第一章　循环神经网络

在前馈神经网络中，信息的传递是单向的，这种限制虽然使得网络变得更容易学习，但在一定程度上也减弱了神经网络模型的能力。在生物神经网络中，神经元之间的连接关系要复杂的多。前馈神经网络可以看着是一个复杂的函数，每次输入都是独立的，即网络的输出只依赖于当前的输入。但是在很多现实任务中，不同时刻的输入可以相互影响，比如视频、语音、文本等序列结构数据。某个时刻的输出可能和前面时刻的输入相关。此外，这些序列结构数据的长度一般是不固定的。而前馈神经网络要求输入和输出的维数都是固定的，不能任意改变。因此，当处理这一类问题时，就需要一种能力更强的模型。

在前馈神经网络模型中，连接存在层与层之间，每层的节点之间是无连接的。**循环神经网络**（Recurrent Neural Networks，RNN）通过使用带自反馈的神经元，能够处理任意长度的序列。循环神经网络比前馈神经网络更加符合生物神经网络的结构。循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上。

RNN 也经常被翻译为**递归神经网络**。这里为了区别与另外一种递归神经网络（Recursive Neural Networks），我们称为**循环神经网络**。

给定一个输入序列 $\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$，循环神经网络通过下面公式更新带反馈边的隐藏层的**活性值** $\mathbf{h}_t$：

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases} \tag{1.1}$$

从数学上讲，公式1.1可以看成一个动态系统。**动态系统**是指系统的状态按照一定的规律随时间变化的系统。因此，活性值 $\mathbf{h}_t$ 在很多文献上也称为**状态**或**隐状态**。但这里的状态是数学上的概念，区别与我们在前馈网络中定义的神经元的状态。理论上循环神经网络可以近似任意的动态系统。图1.1给出了循环神经网络的示例。

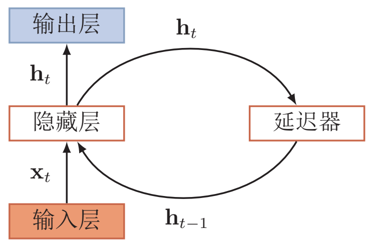


图 1.1: 循环神经网络

循环神经网络的参数训练可以通过**时序反向传播**（Backpropagation Through Time，BPTT）算法 [Werbos, 1990] 来学习。时序反向传播即按照时间的逆序将错误信息一步步地往前传递。这样，当输入序列比较长时，会存在梯度爆炸和消失问题 [Bengio et al., 1994, Hochreiter and Schmidhuber, 1997, Hochreiter et al., 2001]，也称为长期依赖问题。为了解决这个问题，人们对循环神经网络进行了很多的改进，其中最有效的一个改进版本是长短期记忆神经网络。

在本章中，我们先介绍循环神经网络的基本定义以及梯度计算，然后介绍两种扩展模型：长短期记忆神经网络和 GRU 网络。

1.1 简单循环网络

我们先来看一个非常简单的循环神经网络，叫**简单循环网络**（Simple Recurrent Network，SRN）[Elman, 1990]。

假设时刻 t 时，输入为 $\mathbf{x}_t$，隐层状态（隐层神经元活性）为 $\mathbf{h}_t$。$\mathbf{h}_t$ 不仅和当前时刻的输入相关，也和上一个时刻的隐层状态相关。

一般我们使用如下函数：

$$\mathbf{z}_t = U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}, \tag{1.2}$$

$$\mathbf{h}_t = f(\mathbf{z}_t) = f(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}), \tag{1.3}$$

其中 $\mathbf{z}_t$ 为神经元的净输入，$f(\cdot)$ 是非线性激活函数，通常为 logistic 函数或 tanh 函数。

图1.2给出了按时间展开的循环神经网络。如果我们把每个时刻的状态都看作是前馈神经网络的一层的话，循环神经网络可以看作是在时间维度上权值共

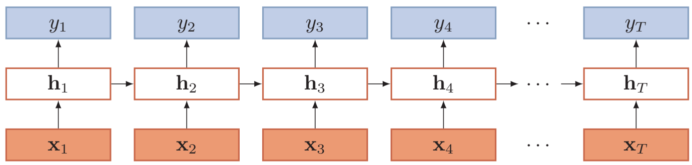


图 1.2: 按时间展开的循环神经网络

享的神经网络。

1.2 应用到机器学习

循环神经网络可以应用到很多不同类型的机器学习任务。根据这些任务的特点可以分为以下几种模式：

- 序列到类别模式：这种模式就是序列数据的分类问题。输入为序列，输出为类别。比如在文本分类中，输入数据为单词的序列，输出为该文本的类别。
- 序列到序列模式：这类任务的输入和输出都为序列。具体又可以分为两种情况：
  - 同步的序列到序列模式：这种模式就是机器学习中的**序列标注**（Sequence Labeling）任务，即每一时刻都有输入和输出，输入序列和输出序列的长度相同。比如**词性标注**（Part-of-Speech Tagging）中，每一个单词都需要标注其对应的词性标签。

参见第??节，第??页。

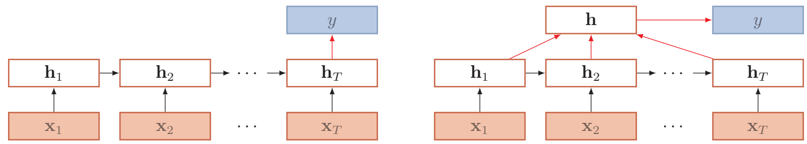
  - 异步的序列到序列模式：这种模式也称为**编码器-解码器**（Encoder-Decoder）模型，即输入和输出不需要有严格的对应关系，也不需要保持相同的长度。比如在机器翻译中，输入为源语言的单词序列，输出为目标语言的单词序列。

参见第??节，第??页。

下面我们分别来看下这几种应用模式。

1.2.1 序列到类别模式

首先，我们来看下序列到类别的应用模式。假设一个样本 $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$ 为一个长度为 T 的序列，输出为一个类别 $y \in \{1, \cdots, C\}$。我们可以将样本 $\mathbf{x}$ 按



(a) 正常模式 (b) 按时间进行平均采样模式

图 1.3: 序列到类别的应用模式

不同时刻输入到循环神经网络中，并得到不同时刻的隐藏状态 $\mathbf{h}_1, \cdots, \mathbf{h}_T$。我们可以将 $\mathbf{h}_T$ 看作整个序列的最终表示（或特征），并输入给分类器 $g(\cdot)$，

$$\hat{y} = g(\mathbf{h}_T), \tag{1.4}$$

这里 $g(\cdot)$ 可以是简单的线性分类器（比如 Logistic 回归）或复杂的分类器（比如多层前馈神经网络）。

除了将最后时刻的隐藏状态作为序列表示（如图1.3a）之外，我们还可以对整个序列的所有隐藏状态进行平均，并用这个平均状态来作为整个序列的表示（如图1.3b）。

$$\hat{y} = g(\frac{1}{T}\sum_{t=1}^{T}\mathbf{h}_t). \tag{1.5}$$

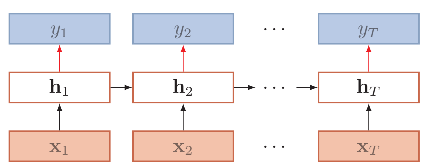


图 1.4: 同步的序列到序列模式

1.2.2 同步的序列到序列模式

在同步的序列到序列模式中（如图1.4所示），输入为一个长度为 T 的序列 $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$，输出为序列 $\mathbf{y} = (y_1, \cdots, y_T)$。样本 $\mathbf{x}$ 按不同时刻输入到循环神经网络中，并得到不同时刻的隐状态 $\mathbf{h}_1, \cdots, \mathbf{h}_T$。每个时刻的隐状态 $\mathbf{h}_t$ 代

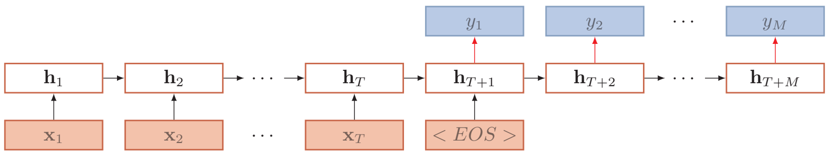


图 1.5: 异步的序列到序列模式

表了当前时刻和历史的信息，并输入给分类器 $g(\cdot)$ 得到当前时刻的标签 $\hat{y}_t$。

$$\hat{y}_t = g(\mathbf{h}_t), \forall i \in [1, T]. \tag{1.6}$$

1.2.3 异步的序列到序列模式

在异步的序列到序列模式中（如图1.5所示），输入为一个长度为 T 的序列 $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$，输出为长度为 M 的序列 $\mathbf{y} = (y_1, \cdots, y_M)$。异步的序列到序列模式的输入和输出不需要有严格的对应关系，也不需要保持相同的长度。因此，异步的序列到序列模式经常通过先编码后解码的方式来实现。先将样本 $\mathbf{x}$ 按不同时刻输入到一个循环神经网络（编码器）中，并得到其编码 $\mathbf{h}_T$。然后在使用另一个循环神经网络（解码器）中，得到输出序列。

1.3 参数学习

循环神经网络的参数学习可以通过**随时间进行反向传播**（Backpropagation Through Time，BPTT）算法 [Werbos, 1990]。

以随机梯度下降为例，给定一个训练样本 $(\mathbf{x}, \mathbf{y})$，其中 $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$ 为长度是 T 的输入序列，$\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_T)$ 是长度为 T 的标签序列。即在每个时刻 t，都有一个监督信息 $\mathbf{y}_t$，我们定义时刻 t 的损失函数为

$$\mathcal{L}_t = \mathcal{L}(\mathbf{y}_t, f(\mathbf{h}_{t-1}, \mathbf{x}_t)), \tag{1.7}$$

这里 $\mathcal{L}$ 为可微分的损失函数，比如交叉熵。那么整个序列上损失函数为

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t. \tag{1.8}$$

整个序列的损失函数 $\mathcal{L}$ 关于参数 U 的梯度为

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial U}, \tag{1.9}$$

即每个时刻损失 $\mathcal{L}_t$ 对参数 U 的偏导数之和。

计算偏导数 $\frac{\partial \mathcal{L}_t}{\partial U}$ 先来计算公式 (1.9) 中第 t 时刻损失对参数 U 的偏导数 $\frac{\partial \mathcal{L}_t}{\partial U}$。

因为参数 U 和隐藏层在每个时刻 $k(1 \leq k \leq t)$ 的净输入 $\mathbf{z}_k = U\mathbf{h}_{k-1} + W\mathbf{x}_k + \mathbf{b}$ 有关，因此第 t 时刻损失的损失函数 $\mathcal{L}_t$ 关于参数 U_{ij} 的梯度为：

链式法则参见公式 (??)，第 ?? 页。

$$\frac{\partial \mathcal{L}_t}{\partial U_{ij}} = \sum_{t=1}^{T} \sum_{k=1}^{t} tr\left((\frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k})^{\mathrm{T}} \frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} \right), \tag{1.10}$$

其中 $\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}}$ 表示“直接”偏导数，即公式 $\mathbf{z}_k = U\mathbf{h}_{k-1} + W\mathbf{x}_k + \mathbf{b}$ 中保持 $\mathbf{h}_{k-1}$ 不变，对 U_{ij} 进行求偏导数，得到

$$\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} = \begin{bmatrix} 0 \\ \vdots \\ \boxed{[\mathbf{h}_k]_j} \\ \vdots \\ 0 \end{bmatrix}, \tag{1.11}$$

其中 $[\mathbf{h}_k]_j$ 为第 k 时刻隐状态的第 j 维。$\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}}$ 除了第 i 行外，其余都为 0。

定义 $\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k}$ 为第 t 时刻的损失对第 k 时刻隐藏神经层的净输入 $\mathbf{z}_k$ 的导数，则

$$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \tag{1.12}$$

$$= \frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_{k+1}} \tag{1.13}$$

$$= \mathbf{diag}(f'(\mathbf{z}_k)) U^{\mathrm{T}} \delta_{t,k+1}. \tag{1.14}$$

将公式 (1.14) 和 (1.11) 代入公式 (1.10) 得到

$$\frac{\partial \mathcal{L}_t}{\partial U_{ij}} = \sum_{k=1}^{t} [\delta_{t,k}]_i [\mathbf{h}_k]_j. \tag{1.15}$$

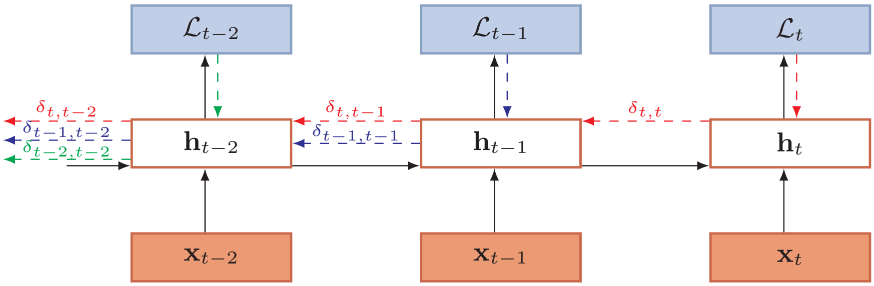


图 1.6: 随时间进行反向传播算法示例。

将上式写成矩阵形式为

$$\frac{\partial \mathcal{L}_t}{\partial U} = \sum_{k=1}^{t} \delta_{t,k} \mathbf{h}_k^{\mathrm{T}}. \tag{1.16}$$

图1.6给出了误差项随时间进行反向传播算法的示例。

参数梯度 将公式 (1.16) 代入到将公式 (1.9) 得到整个序列的损失函数 $\mathcal{L}$ 关于参数 U 的梯度

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} \mathbf{h}_k^{\mathrm{T}}. \tag{1.17}$$

同理可得，$\mathcal{L}$ 关于权重 W 和偏置 $\mathbf{b}$ 的梯度为

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} \mathbf{x}_k^{\mathrm{T}}, \tag{1.18}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k}. \tag{1.19}$$

长期依赖问题 将公式 (1.14) 展开得到

$$\delta_{t,k} = \prod_{i=k}^{t-1} \Big(\mathbf{diag}(f'(\mathbf{z}_i)) U^{\mathrm{T}} \Big) \delta_{t,t}. \tag{1.20}$$

我们定义 $\gamma = \|\,\mathbf{diag}(f'(\mathbf{z}_i))U^{\mathrm{T}}\|$，则

$$\delta_{t,k} = \gamma^{t-k}\delta_{t,t}. \tag{1.21}$$

若 $\gamma > 1$，当 $t-k \to \infty$ 时，$\gamma^{t-k} \to \infty$，会造成系统不稳定，称为**梯度爆炸问题**（Gradient Exploding Problem）；相反，若 $\gamma < 1$，当 $t-k \to \infty$ 时，$\gamma^{t-k} \to 0$，会出现和深度前馈神经网络类似的**梯度消失问题**（Gradient Vanishing Problem）。

要注意的是，在循环神经网络中的梯度消失不是说 $\frac{\partial \mathcal{L}_{\sqcup}}{\partial U}$ 的梯度消失了，而是 $\frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_k}$ 的梯度消失了（当 $t-k$ 比较大时）。也就是说，参数 U 的更新主要靠当前时刻 k 的几个相邻状态 h_k 来更新，长距离的状态对 U 没有影响。

由于循环神经网络经常使用非线性激活函数为 logistic 函数或 tanh 函数作为非线性激活函数，其导数值都小于1；并且权重矩阵 $\|U\|$ 也不会太大，因此如果时间间隔 $t-k$ 过大，$\delta_{t,k}$ 会趋向于0，因此经常会出现梯度消失问题。

虽然简单循环网络从理论上可以建立长时间间隔的状态之间的依赖关系，但是由于梯度爆炸或消失问题，实际上只能学习到短期的依赖关系。这就是所谓的**长期依赖问题**（Long-Term Dependencies Problem）。

1.3.1 改进方案

为了避免梯度爆炸或消失问题，一种最直接的方式就是选取合适的参数，同时使用非饱和的激活函数，尽量使得 $\mathbf{diag}(f'(\mathbf{z}_i))U^{\mathrm{T}} \approx 1$，这种方式需要足够的人工调参经验，限制了模型的广泛应用。比较有效的方式通过改进模型或优化方法来缓解循环网络的梯度爆炸和梯度消失问题。

梯度截断参见算法(??)，第??页。

梯度爆炸 一般而言，循环网络的梯度爆炸问题比较容易解决，一般通过权重衰减或梯度截断来避免。

权重衰减是通过给参数增加 L_1 或 L_2 范式的正则化项来限制参数的取值范围，从而使得 $\gamma \le 1$。梯度截断是另一种有效的启发式方法，当梯度的模大于一定阈值时，就将它截断成为一个较小的数。

梯度消失　梯度消失是循环网络的主要问题。除了使用一些优化技巧外，更有效的方式就是改变模型，比如让 $U=1$，同时使用 $f'(\mathbf{z}_i)=1$，即

$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t;\theta), \tag{1.22}$$

其中 $g(\cdot)$ 是一个非线性函数，θ 为参数。

公式 (1.22) 中，$\mathbf{h}_t$ 和 $\mathbf{h}_{t-1}$ 之间为线性依赖关系，且权重系数为1，这样就不存在梯度爆炸或消失问题。但是，这种改变也丢失了神经元在反馈边上的非线性激活的性质，因此也降低了模型的表示能力。

为了避免这个缺点，我们可以采用一个更加有效的改进策略：

$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t, \mathbf{h}_{t-1};\theta), \tag{1.23}$$

这样 $\mathbf{h}_t$ 和 $\mathbf{h}_{t-1}$ 之间为既有线性关系，也有非线性关系，在一定程度上可以缓解梯度消失问题。但这种改进依然有一个问题就是**记忆容量**（memory capacity）。随着 $\mathbf{h}_t$ 不断累积存储新的输入信息，会发生饱和现象。假设 $g(\cdot)$ 为 logistic 函数，则随着时间 t 的增长，$\mathbf{h}_t$ 会变得越来越大，从而导致 $\mathbf{h}$ 变得饱和。也就是说，隐藏状态 $\mathbf{h}_t$ 可以存储的信息是有限的，随着记忆单元存储的内容越来越多，其丢失的信息也越来越多。

为了解决容量问题，可以有两种方法。一种是增加一些额外的存储单元：外部记忆；另一种是进行选择性的遗忘，同时也进行有选择的更新。增加外部记忆的方法将在第**??**章中介绍，本章主要介绍后一种方法。

1.4　基于门控制的循环神经网络

为了解决上节中提到的记忆容量问题，一种非常好的解决方案是引入门控制Hochreiter and Schmidhuber [1997] 来控制信息的累积速度，包括有选择地加入新的信息，并有选择地遗忘之前累积的信息。这一类网络可以称为**基于门控制的循环神经网络**（Gated RNN）。本节中，主要介绍两种基于门控制的循环神经网络：LSTM 网络和 GRU 网络。

1.4.1　LSTM 网络

长短期记忆神经网络（Long Short-Term Memory Neural Network，LSTM 网络）[Gers et al., 2000, Hochreiter and Schmidhuber, 1997] 是循环神经网络

的一个变体，可以有效地解决简单循环神经网络的梯度爆炸或消失问题。

在公式 (1.23) 的基础上，LSTM 网络主要改进在以下两个方面：

新的内部状态 LSTM 网络引入一个新的内部状态（internal state）$\mathbf{c}_t$ 专门进行线性的循环信息传递，同时（非线性）输出信息给隐藏层的外部状态 $\mathbf{h}_t$。

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{1.24}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{1.25}$$

其中 $\mathbf{f}_t$，$\mathbf{i}_t$ 和 $\mathbf{o}_t$ 为三个门（gate）来控制信息传递的路径；$\odot$ 为向量元素乘积；$\mathbf{c}_{t-1}$ 为上一时刻的记忆单元；$\tilde{\mathbf{c}}_t$ 是通过非线性函数得到候选状态，

$$\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c). \tag{1.26}$$

公式 (1.26)∼(1.29) 中的 $W_*, U_*, \mathbf{b}_*$ 为可学习的网络参数，其中 $* \in \{i, f, o, c\}$。

在每个时刻 t，LSTM 网络的内部状态 $\mathbf{c}_t$ 记录了到当前时刻为止的历史信息。

门机制 LSTM 网络引入**门机制**（Gating Mechanism）来控制信息传递的路径。公式 (1.24) 和 (1.24) 中三个“门”分别为输入门 $\mathbf{i}_t$, 遗忘门 $\mathbf{f}_t$ 和输出门 $\mathbf{o}_t$，

在数字电路中，**门**（gate）为一个二值变量 $\{0, 1\}$，0 代表关闭状态，不许任何信息通过；1 代表开放状态，允许所有信息通过。LSTM 网络中的“门”是一种“软”门，取值在 $(0, 1)$ 之间，表示以一定的比例运行信息通过。LSTM 网络中三个门的作用为

- 遗忘门 $\mathbf{f}_t$ 控制上一个时刻的内部状态 $\mathbf{c}_{t-1}$ 需要遗忘多少信息。
- 输入门 $\mathbf{i}_t$ 控制当前时刻的候选状态 $\tilde{\mathbf{c}}_t$ 有多少信息需要保存。
- 输出门 $\mathbf{o}_t$ 控制当前时刻的内部状态 $\mathbf{c}_t$ 有多少信息需要输出给外部状态 $\mathbf{h}_t$。

当 $\mathbf{f}_t = 0, \mathbf{i}_t = 1$ 时，记忆单元将历史信息清空，并将候选状态向量 $\tilde{\mathbf{c}}_t$ 写入。但此时记忆单元 $\mathbf{c}_t$ 依然和上一时刻的历史信息相关。当 $\mathbf{f}_t = 1, \mathbf{i}_t = 0$ 时，记忆单元将复制上一时刻的内容，不写入新的信息。

三个门的计算方式为：

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{1.27}$$

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{1.28}$$

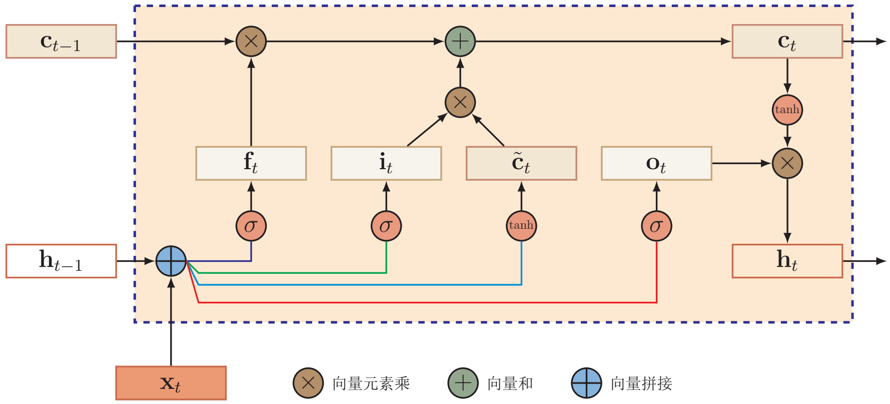


图 1.7: LSTM 循环单元结构。虚线框内为一个循环单元。

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{1.29}$$

其中 $\sigma(\cdot)$ 为 logistic 函数，其输出区间为 $(0,1)$，$\mathbf{x}_t$ 为当前时刻的输入，$\mathbf{h}_{t-1}$ 为上一时刻的外部状态。

图1.7给出了 LSTM 网络的循环单元结构，其计算过程为：（1）首先利用上一时刻的外部状态 $\mathbf{h}_{t-1}$ 和当前时刻的输入 $\mathbf{x}_t$，计算出三个门，以及候选状态 $\tilde{\mathbf{c}}_t$；（2）结合遗忘门 $\mathbf{f}_t$ 和输入门 $\mathbf{i}_t$ 来更新记忆单元 $\mathbf{c}_t$；（3）结合输出门 $\mathbf{o}_t$，将内部状态的信息传递给外部状态 $\mathbf{h}_t$。

通过 LSTM 循环单元，整个网络可以建立长距离的时序依赖关系。因此，对于输入和输出之间的信息传递路径上有较长的时间间隔时，可以选择 LSTM 网络，也可以利用 LSTM 网络的思想来改造一般的前馈网络，比如**高速网络**（Highway Network）。

公式 (1.24)~(1.29) 可以简洁地描述为

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right), \tag{1.30}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{1.31}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{1.32}$$

其中 $\mathbf{x}_t \in \mathbb{R}^e$ 为当前时刻的输入，$W \in \mathbb{R}^{4d\times(d+e)}$ 和 $\mathbf{b} \in \mathbb{R}^{4d}$ 为网络参数。

记忆 循环神经网络中的隐状态 $\mathbf{h}$ 存储了历史信息，可以看作是一种**记忆**（Memory）。在简单循环网络中，隐状态每个时刻都会被重写，因此可以看作是一种**短期记忆**（Short-term Memory）。在神经网络中，还有一种**长期记忆**（Long-term Memory）。长期记忆可以看作是网络参数，隐含了从训练数据中学到的经验，并更新周期要远远慢于短期记忆。而在 LSTM 网络中，记忆单元可以在某个时刻捕捉到某个关键信息，并有能力将此关键信息保存一定的时间间隔。记忆单元中保存信息的生命周期要长于短期记忆，但又远远短于长期记忆，因此称为**长的短期记忆**（Long Short-Term Memory）。

一般在深度网络参数学习时，参数初始化的值一般都比较小。但是在训练 LSTM 网络时，过小的值会使得遗忘门的值比较小。这意味着前一时刻的信息大部分都丢失了，这样网络很难捕捉到长距离的依赖信息。并且相邻时间间隔的梯度会非常小，这会导致梯度弥散问题。因此遗忘的参数初始值一般都设得比较大，其偏置向量 $\mathbf{b}_f$ 设为 1 或 2.

1.4.2 LSTM 网络的各种变体

LSTM 网络有非常多的改进版本。本节中介绍几种在门机制上比较简单的改进模型。

目前主流的 LSTM 网络用的三个门来动态地控制内部状态的应该遗忘多少历史信息，输入多少新信息，以及输出多少信息。

无遗忘门的 LSTM 网络 Hochreiter and Schmidhuber [1997] 最早提出的 LSTM 网络是没有遗忘门的，其内部状态的更新为

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t. \tag{1.33}$$

如之前的分析，记忆单元 $\mathbf{c}$ 会不断增大。当输入序列的长度非常大时，记忆单元的容量会饱和，从而大大降低 LSTM 模型的性能。

peephole 连接 另外一种变体是三个门不但依赖于输入 $\mathbf{x}_t$ 和上一时刻的隐状态 $\mathbf{h}_{t-1}$，也依赖于上一个时刻的记忆单元 $\mathbf{c}_{t-1}$。

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{1.34}$$

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_f\mathbf{c}_{t-1} + \mathbf{b}_f), \tag{1.35}$$

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c}_t + \mathbf{b}_o), \tag{1.36}$$

其中 V_i，V_f 和 V_o 为对角阵形式的参数。

耦合输入门和遗忘门 LSTM 网络中的输入门和遗忘门有些互补关系，因此同时用两个门比较冗余。为了减少 LSTM 网络的计算复杂度，将这两门合并为一个门。令

$$\mathbf{f}_t = \mathbf{1} - \mathbf{i}_t. \tag{1.37}$$

这样，内部状态的更新方式为

$$\mathbf{c}_t = (\mathbf{1} - \mathbf{i}_t) \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t. \tag{1.38}$$

1.4.3 GRU 网络

GRU 网络（gated recurrent unit network）[Cho et al., 2014, Chung et al., 2014] 是一种比 LSTM 网络更加简单的循环神经网络。GRU 也是在公式 (1.23) 的基础上，引入门机制来控制信息更新的方式。

在 LSTM 网络中，输入门和遗忘门是互补关系，用两个门比较冗余。GRU 将输入门与和遗忘门合并成一个门：更新门（Update Gate）。同时，GRU 也不引入额外的记忆单元，直接在当前状态 $\mathbf{h}_t$ 和历史状态 $\mathbf{h}_{t-1}$ 之间引入线性依赖关系。

在 GRU 网络中，当前时刻的候选状态 $\tilde{\mathbf{h}}_t$ 为

计算候选状态 $\tilde{\mathbf{h}}_t$ 时，使用 tanh 激活函数是由于其导数有比较大的值域，缓解梯度消失问题。

$$\tilde{\mathbf{h}}_t = \tanh(W_c\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}), \tag{1.39}$$

其中 $\mathbf{r}_t \in [0, 1]$ 为**重置门**（reset gate），用来控制候选状态 $\tilde{\mathbf{h}}_t$ 的计算是否依赖上一时刻的状态 $\mathbf{h}_{t-1}$。

$$\mathbf{r}_t = \sigma(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1} + \mathbf{b}_r), \tag{1.40}$$

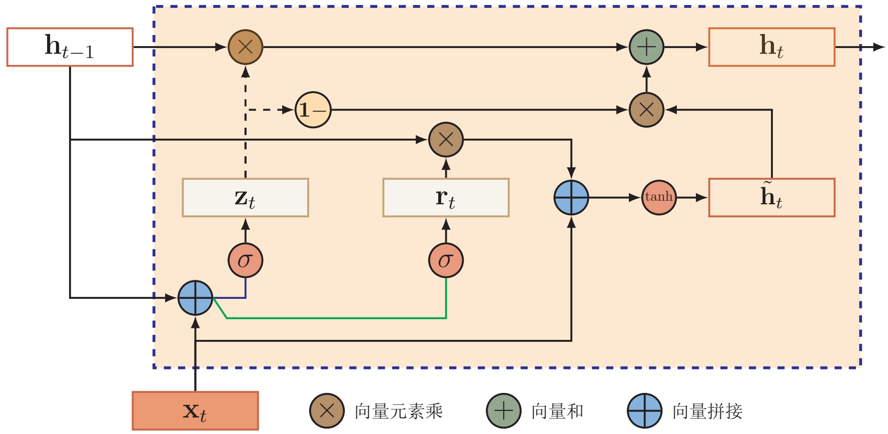


图 1.8: GRU 循环单元结构。虚线框内为一个循环单元。

当 $\mathbf{r}=0$ 时，候选状态 $\tilde{\mathbf{h}}_t=\tanh(W_c\mathbf{x}_t+\mathbf{b})$ 只和当前输入 $\mathbf{x}_t$ 相关，和历史状态无关。当 $\mathbf{r}=1$ 时，候选状态 $\tilde{\mathbf{h}}_t=\tanh(W_c\mathbf{x}_t+U\mathbf{h}_{t-1}+\mathbf{b})$ 和当前输入 $\mathbf{x}_t$ 和历史状态 $\mathbf{h}_{t-1}$ 相关，和简单循环网络一致。

GRU 网络的隐状态 $\mathbf{h}_t$ 更新方式为

$$\mathbf{h}_t=\mathbf{z}_t\odot\mathbf{h}_{t-1}+(1-\mathbf{z}_t)\odot\tilde{\mathbf{h}}_t, \tag{1.41}$$

其中 $\mathbf{z}\in(0,1)$ 为**更新门**（update gate），用来控制当前状态需要从历史状态中保留多少信息（不经过非线性变换），以及需要从候选状态中接受多少新信息。

$$\mathbf{z}_t=\sigma(\mathbf{W}_z\mathbf{x}_t+\mathbf{U}_z\mathbf{h}_{t-1}+\mathbf{b}_z), \tag{1.42}$$

$$\tilde{\mathbf{h}}_t=\tanh(\mathbf{W}_c\mathbf{x}_t+\mathbf{U}(\mathbf{r}_t\odot\mathbf{h}_{t-1})), \tag{1.43}$$

当 $\mathbf{z}=0$ 时，当前状态 $\mathbf{h}_t$ 和历史状态 $\mathbf{h}_{t-1}$ 之间为非线性函数。若同时有 $\mathbf{z}=0,\mathbf{r}=1$ 时，GRU 网络退化为简单循环网络；若同时有 $\mathbf{z}=0,\mathbf{r}=0$ 时，当前状态 $\mathbf{h}_t$ 只和当前输入 $\mathbf{x}_t$ 相关，和历史状态 $\mathbf{h}_{t-1}$ 无关。当 $\mathbf{z}=1$ 时，当前状态 $\mathbf{h}_t=\mathbf{h}_{t-1}$) 等于上一时刻状态 $\mathbf{h}_{t-1}$，和当前输入 $\mathbf{x}_t$ 无关。

图1.8给出了 GRU 循环单元结构。

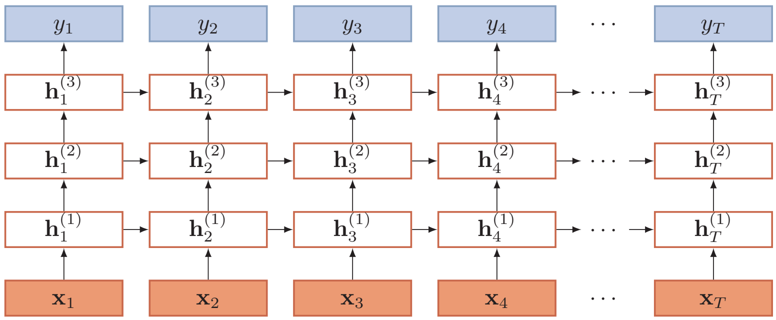


图 1.9: 按时间展开的堆叠循环神经网络

1.5 深层循环神经网络

循环神经网络的可以看作是既“深”又“浅”的网络。一方面来说，如果我们把循环网络按时间展开，不同时刻的状态之间存在非线性连接，循环网络可以看作是一个非常深的网络了。从另一方面来说，这个网络是非常浅的。任意两个相邻时刻的隐藏状态（$\mathbf{h}_{t-1} \to \mathbf{h}_t$），隐藏状态到输出（$\mathbf{h}_t \to \mathbf{y}_t$），以及输入到隐藏状态之间（$\mathbf{x}_t \to \mathbf{h}_t$）之间的转换只有一个非线性函数。

既然增加深度可以极大地增强前馈神经网络的能力，那么如何增加循环神经网络的深度呢？一种常见的做法是将多个循环网络堆叠起来，称为**堆叠循环神经网络**（Stacked Recurrent Neural Network，SRNN）。

图1.9给出了按时间展开的堆叠循环神经网络。第 l 层网络的输入是第 $l-1$ 层网络的输出。我们定义 $\mathbf{h}_t^{(l)}$ 为在时刻 t 时第 l 层的隐藏状态

$$\mathbf{h}_t^{(l)} = f(U^{(l)}\mathbf{h}_{t-1}^{(l)} + W^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{b}^{(l)}), \tag{1.44}$$

其中 $U^{(l)}$，$W^{(l)}$ 和 $\mathbf{b}^{(l)}$ 为权重矩阵和偏置向量。当 $l = 1$ 时，$\mathbf{h}_t^{(0)} = \mathbf{x}_t$。

1.6 双向循环神经网络

在有些任务中，一个时刻的输出不但和过去时刻的信息有关，也和后续时刻的信息有关。比如给定一个句子，其中一个词的词性由它的上下文决定，即包含左右两边的信息。因此，在这些任务中，我们可以增加一个按照时间的逆序来传递信息的网络层，来增强网络的能力。

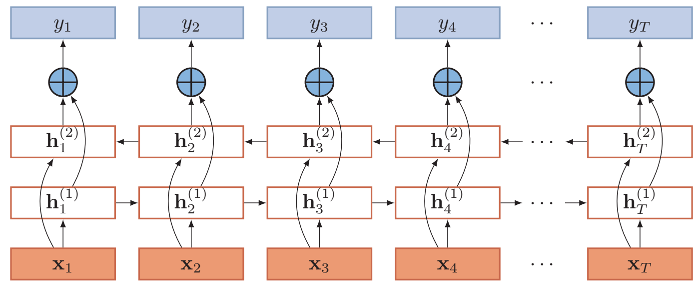


图 1.10: 按时间展开的双向循环神经网络

双向循环神经网络（Bidirectional Recurrent Neural Network，Bi-RNN）由两层循环神经网络组成，它们的输入相同，只是信息传递的方向不同。

假设第 1 层按时间顺序，第 2 层按时间逆序，在时刻 t 时的隐藏状态定义为 $\mathbf{h}_t^{(1)}$ 和 $\mathbf{h}_t^{(2)}$，则

$$\mathbf{h}_t^{(1)} = f(U^{(1)}\mathbf{h}_{t-1}^{(1)} + W^{(1)}\mathbf{x}_t + \mathbf{b}^{(1)}), \tag{1.45}$$

$$\mathbf{h}_t^{(2)} = f(U^{(2)}\mathbf{h}_{t+1}^{(2)} + W^{(2)}\mathbf{x}_t + \mathbf{b}^{(2)}), \tag{1.46}$$

$$\mathbf{h}_t = \mathbf{h}_t^{(1)} \oplus \mathbf{h}_t^{(2)}, \tag{1.47}$$

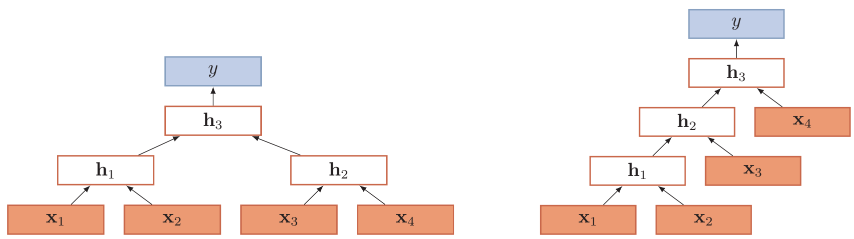
其中 $\oplus$ 为向量拼接操作。

图1.10给出了按时间展开的双向循环神经网络。

1.7 递归神经网络

递归神经网络（Recursive Neural Network，RNN 或 RecNN）是循环神经网络的扩展。如果把循环神经网络展开，可以看作是在时序维度上共享一个组合函数，而递归神经网络实在一个有向图无循环图上共享一个组合函数 [Pollack, 1990]。递归神经网络的一般结构为层次结构，如图1.11a所示。

以图1.11a中的结构为例，有三个隐藏层 $\mathbf{h}_1$、$\mathbf{h}_2$ 和 $\mathbf{h}_3$，其中 $\mathbf{h}_1$ 由两个输入 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 计算得到，$\mathbf{h}_2$ 由另外两个输入层 $\mathbf{x}_3$ 和 $\mathbf{x}_4$ 计算得到，$\mathbf{h}_3$ 由两个隐藏层



(a) 一般结构 (b) 退化结构

图 1.11: 递归神经网络

$\mathbf{h}_1$ 和 $\mathbf{h}_2$ 计算得到。三个隐藏层的组合函数是共享的，即

$$\mathbf{h}_1 = f(W \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \mathbf{b}), \tag{1.48}$$

$$\mathbf{h}_2 = f(W \begin{bmatrix} \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \mathbf{b}), \tag{1.49}$$

$$\mathbf{h}_3 = f(W \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \mathbf{b}), \tag{1.50}$$

其中 $f(\cdot)$ 是非线性激活函数，W 和 $\mathbf{b}$ 是可学习的参数。同样，输出层 y 可以为一个分类器，比如

$$y = g(W' \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \mathbf{b}'), \tag{1.51}$$

其中 $g(\cdot)$ 为分类器，W' 和 $\mathbf{b}'$ 为分类器的参数。

当递归神经网络的结构退化为线性序列结构时，递归神经网络就等价于简单循环神经网络。

参见习题(1-4)，第19页。

树结构的长短期记忆模型（tree-structured LSTM）[Tai et al., 2015, Zhu et al., 2015] 将 LSTM 模型的思想应用到树结构的网络中，来实现更灵活的组合函数。

递归神经网络主要用来建模自然语言句子的语义 [Socher et al., 2011, 2013]。给定一个句子的语法结构（一般为树状结构），可以使用递归神经网络来按照句

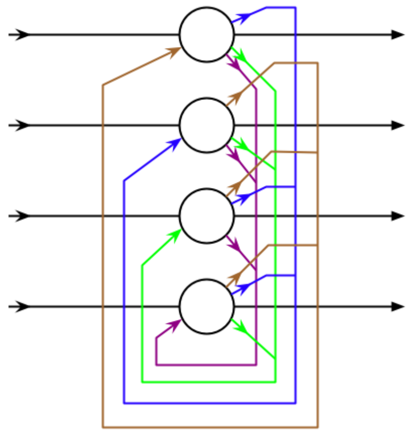


图 1.12: Hopfield 网络。图片来源于 https://en.wikipedia.org/wiki/Hopfield_-network。

法的组合关系来合成一个句子的语义。句子中每个短语成分可以在分成一些子成分，即每个短语的语义都可以由它的子成分语义组合而来，并进而合成整句的语义。

1.8 总结和深入阅读

Hopfield 网络（Hopfield network）是最早提出的循环神经网络模型 [Hopfield, 1982]。

除了循环神经网络，为了使得前馈神经网络能处理变长的序列数据，还有一种方法是使用**延时神经网络**（Time-Delay Neural Networks，TDNN）[Waibel et al., 1989]。

循环神经网络的一个难点是长期依赖问题 [Bengio et al., 1994, Hochreiter et al., 2001]。为了解决这个问题，人们对循环神经网络进行了很多的改进，其中最有效的一个改进版本是 *LSTM* 网络，LSTM 网络由 [Hochreiter and Schmidhuber, 1997] 提出，被 [Gers et al., 2000] 改进。当然还有一些其它方法，比如**时钟循环神经网络**（Clockwork RNN）[Koutnik et al., 2014]、以及引入注意力机制等。

注意力机制参见第??节，第??页。

LSTM 网络目前为止最成功的循环神经网络模型，成功应用在很多领域，比

如语音识别、机器翻译 [Sutskever et al., 2014]、语音模型以及文本生成。LSTM 网络通过引入线性连接来缓解长距离依赖问题。虽然LSTM取得了很大的成功，其结构的合理性一直受到广泛关注。人们不断有尝试对其进行改进来寻找最优结构，比如减少门的数量、提高并行能力等。

LSTM 的改进版本众多，其中使用比较广泛的有 GRU 网络 [Chung et al., 2014]。关于 LSTM 的分析可以参考文献 [Greff et al., 2017, Jozefowicz et al., 2015, Karpathy et al., 2015]。

LSTM 的线性连接以及门控制的想法十分有效。受此启发，残差网络 [He et al., 2016] 和高速网络 [Srivastava et al., 2015] 都通过引入线性连接来训练非常深的卷积网络。Grid LSTM 网络 [Kalchbrenner et al., 2015]

习题 1-1 计算公式 (1.18) 和公式 (1.19) 中的梯度。

习题 1-2 计算 LSTM 网络中参数的梯度，并分析其避免梯度消失的效果。

习题 1-3 计算 GRU 网络中参数的梯度，并分析其避免梯度消失的效果。

习题 1-4 证明当递归神经网络的结构退化为线性序列结构时，递归神经网络就等价于简单循环神经网络。

参考文献

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 2000.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79 (8):2554–2558, 1982.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1863–1871, 2014.

Jordan B Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105, 1990.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*, 2011.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10): 1550–1560, 1990.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. In *Proceedings of LCML*, pages 1604–1612, 2015.