 网络的输出形状也会随之改变。关于所有变量的精确维度，请参见下文的“输入/输出”部分。

更多细节可参考论文：https://arxiv.org/abs/1402.1128。

参数：

> input_size-输入x中预期特征的数量

> hidden size-隐藏状态h中的特征数量·num_layers-循环层数。例如，设置num_layers=2意味着将两个LSTM堆叠在一起形成一个“堆叠 LSTM”,第二个LSTM接收第一个LSTM的输出来计算最终结果。默认为1。

> bias-如果为False,则该层不使用偏置权重bh和bhh。默认为True o

> batch_first-如果为True,则输入和输出张量将提供为(batch,seq,feature),而不是(seq,batch,feature)o请注意，这不适用于隐藏状态或单元状态。有关详细信息，请参阅下面的“输入/输出"部分。默认为False。

> dropout-如果非零，则在除最后一层外的每个LSTM层的输出上引入Dropout层，dropout概率等于 dropout。默认为O。

> bidirectional-如果为True,则变为双向LSTM。默认为False。

> proj_size-如果>0，则将使用具有相应大小的投影的LSTM。默认为0。

输入：input,.(h0,cO)

> input:形状为(L,Hn)的张量，用于未批处理的输入；当batch_first=False时，形状为(L,N,Hm),或者当batch_first=True时，形状为(N,L,Hm),其中包含输入序列的特征。输入也可以是打包的可变长度序列。有关详细信息，请参阅torch.nn.utils.rnn.pack_padded_sequence()或torch.nn.utils.rnn.pack_sequence()

> h0:形状为(D*num_layers,Hot)的张量，用于未批处理的输入；当batch first:=False时，形状为(D*num_layers,N,Hout)的张量，用于批处理的输入，包含输入序列的初始隐藏状态。如果未提供 h0,c0),则默认为零。

> c_0:形状为(D*num layers,.Hce)的张量，用于未批处理的输入；当batch_first=False时，形状为(D*num layers,N,Hcen)的张量，用于批处理的输入，包含输入序列的初始单元状态。如果未提供 h0,c0),则默认为零

其中

$$
\begin{aligned}
N &= \text{batch size} \\
L &= \text{sequence length} \\
D &= 2 \text{ if bidirectional=True otherwise } 1 \\
H_{in} &= \text{input}_{\text{size}} \\
H_{cell} &= \text{hidden}_{\text{size}} \\
H_{out} &= \text{proj}_{\text{size}} \text{ if } \text{proj}_{\text{size}} > 0 \text{ otherwise } \text{hidden}_{\text{size}}
\end{aligned}
$$

输出：output,(hn,cn)

> output:形状为(L,D*Hot)的张量，用于未批处理的输入；当batch first=False时，形状为(L,N,D*Hot),或者当batch_first=True时，形状为(N,L,D*Hout),包含LSTM最后-个时间步的输出特征ht)。如果输入是torch.nn.utils.rnn.PackedSequence,则输出也将是打包序列。当 bidirectional=:True时，output将包含每个时间步的前向和后向隐藏状态的连接。

> hn:形状为(D*num_layers,Hot)的张量，用于未批处理的输入；当batch_first=False时，形状为(D*num_layers,N,Hot)的张量，用于批处理的输入，包含序列中每个元素的最终隐藏状态。当bidirectional=True时，hn将分别包含最终前向和后向隐藏状态的连接。

> cn:形状为(D*num_layers,.Hce)的张量，用于未批处理的输入；当batch_first=False时，形状为(D*num_layers,N,Hceu)的张量，用于批处理的输入，包含序列中每个元素的最终单元状态。当bidirectional=True时，Gn将分别包含最终前向和后向单元状态的连接。

参数：

weight_ih_l[k]：第 k 层的可学习输入到隐藏层权重（对应输入门、遗忘门、细胞门、输出门，即 Wii,Wif,Wig,Wio ），其形状为：

当 k=0 时：(4×hidden_size,input_size) ；

- 当 k>0 时：(4×hidden_size,num_directions×hidden_size) ；
- 若指定了 proj_size>0 且 k>0 ：形状为 (4×hidden_size,num_directions×proj_size) 。

---

weight_hh_l[k]：第 k 层的可学习隐藏层到隐藏层权重（即 Whi,Whf,Whg,Who ），其形状为：

- (4×hidden_size,hidden_size) ；
- 若指定了 proj_size>0 ，则形状为 (4×hidden_size,proj_size) 。

---

bias_ih_l[k]：第 k 层的可学习输入到隐藏层偏置（即 bii,bif,big,bio ），形状为 (4×hidden_size) 。

---

bias_hh_l[k]：第 k 层的可学习隐藏层到隐藏层偏置（即 bhi,bhf,bhg,bho ），形状为 (4×hidden_size)。

---

weight_hr_l[k]：第 k 层的可学习投影权重，形状为 (proj_size,hidden_size) 。
仅在指定了 proj_size>0 时存在。

---

weight_ih_l[k]_reverse：反向传播方向中，第 k 层的输入到隐藏层权重，结构与 `weight_ih_l[k]` 类似。
仅在 `bidirectional=True` 时存在。

---

weight_hh_l[k]_reverse：反向传播方向中，第 k 层的隐藏层到隐藏层权重，结构与 `weight_hh_l[k]` 类似。
仅在 `bidirectional=True` 时存在。

---

bias_ih_l[k]_reverse：反向传播方向中，第 k 层的输入到隐藏层偏置，结构与 `bias_ih_l[k]` 类似。
仅在 `bidirectional=True` 时存在。

---

bias_hh_l[k]_reverse：反向传播方向中，第 k 层的隐藏层到隐藏层偏置，结构与 `bias_hh_l[k]` 类似。
仅在 `bidirectional=True` 时存在。

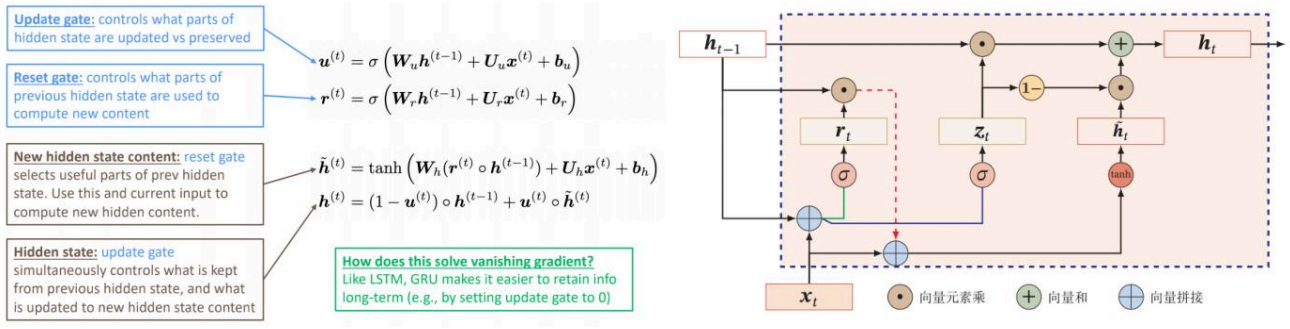
---

weight_hr_l[k]_reverse：反向传播方向中，第 k 层的可学习投影权重，结构与 `weight_hr_l[k]` 类似。
仅在 `bidirectional=True` 且指定了 proj_size>0 时存在。

```
rnn = nn.LSTM(10, 20, 2)
input = torch.randn(5, 3, 10)
h0 = torch.randn(2, 3, 20)
c0 = torch.randn(2, 3, 20)
output, (hn, cn) = rnn(input, (h0, c0))
```

GRU计算方式

`torch.nn.GRU(input_size, hidden_size, num_layers=1, bias=True, batch_first=False, dropout=0.0, bidirectional=False, device=None, dtype=None)`



$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \\
h_t &= (1 - z_t) \odot n_t + z_t \odot h_{(t-1)}
\end{aligned}
$$

其中，ht是时间t的隐藏状态，xt是时间t的输入，九(t-1)是时间t1的层隐藏状态或时间0的初始隐藏状态，rt、 t、nt分别是重置门、更新门和新门。o是sigmoid函数，⊙是阿达玛积(Hadamard product)。

在多层GRU中，第l层≥2)的输入x四是前一层的隐藏状态h-1)乘以dropout-,其中每个d-)都是一个伯努利随机变量，其取值为0的概率等于dropout。

参数：

input_size-输入X中预期特征的数量

hidden size-隐藏状态h中的特征数量·num_layers-循环层的数量。例如，设置 num_layers=2意味着将两个GRU堆叠在一起形成一个堆叠 GRU,其中第二个GRU接收第一个GRU的输出并计算最终结果。默认值：1

bias-如果为False,则层不使用偏差权重bh和bhh。默认值：True

batch_first-如果为True,则输入和输出张量以batch,seq,feature)形式提供，而不是(seq,batch, feature)。请注意，这不适用于隐藏状态或单元状态。有关详细信息，请参阅下面的输入/输出部分。默认值： False

dropout-如果非零，则在除最后一层之外的每个GRU层的输出上引入一个Dropout层，dropout概率等于 dropout。默认值：0

bidirectional-如果为True,则成为双向GRU。默认值： False

输入：input,hO

input:对于未批处理输入，形状为(L,Hm)的张量；当batch_first:=False时，形状为(L,W,Hn),当batch first:=True时，形状为(N,L,Hn),包含输入序列的特征。输入也可以是填充后的可变长度序列。详情请参见torch.nn.utils.rnn.pack_padded_sequence()或 torch.nn.utils.rnn.pack sequence()o。

h0:形状为(D*num_layers,Hot)或(D*num layers,N,Hout)的张量，包含输入序列的初始隐藏状态。如果未提供，则默认为零。

其中：

$$
\begin{aligned}
N &= \text{batch size} \\
L &= \text{sequence length} \\
D &= 2 \text{ if bidirectional=True otherwise } 1 \\
H_{in} &= \text{input size} \\
H_{out} &= \text{hidden size}
\end{aligned}
$$

输出：output,h_n

output:对于未批处理输入，形状为（L,D*Hout）的张量；当batch_first=False时，形状为（L,N，D*Hout），当batch_first=True时，形状为（N，L,D*Hout），包含GRU最后一层在每个t的输出特征 (h_t)。如果输入是torch.nn.utils.rnn.PackedSequence，则输出也将是打包序列。

h_n:形状为（Dnum_layers,Hout）或（D*num_layers,N,Hout）的张量，包含输入序列的最终隐藏状态。

变量：

weight_ih_l[k-第kh层的可学习输入-隐藏权重(W_irlW_izlW_in),当k=0时，形状为(3*hidden_size, input_size)。否则，形状为(3*hidden_size,num_directions*hidden_size)

weight_hh_l[-第kh层的可学习隐藏-隐藏权重(W_hrW_hzW_hn),形状为(3 hidden_size, hidden_size)

bias_ih[-第k执层的可学习输入-隐藏偏差(b irb izb in),形状为(3 hidden_size)

bias_hh_l[k-第kh层的可学习隐藏-隐藏偏差(b hrlb_hzlb hn),形状为(3*hidden_.size)

```
rnn = nn.GRU(10, 20, 2)
input = torch.randn(5, 3, 10)
h0 = torch.randn(2, 3, 20)
output, hn = rnn(input, h0)
```