## A6.1 (5 分) 分析卷积神经网络中用 1×1 的卷积核的作用

解答：

1. 特征图通道变换：1x1 卷积可以改变特征图的通道数。它可以用来增加或减少特征图的深度。例如，如果输入特征图有 256 个通道，而我们希望输出特征图只有 64 个通道，那么可以使用 64 个 1x1 的卷积核来实现。

2. 计算参数的减少：相比于使用大的卷积核（如 3x3 或 5x5），1x1 卷积核可以大大减少模型的参数数量。这有助于构建更轻量级的网络，减少计算资源消耗和过拟合的风险。

3. 非线性增强：虽然 1x1 卷积本身是线性操作，但在卷积层之后通常会接一个非线性激活函数（如 ReLU）。这样，即使是小尺寸的卷积核也能增加网络的非线性表达能力。

4. 跨通道信息整合：1x1 卷积可以实现不同通道之间的信息整合。在传统的卷积操作中，每个卷积核只作用于单个通道，而 1x1 卷积可以将不同通道的信息混合在一起，从而实现特征融合。

5. 网络深度的增加：在不显著增加计算负担的前提下，使用 1x1 卷积可以增加网络的深度，有助于学习更复杂的特征表示。

6. 维度压缩和特征选择：通过调整 1x1 卷积核的数量，可以对输入特征图进行维度压缩，这在某种程度上起到了特征选择的作用，有助于提升网络的泛化能力。

## A6.2 (5 分) 计算函数 $y = \max(x\_1, \cdots, x\_D)$ 和函数 $y = argmax(x\_1, \cdots, x\_D)$ 的梯度。

解答：

1. 函数 $y = \max(x\_1, \ldots, x\_D)$ 的梯度：

这个函数返回所有输入中的最大值。在梯度的计算中，只有最大值对应的输入会对输出有影响，因此该输入的梯度为 1，其余输入的梯度为 0。

$$\frac{\partial y}{\partial x_i} = \begin{cases} 1, & \text{if } x_i = \max(x_1, \ldots, x_D) \\ 0, & \text{otherwise} \end{cases}$$

2. 函数 y=argmax(x1,⋯,xD) 的梯度：

y = argmax(x1,⋯ ,xD)表示的是使函数 y = max(x1,⋯ ,xD)取得最大值的 x 的索引，也就是说，y 是一个整数，它的值是 x 中最大元素的位置。

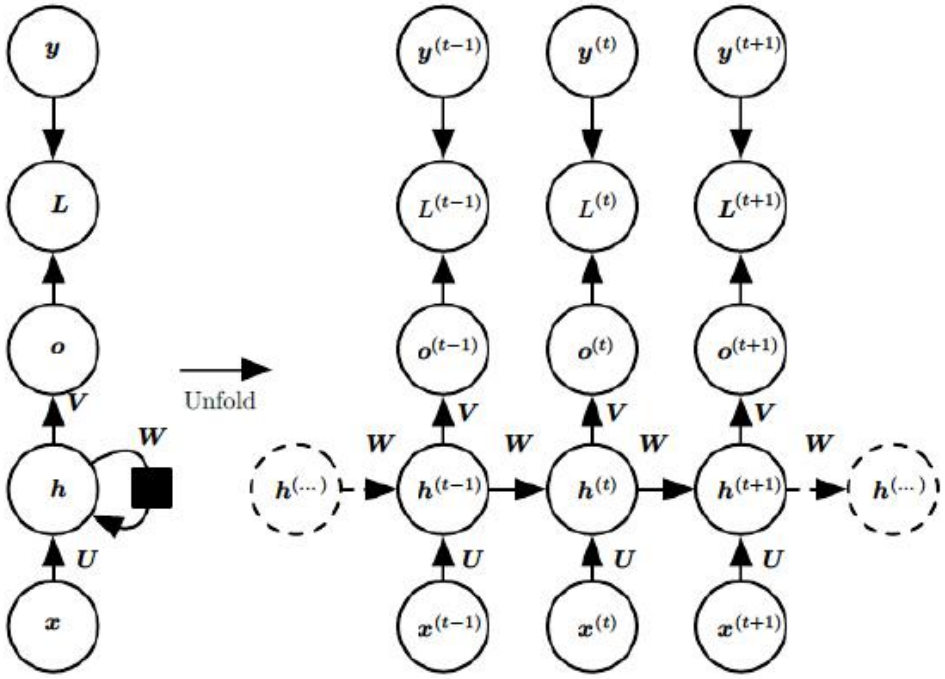
由于 y 是一个离散的函数，它的值只能在{1, 2, ⋯, D}中取，因此，当 x 中有多个元素达到最大值时，y 是不连续的，也就是说，y 在这些点上是不可导的，

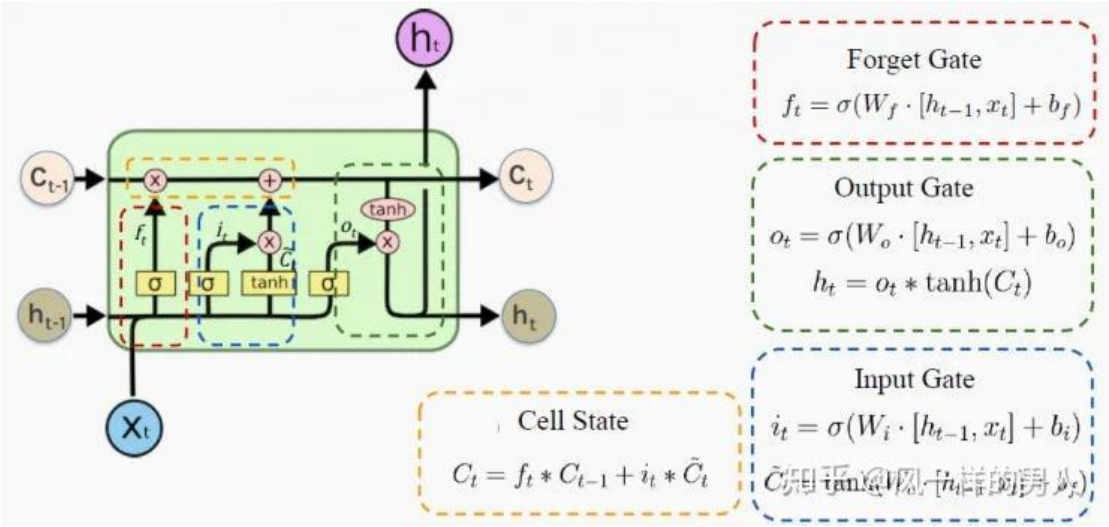
梯度是未定义的。例如，如果 x = (1, 2, 2, 1)，那么 y 可以是 2 或者 3，这取决于 argmax 的实现方式，但无论如何，y 在 x 的这个值上都是不可导的。
当 x 中只有一个元素达到最大值时，y 是一个常数，它的值是这个元素的位置，因此，y 在 x 的这个值上是可导的，梯度是零向量。例如，如果 x = (1, 2, 3, 1)，那么 y = 3，这是一个常数，所以 y 对 x 的任何分量的偏导数都是 0。

$$\frac{\partial y}{\partial x_i} = \begin{cases} 0, & \text{if } x_i \neq \max(x_1, \ldots, x_D) \\ \text{undefined}, & \text{if } x_i = \max(x_1, \ldots, x_D) \end{cases}$$

## A6.3 (5 分) 推导 LSTM 网络中参数的梯度，并分析其避免梯度消失的效果。

解答：






LSTM 网络的反向传播过程需要计算损失函数 L 对各个参数的梯度
LSTM 网络由四个门控单元组成：输入门、遗忘门、输出门和单元状态。每个门控单元都有一个权重矩阵和一个偏置向量，分别记为 W 和 b 。LSTM 网络的参数包括：
输入门的权重矩阵 $W_i$ 和偏置向量 $b_i$

遗忘门的权重矩阵 $W_f$ 和偏置向量 $b_f$

输出门的权重矩阵 $W_o$ 和偏置向量 $b_o$
单元状态的权重矩阵 $W_c$ 和偏置向量 $b_c$

对于隐藏层的参数，需要考虑每个门控单元的影响，得到：
输入门参数的梯度：
输入门权重梯度：$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial i_t}\frac{\partial i_t}{\partial W_i}$
输入门偏置梯度：$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial i_t}\frac{\partial i_t}{\partial b_i}$

遗忘门参数的梯度：
遗忘门权重梯度：$\frac{\partial L}{\partial W_f} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial W_f}$
遗忘门偏置梯度：$\frac{\partial L}{\partial b_f} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial b_f}$

输出门参数的梯度：
输出门权重梯度：$\frac{\partial L}{\partial W_o} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial o_t}\frac{\partial o_t}{\partial W_o}$
输出门偏置梯度：$\frac{\partial L}{\partial b_f} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial b_f}$

单元状态参数的梯度：

单元状态权重梯度：$\frac{\partial L}{\partial W_c} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial C_t}\frac{\partial C_t}{\partial \tilde{C}_t}\frac{\partial \tilde{C}_t}{\partial W_c}$

单元状态偏置梯度：$\frac{\partial L}{\partial b_c} = \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial C_t}\frac{\partial C_t}{\partial \tilde{C}_t}\frac{\partial C_t}{\partial b_c}$

LSTM 网络避免梯度消失的效果主要基于其独特的结构设计，这包括：
1.门控机制：
遗忘门（Forget Gate）：允许网络“忘记”不必要的信息，这有助于调节梯度流，并且防止梯度在反向传播时由于累乘效应而过小。
输入门（Input Gate）和输出门（Output Gate）：它们控制新信息的输入和当前单元状态的输出。这种控制能力意味着网络可以保护和传递重要的梯度，不让它们在多个时间步中消失。

常量误差轭（Constant Error Carousel, CEC）：
LSTM 的核心是它的单元状态$C_t$，它的设计允许误差以不变的形式在单元之间循环。在理想情况下，这意味着梯度可以在很长的序列中传播而不衰减，这对于学习长期依赖至关重要。

2.线性依赖：

在更新单元状态时，部分更新是线性的 $(C_t = f_t * C_{t-1} + i_t * \tilde{C}_t)$。线性操作的梯度是常数，这有助于减少在反向传播时的梯度衰减。

3.激活函数的选择：
LSTM 使用 sigmoid 和 tanh 函数，它们的导数在输入接近 0 时较大，在输入远离 0 时接近 0。这意味着在门控信号较强时，梯度可以较好地流动，而在信号较弱时减少其影响，这有助于稳定梯度。

## A6.4 (5 分) 当将自注意力模型作为神经网络的一层使用时，分析它和卷积层以及循环层在建模长距离依赖关系的效率和计算复杂度方面的差异

解答：

1.建模长距离依赖关系的能力：

自注意力层：自注意力机制能够直接计算序列中任何两个位置之间的关系，使其在处理长距离依赖关系时非常有效。它可以在单个操作中捕捉序列内的远程依赖，而不受序列长度的限制。

卷积层：卷积网络通过局部感受野捕捉空间上的特征，对于长距离依赖的建模较为依赖层数和感受野的大小。要捕捉更长距离的依赖，需要多层堆叠或使用较大的卷积核。

循环层：RNN（尤其是 LSTM 和 GRU）设计用于处理序列数据，理论上可以捕捉长序列中的长期依赖，但在实践中，由于梯度消失或爆炸问题，它们在捕捉非常长距离依赖方面可能会遇到困难。

2.计算复杂度：

自注意力层：自注意力层的计算复杂度主要取决于序列的长度，因为它需要计算序列中每个元素对的关系。对于长度为 NN 的序列，其复杂度大约为 O(N2)O(N2)。虽然这对于短序列来说是可管理的，但对于非常长的序列，这可能变得非常昂贵。

卷积层：卷积层的计算复杂度通常与输入大小、卷积核大小和层数相关。对于固定大小的卷积核，其复杂度通常低于自注意力层，尤其是在处理长序列时。

循环层：RNN 的计算复杂度与序列长度线性相关，对于每个时间步，其复杂度大致为 O(1)O(1)。但 RNN 的一个主要缺点是它的顺序性质，这限制了并行处理能力。

总的来说，自注意力层在捕捉长距离依赖方面更为高效，但在处理非常长的序列时，其计算复杂度可能是一个问题。相比之下，卷积层和循环层在处理长序列时可能更加高效，尽管它们在捕捉长距离依赖方面可能不如自注意力层强大。