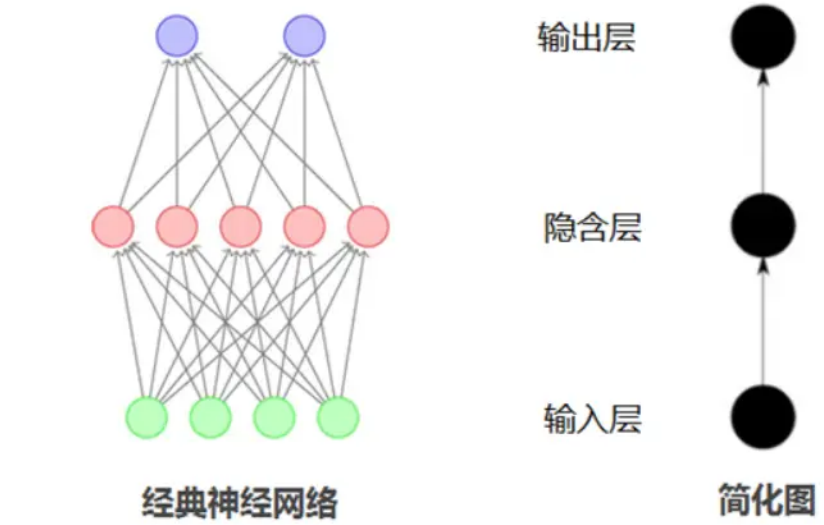
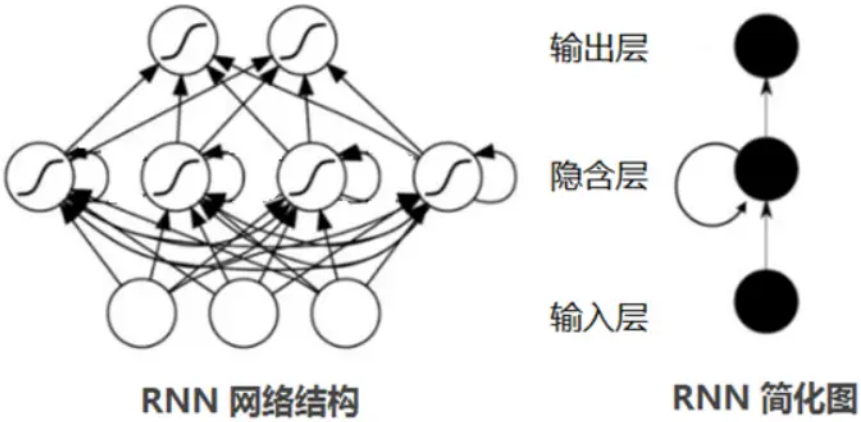
回顾和复习常见神经网络结构的底层原理————RNN（循环神经网络），这个只是听说过，没有具体的项目实践，在这里扫盲总结一下~

Day3：RNN

在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，**每层之间的节点是无连接的。**但是这种普通的神经网络对于很多问题却无能无力。例如，你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。





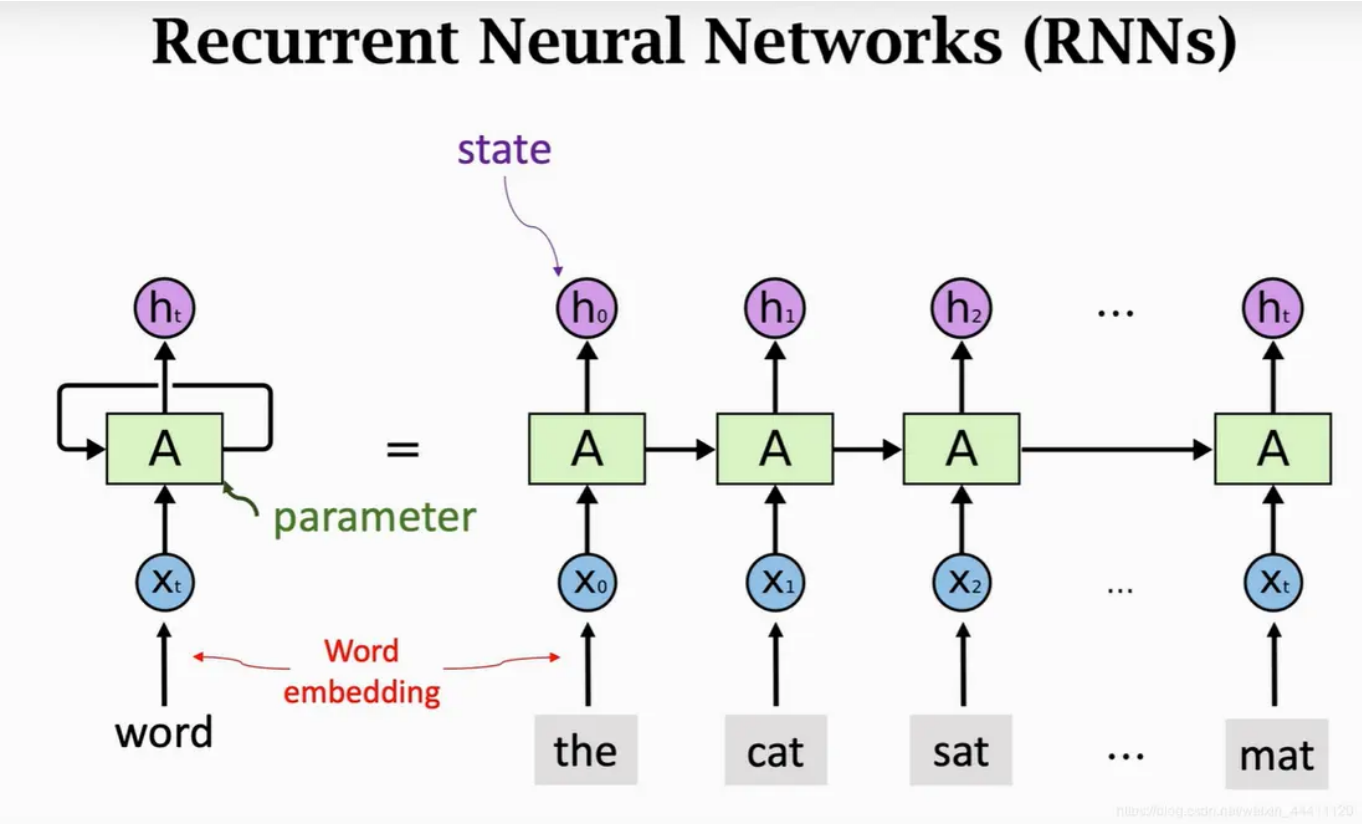
而RNN用于处理序列数据，一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。理论上，RNN能够对任何长度的序列数据进行处理。但是在实践中，为了降低复杂性往往假设当前的状态只与前面的几个状态相关。

```python
class SimpleRNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(SimpleRNN, self).__init__()
        # 定义 RNN 层
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
        # 定义全连接层
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # x: (batch_size, seq_len, input_size)
        out, _ = self.rnn(x)  # out: (batch_size, seq_len, hidden_size)
        # 取序列最后一个时间步的输出作为模型的输出
        out = out[:, -1, :]  # (batch_size, hidden_size)
        out = self.fc(out)  # 全连接层
        return out outturn out
```

- RNN的重要特点：**每一步的参数共享**

1. RNN的结构详解



- **输入层**：RNN能够接受一个输入序列（例如文字、股票价格、语音信号等）并将其传递到隐藏层。

- **隐藏层**：隐藏层之间存在循环连接，使得网络能够维护一个“记忆”状态，这一状态包含了过去的信息。这使得RNN能够理解序列中的上下文信息。

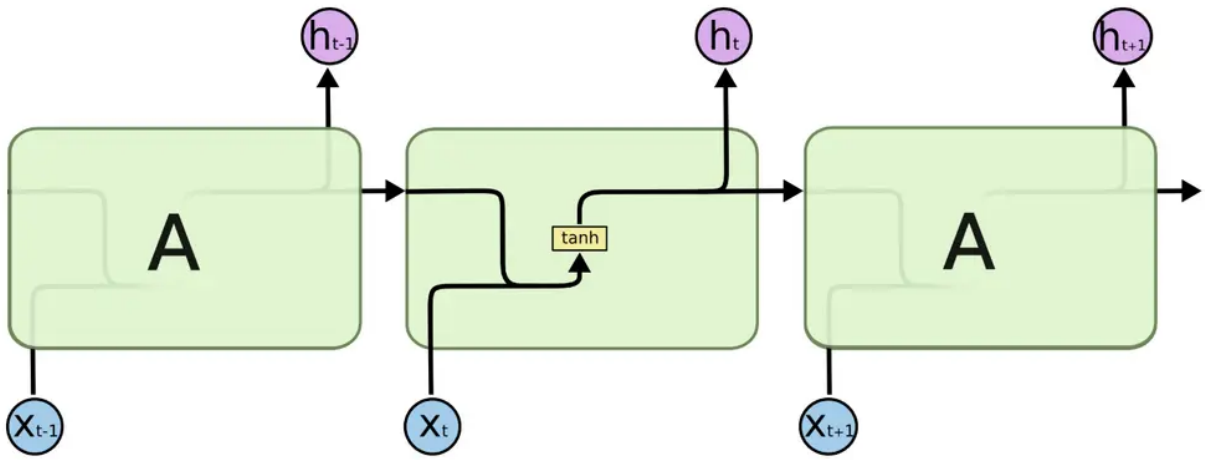
  > h_t，h_{t-1}分别表示当前和前一时刻的隐藏态（绿色的状态），W_{hh}，W_{xh}分别表示输入到隐藏状态的权重矩阵和隐藏状态到隐藏状态的权重矩阵，表示公式：$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b)$

- **输出层**：RNN可以有一个或多个输出，例如在序列生成任务中，每个时间步都会有一个输出。

  > W_{hy}隐藏状态到输出的权重矩阵，输出$y_t = W_{hy}h_t + b_y$

梯度问题：由于RNN的循环结构，在训练中可能会出现梯度消失或梯度爆炸的问题。

2. RNN的变种——LSTM



The repeating module in a standard RNN contains a single layer.

RNN什么信息它都存下来，因为它没有挑选的能力，而LSTM不一样，它会选择性的存储信息，因为它能力强，它有门控装置，它可以尽情的选择。