AI 期中 Project

17341146 王程钥
17341225 朱绪思

1. 算法原理

1.1 卷积神经网络

卷积神经网络(CNN)是近年来人工智能，尤其是计算机视觉领域非常热门的一项技术，它拥有强大的特征学习能力。卷积神经网络学习到的特征在分类(classification)、检测(detection)、分割(segmentation)、跟踪(tracking)等计算机视觉任务中都取得了远胜于传统方法的性能，对计算机视觉的发展提到了很大的推进作用。本节将列举卷积神经网络中一些重要结构的算法原理。

1.1.1 卷积

卷积操作常用于信号处理与图像处理中，常见方法是按照公式(1)的思路把一个设计好的卷积核 w 在输入 x 的每个对应位置上对像素做加权求和以获取特征输出 y。传统方法采用人工设计的卷积核提取特征，并将特征放入分类器（如 SVM）中进行分类。

$$y_{i,j} = \sum_{u=1}^{n}\sum_{v=1}^{m} w_{u,v} * x_{i-u+1,j-v+1} \qquad (1)$$

1.1.2 卷积神经网络

不同于传统方法，卷积神经网络不采用人工设计的卷积核，而是将卷积核嵌入神经网络中，通过反向传播算法学习卷积层的参数。[1]最早将卷积神经网络用于图像分类任务。

卷积神经网络一般包含卷积层，池化层和全连接层。

卷积层包含若干个待学习的卷积核，用于数据集特征的提取。D 通道输入 x 通过卷积核 w 和激活函数σ对第 p 个通道的输出y^p的贡献可以用公式(2)表示。

$$y^p = \sigma(\sum_{d=1}^{D} w^{p,d} * x^d + b^p) \qquad (2)$$

卷积层输出的维度很高，为了降低数据维度，对特征进行筛选并提取更多的语义特征，又引入了池化层。池化层有下采样的作用，最大池化层可以提取更显著的特征，平均池化层可以提取更完整的特征。比如，步长为 2 的平均池化层可以用公式(3)表示。

$$y_{i,j} = \sum_{u=2i}^{2i+1}\sum_{u=2j}^{2j+1} x_{i,j} \Big/ 4 \qquad (3)$$

在卷积层和池化层后一般接若干个全连接层完成分类、回归或检测等其他任务。相关文献表明，全连接层有时可以被 GAP 层替代。具体见 1.1.4 节。

1.1.3 批归一化

根据反向传播算法的原理，在反向传播的时候，每层神经元收到的梯度是后层梯度的乘积。这也就意味，当神经网络的深度较大时，大梯度的累计会导致梯度值过大，即发生梯度爆炸

的问题，小梯度的累计则会导致梯度值过小，发生梯度消失的问题。
一种简单的方法是在神经网络的输入层对数据进行归一化，但因为前向传播时数据大小无法控制，在网络深度较大的时候梯度消失的问题并没有办法得到解决。
批归一化(batch normalize)[4]的引入一定程度上解决了这个问题。批归一化并不是仅是简单的把前向传播的数据归一化到正态分布上。简单的归一化到正态分布上会使网络的表达能力下降，损失性能。它先将传入的数据采用公式(4)归一化到正态分布上，再根据公式(5)进行尺度变换的偏移。其中μ_B和σ_B^2分别表示样本的均值和方差，ϵ为一个趋近于 0 的参数防止分母为 0，γ和β分别表示尺度大小的偏置，是需要学习的参数。
除了解决梯度消失的问题外，批归一化还可以使数据更简单，减小数据分布，使优化器更容易达到最优解，提高训练速度。

$$\hat{y_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad (4)$$

$$\hat{y_i} = \gamma\hat{y_i} + \beta \qquad (5)$$

1.1.4 全连接层与 GAP 层

对于图像分类问题，常见的方法是现将卷积层输出拉平，后接上全连接层用于分类。这需要存储大量的参数，对显卡的显存要求较高，同时还会带来过拟合问题。为了解决这个问题，有些文章采用了 dropout 的方法，即在全连接层中去掉部分连接。
实际上，相关研究[5]发现对于图像分类等计算机视觉任务，全连接层并不会发挥很大的作用。近年来流行的一种做法是，直接在卷积层后接一个全局平均池化层(GAP)将 W*H*C 维的输出映射到 C 维，后接全连接层和 softmax 分类。这种做法首先在[5]中提出，后在[3]中也有应用。这样做既可以减少参数数量，加快模型训练，又可以降低过拟合的影响。

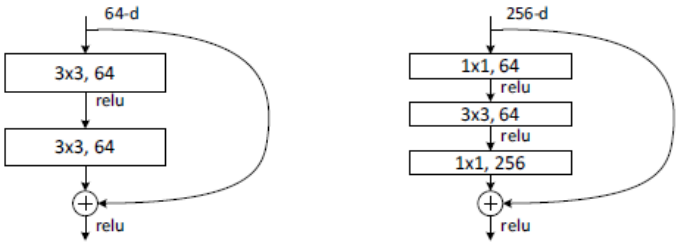
1.1.5 残差神经网络

尽管有了批归一化、RELU 等方法，但深度神经网络的学习仍然是一个很大的问题。在经典的网络结构 ResNet[3]中提出的残差层对解决这个问题提出了一种思路。
对于传统的神经网络，我们按照(6)的形式学习参数 W，对输入 x 做非线性变换得到 y。如果 x 已经是一组比较完备的输出，那么参数 w 很可能会破坏这个结果。残差层按照(7)的形式学习参数 w，后面的 x 项是一个恒等映射。在这种形式下，w*x 的数学意义就是 x 和 y 的残差。如果 x 是一组比较完备的输出，可以直接映射到 y，只需学习较小的 w 进行微调即可。这样减小了参数 w 的大小，增强了神经网络反向传播时的优化能力。
从另一个角度来说，残差层可以看作是浅层神经特征和深层特征的 ensemble。根据[4]中的实验结果，使用 ResNet 预训练好的参数，删掉其中的部分模块，并不会显著地影响它的分类准确率。

$$y = F(x, \{W_i\}) \qquad (6)$$

$$y = F(x, \{W_i\}) + x \qquad (7)$$

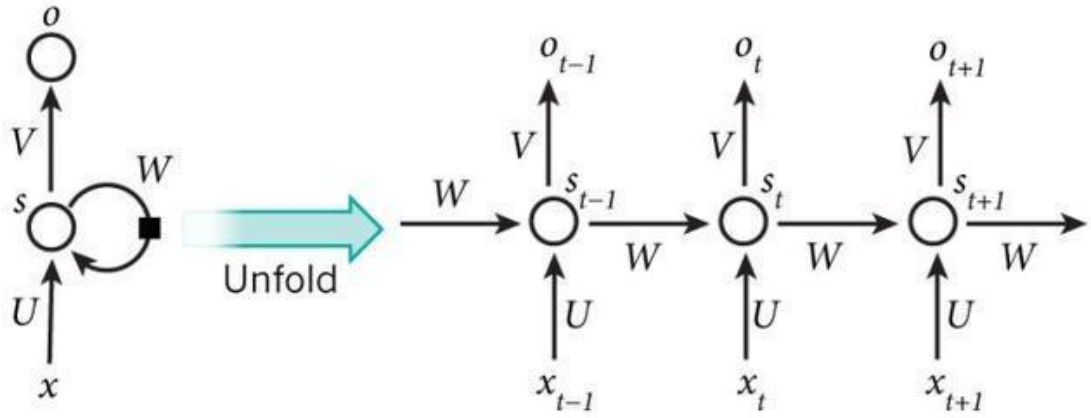
论文[3]中设计了两种残差层（如图 1），一种是常规的 BasicBlock，由两个 3*3 卷积级联。它希望使用两个卷积层的参数学习残差。另一种是 Bottleneck，它保留更大的特征空间（一般为 4 倍），先利用一个 1*1 卷积对特征进行降维，后接一个 3*3 卷积作下采样，最后接一个 1*1 卷积将特征升维回 4 倍的特征空间。这样做的目的是减少参数个数。假设当前卷积层通道数为 x，两个 3*3 卷积的参数数量为 3*3*x*x*2=18x，而两个 1*1 卷积加一个 3*3 卷积的参数数量为 1*1*4x*x*2+3*3*x*x=17x。也就是说，授予 Bottleneck 结构可以在不增加参数数量的情况下增加网络深度，这使得更深的网络变得可行。



（图 1 左为 Basicblock，右为 Bottleneck）

1.2 循环神经网络

RNN 背后的思想是利用顺序信息。在传统的神经网络中，我们假设所有的输入（包括输出）之间是相互独立的。对于很多任务来说，这是一个非常糟糕的假设。如果你想预测一个序列中的下一个词，你最好能知道哪些词在它前面。RNN 之所以循环的，是因为它针对系列中的每一个元素都执行相同的操作，每一个操作都依赖于之前的计算结果。换一种方式思考，可以认为 RNN 记忆了到当前为止已经计算过的信息。理论上，RNN 可以利用任意长的序列信息，但实际中只能回顾之前的几步。下面是 RNN 的一个典型结构图：



1.2.1 RNN 的前向传播

对于 RNN 中的其中一个神经元，它随着时序的展开，会结合之前的学习到的，以及新的输入，来更新自己的权重。神经元在时序上的迭代公式为(注意，在时序上共享权重)：

$$\boldsymbol{h}_t = f(U\boldsymbol{x}_t + W\boldsymbol{s}_{t-1} + \boldsymbol{b}_h),\ f\ is\ one\ kind\ of\ activation\ functions.$$

$$\boldsymbol{o}_t = softmax(V\boldsymbol{h}_t + \boldsymbol{b}_z)$$

1.2.2 RNN 的反向传播

通过定义一个总损失函数，对权重进行求导，并且沿着梯度的反方向更新，这就是反向传播的过程，由于 RNN 是一个时序演化过程，所以对有些权重的求导会用到之前的输出，所以 RNN 的反向传播叫做 BPTT。

如上图，对于每个输出神经元输出$\boldsymbol{o}_t$，标准输出为$\boldsymbol{y}_t$传递给下一个神经元参数为$\boldsymbol{s}_t$则时序损失函数公式为：

$$\boldsymbol{E}_t(\boldsymbol{o}_t, y_t) = -\boldsymbol{y}_t \log \boldsymbol{o}_t$$

总损失函数:

$$E(o, y) = -\sum_t \boldsymbol{y}_t \log \boldsymbol{o}_t$$

然后就可以求对 U,V,W 的梯度,即对于所有时刻的梯度求和:

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial \boldsymbol{E}_t}{\partial V} \quad \text{其余同理}$$

对于 V 在 t 时刻梯度为:

$$\frac{\partial \boldsymbol{E}_t}{\partial V} = \frac{\partial \boldsymbol{E}_t}{\partial \boldsymbol{o}_t} \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{V}}$$

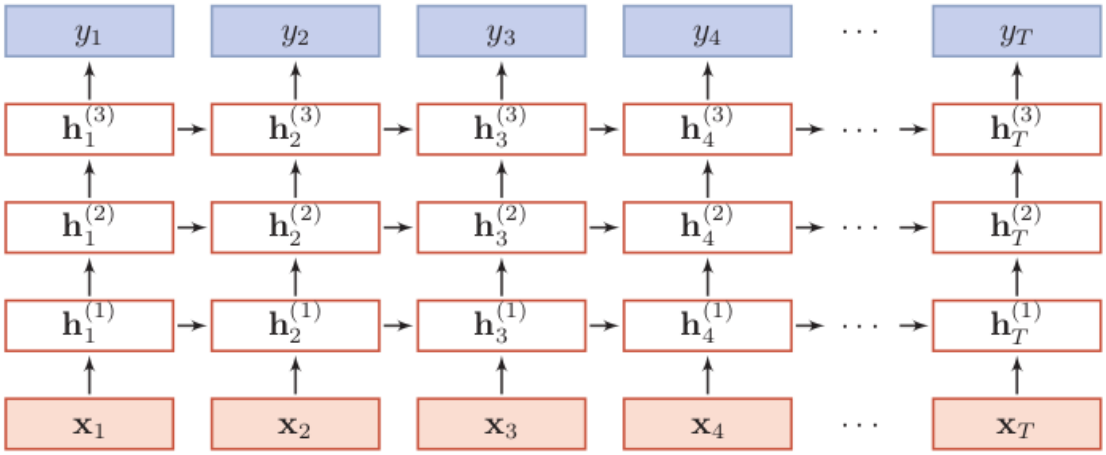
对于 W 在 t 时刻梯度为:

$$\frac{\partial \boldsymbol{E}_t}{\partial W} = \sum_{i=0}^{t} \frac{\partial \boldsymbol{E}_t}{\partial \boldsymbol{o}_t} \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{W}} \left(\prod_{j=i+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \frac{\partial \boldsymbol{s}_i}{\partial W}$$

对于 U 在 t 时刻梯度为:

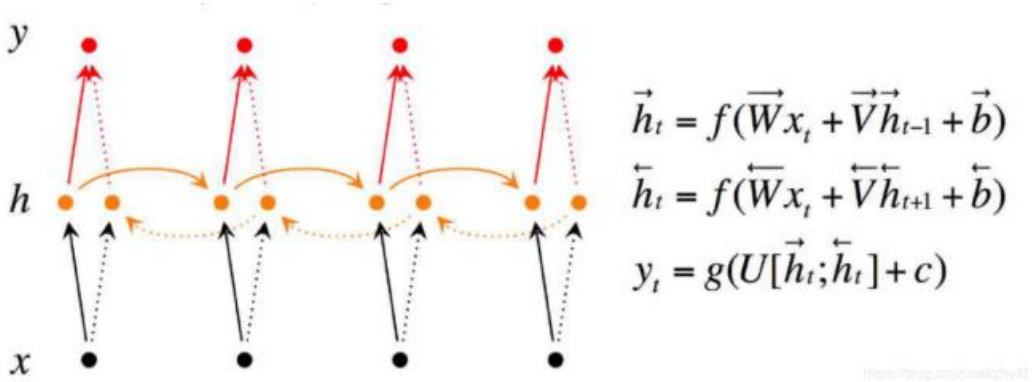
$$\frac{\partial \boldsymbol{E}_t}{\partial U} = \sum_{i=0}^{t} \frac{\partial \boldsymbol{E}_t}{\partial \boldsymbol{o}_t} \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{U}} \left(\prod_{j=i+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \frac{\partial \boldsymbol{s}_i}{\partial U}$$

1.2.3 堆叠 RNN



RNN 像全连接神经网络一样进行层层链接，这样加强了曾与曾之间的联络增强了数据之间的联系性，但是容易出现梯度消失或梯度爆炸。

1.2.4 双向 RNN

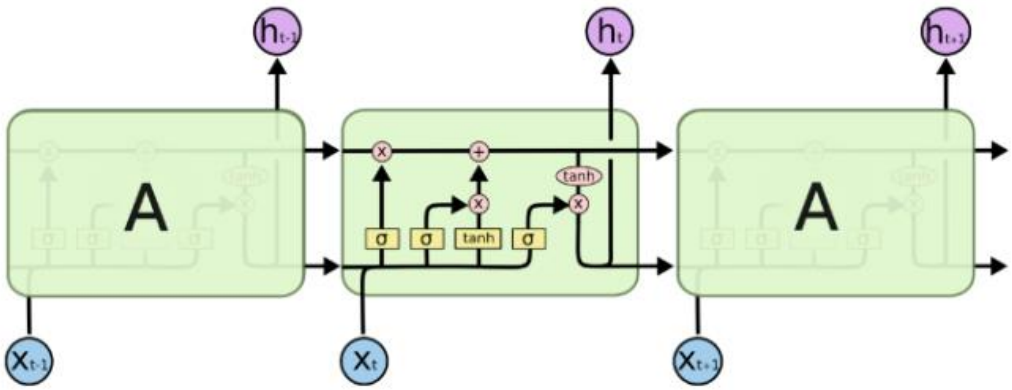


有些情况下，当前的输出不只依赖于之前的序列元素，还可能依赖之后的序列元素； 比如做完形填空，机器翻译等应用。既可正向传播也可反向传播

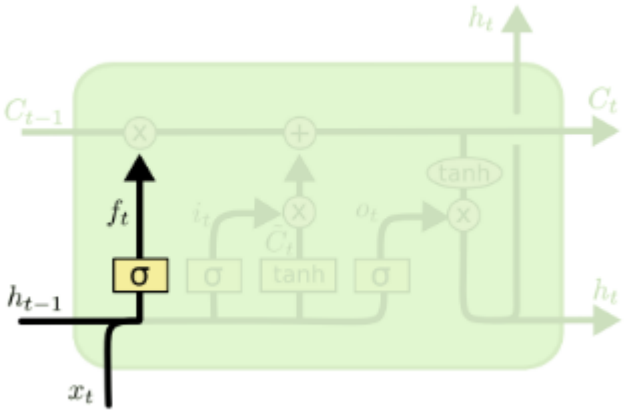
1.2.5 长短期记忆网络(LSTM)

由于在求梯度的时候会有$\prod_{j=i+1}^{t}\frac{\partial s_j}{\partial s_{j-1}}$项，而对于$\boldsymbol{s}_j = \tanh(U\boldsymbol{x}_j + W\boldsymbol{s}_{j-1})$，这里$\frac{\partial s_j}{\partial s_{j-1}}$为 tanh 的梯度，而当$\frac{\partial s_j}{\partial s_{j-1}}$足够大时$\prod_{j=i+1}^{t}\frac{\partial s_j}{\partial s_{j-1}}$就会过大而长生梯度爆炸的问题；同理，$\frac{\partial s_j}{\partial s_{j-1}}$足够小时，就会产生梯度消失的问题。解觉方案可以换激活函数，或者改进网络结构，LSTM 就是 RNN 的改进结构。

LSTM 的细胞结构如下图:



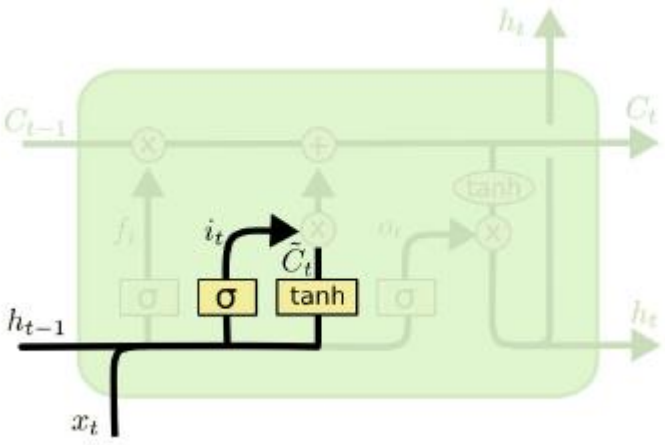
Forget Gate:



对于上个细胞的输入，利用 sigmoid 函数来得到一个在 0~1 之间的数来确定h_{t-1}的保留程度，其中

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

`Input Gate`:



先将当前输入经过两个激活函数，然后点乘，表示当前产生的隐状态有多少需要被保留并加入到历史信息中。

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

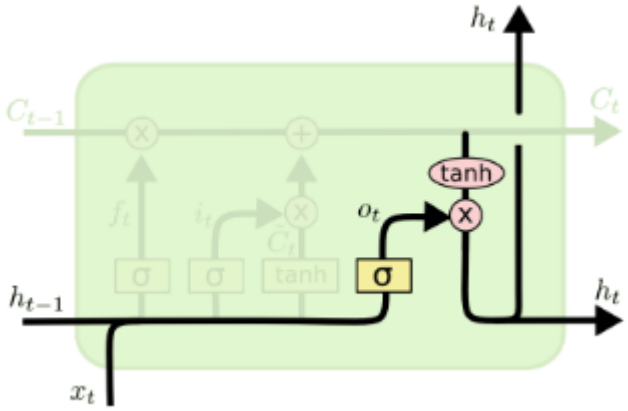
然后再进行细胞状态更新:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

得到当前的细胞状态C_t

剩下的为输出门

`Output Gate`



根据输入和当前细胞状态C_t来确定输出h_t，其中：

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

2. 网络结构与方法

2.1 CNN

我设计了一个简单的卷积神经网络结构，先使用三个卷积层提取特征，后使用两个全连接层进行分类，每个卷积层后接一个 RELU 函数做激活函数，再接一个最大池化层。具体结构见表 1。

Conv3-16 + RELU
MaxPool
Conv3-32 + RELU
MaxPool
Conv3-64 + RELU
MaxPool
FC-512
FC-10

（表 1 CNN）

2.1.1 VGGNet

我采用了[2]中的 VGG 网络的结构，对于 VGG11，VGG13，VGG16 三种结构分别进行了实验，并在网络中加入了批归一化。该结构的卷积部分由 5 个卷积块构成，分类器部分包含三个全连接层。对于 Batch_Normalize 结构，加在每个卷积层输出的后面。具体结构见表 2。

	VGG11	VGG13	VGG16
Block1	Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64

Pool1	MaxPool		
Block2	Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128
Pool2	MaxPool		
Block3	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv3-256
Pool3	MaxPool		
Block4	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512
Pool4	MaxPool		
Block5	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512
Pool5	MaxPool		
Fc1	Fc-4096		
Fc2	Fc-4096		
Fc3	Fc-10		

（表 2 VGGNet）

2.1.2 ResNet

参照论文[3]中的结构，我选择了 ResNet18，ResNet34，ResNet50 三种结构进行实验。对于 ResNet18 和 ResNet34，参照[3]我采用了 BasicBlock 作为残差层，对于 ResNet50 我采用 BottleNeck 作为残差层。为了提高准确率，我选择了[3]中在 Imagenet 数据集上训练的网络结构。因为 CIFAR-10 数据集的图片大小只有 32*32，所以对于第一个卷积层我没有像[3]一样采用 7*7 卷积核，而是采用了 3*3 卷积核，并且删去了 conv1 后的最大池化层。

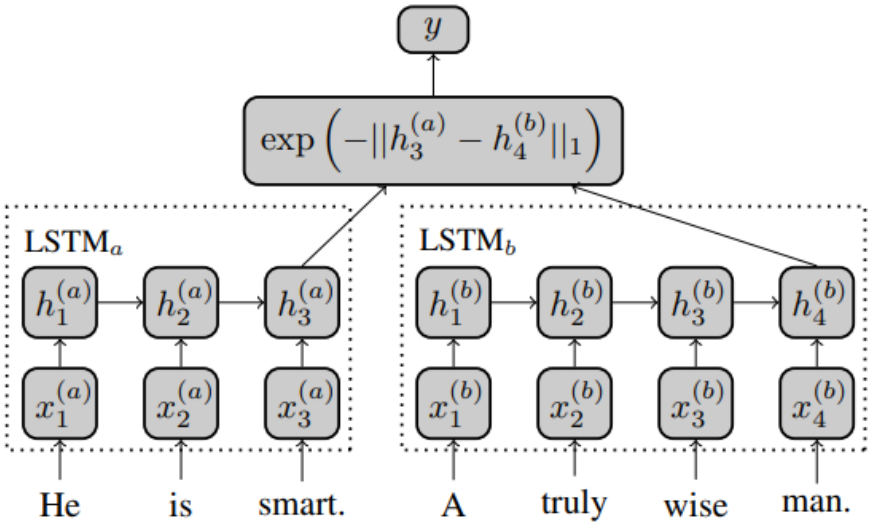
	ResNet18	ResNet34	ResNet50
Block1	Conv3-64		
Blovk2	$\begin{bmatrix} conv3-64 \\ conv3-64 \end{bmatrix}$*2	$\begin{bmatrix} conv3-64 \\ conv3-64 \end{bmatrix}$*3	$\begin{bmatrix} conv1-64 \\ conv3-64 \\ conv1-256 \end{bmatrix}$*3
Blovk3	$\begin{bmatrix} conv3-128 \\ conv3-128 \end{bmatrix}$*2	$\begin{bmatrix} conv3-128 \\ conv3-128 \end{bmatrix}$*4	$\begin{bmatrix} conv1-128 \\ conv3-128 \\ conv1-512 \end{bmatrix}$*4
Blovk4	$\begin{bmatrix} conv3-256 \\ conv3-256 \end{bmatrix}$*3	$\begin{bmatrix} conv3-256 \\ conv3-256 \end{bmatrix}$*6	$\begin{bmatrix} conv1-256 \\ conv3-256 \\ conv1-1024 \end{bmatrix}$*6
Blovk5	$\begin{bmatrix} conv3-512 \\ conv3-512 \end{bmatrix}$*3	$\begin{bmatrix} conv3-512 \\ conv3-512 \end{bmatrix}$*3	$\begin{bmatrix} conv1-512 \\ conv3-512 \\ conv1-2048 \end{bmatrix}$*3

pool	global average pool
Fc	Fc-10

(表 3 ResNet)

2.2 RNN

本次实验任务为判断两句子在语义上的相似度，考虑 LSTM 对比 RNN 的优点，我们以 LSTM 为起点，参照论文[7]中的模型，如下图:



定义一个孪生的 LSTM（即两个相同的 LSTM 网络）来得到两个句子通过网络的结果，这里不同的是，为了解决长度不等的问题，就预先将句子处理为 15x50 维的句向量传入网络中。然后通过比较最后输出结果的 h 来测得句子的相似度。

h 的取值在实验中我们分别尝试取最后一个 cell 的输出$h_{last}^{(a)}$, $h_{last}^{(b)}$和取所有 cell 输出的平均值$h_{avg}^{(a)}, h_{avg}^{(b)}$。并记录其相似度结果为 score,对于计算 score 我们采用的算法有:

1) $score = e^{-\left\|h^{(a)}-h^{(b)}\right\|_1}$

2) $score = e^{-\left\|h^{(a)}-h^{(b)}\right\|_2}$

3) $score = (1 + \frac{\sum_{i=1}^{hidden_{size}} h_i^{(a)} \times h_i^{(b)}}{\sqrt{h^{(a)^2}} \times \sqrt{h^{(b)^2}}}) \times 2.5$

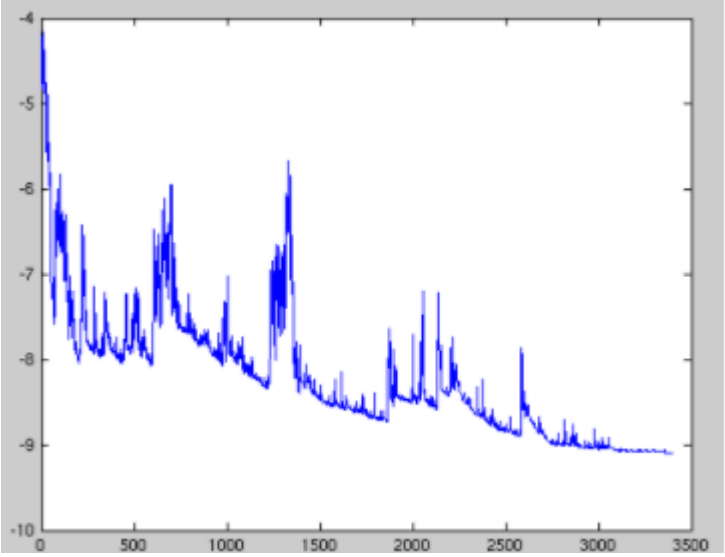
将 score 置为网络的输出，与标准的相似度比较。设置均方误差损失函数,优化算法采取随机梯度下降(SGD)优化算法。

随机梯度下降算法每次从训练集中随机选择一个样本来进行学习，即:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

批量梯度下降算法每次都会使用全部训练样本，因此这些计算是冗余的，因为每次都使用完全相同的样本集。而随机梯度下降算法每次只随机选择一个样本来更新模型参数，因此每次

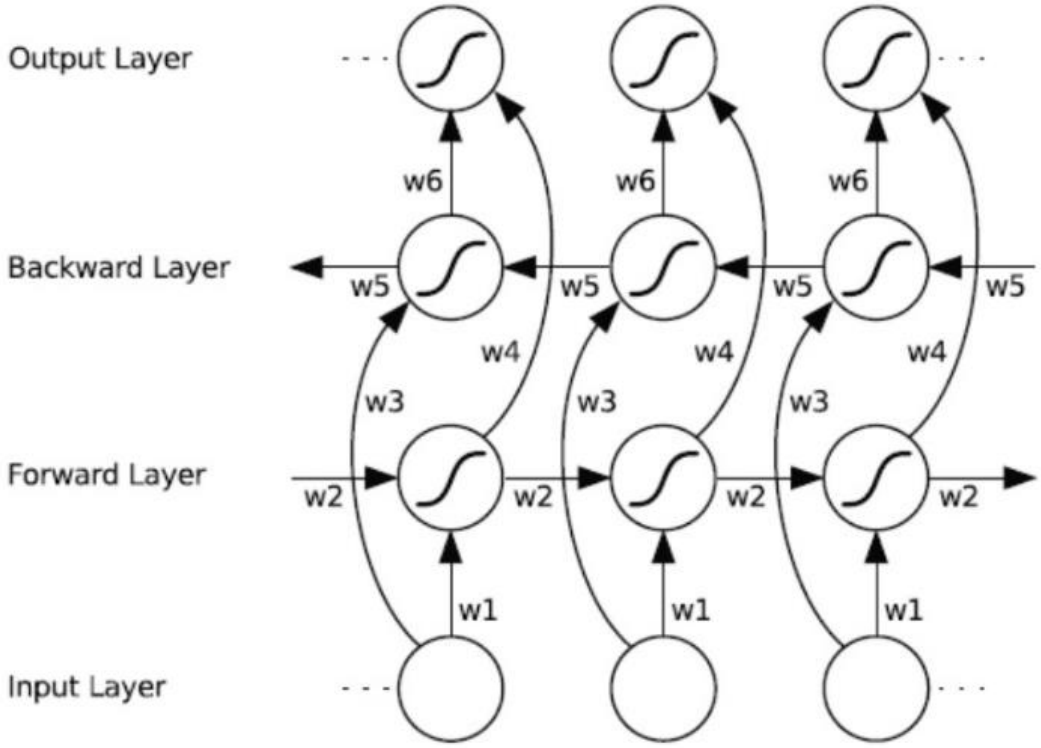
的学习是非常快速的，并且可以进行在线更新。缺点是 SGD 的噪音较 BGD 要多，使得 SGD 并不是每次迭代都向着整体最优化方向。所以虽然训练速度快，但是准确度下降，并不是全局最优,会造成优化波动,如下图:



我们设定评判模型好坏的标准为皮尔逊系数或均方误差。

2.2.1 BiLSTM

由于双向的 LSTM 可以更捕捉双向的语义依赖，进行正向和反向传播，大致示意图如下:



原理类似双向 RNN。将其细胞替换为 LSTM cell，参数传递的形式也替换为 LSTM 网络的传递形式即可。

3. 实验结果

3.1 卷积神经网络

3.1.1 数据集与相关说明

本次实验采用 CIFAR-10 数据集，该数据集共包含 60000 张 32*32 的图片，其中 50000 张为训练集，10000 张为测试集。数据集中的图片总共有 10 中类别，分别是猫、青蛙、货车、鹿、汽车、鸟、马、船、飞机、狗。

因为测试集中有标签，我直接将 50000 个样本全部用于训练集，并将测试集当成验证集，评估结果。

除了 ResNet18*外，其他实验中都进行了数据增强操作，先在图像外部填充 4 个像素的 0，再将其随机裁剪为 32*32 的形式，最后将其归一化。ResNet18*没有进行数据增强。

对于所有的模型，我训练 200 个 epoch 并取验证集准确率最高的模型作为结果。前 100 个 epoch 学习设为 0.1，100-150 个 epoch 学习率设为 0.01，最后 50 个 epoch 学习率设为 0.001。使用随机梯度下降用于最优化，momentum 设为 0.9，weight_decay 设为 1e-4。

Method	aero	auto	bird	cat	deer	dog	frog	horse	ship	truck	Mean
CNN	84.50	92.40	74.00	64.10	79.90	74.30	85.20	85.90	89.90	86.70	81.69
CNN_GAP	80.60	90.80	72.60	61.00	78.80	76.50	85.20	83.50	91.30	88.20	80.85
VGG11_BN	93.10	97.10	90.10	82.60	93.20	85.20	93.70	94.40	95.10	93.70	91.82
VGG11_GAP	93.80	96.50	87.90	83.30	92.80	97.30	95.60	93.80	95.60	94.30	92.09
VGG13_BN	94.90	97.20	91.50	87.00	94.10	90.20	95.10	95.60	96.70	95.80	93.81
VGG13_GAP	95.10	96.50	**93.50**	85.00	95.00	90.50	95.30	95.60	97.00	96.20	93.97
VGG16_BN	95.00	97.00	91.50	86.30	93.40	89.30	95.30	95.70	95.40	95.50	93.44
VGG16_GAP	94.00	97.50	91.10	86.00	94.10	89.50	95.80	95.20	96.20	**96.70**	93.61
ResNet18	95.50	**98.10**	91.70	89.10	95.40	90.90	96.50	97.30	96.00	95.70	94.62
ResNet34	95.50	97.50	93.30	**89.20**	**96.40**	**91.10**	96.40	**96.00**	97.00	96.60	**94.90**
ResNet50	95.30	97.80	92.90	88.00	96.10	90.60	**97.30**	94.80	**97.10**	96.00	94.59
ResNet18*	**95.70**	97.40	91.50	88.00	95.50	90.90	97.40	96.10	96.30	96.50	94.53

（表 4 不同模型的准确率）

3.1.2 简单的 CNN

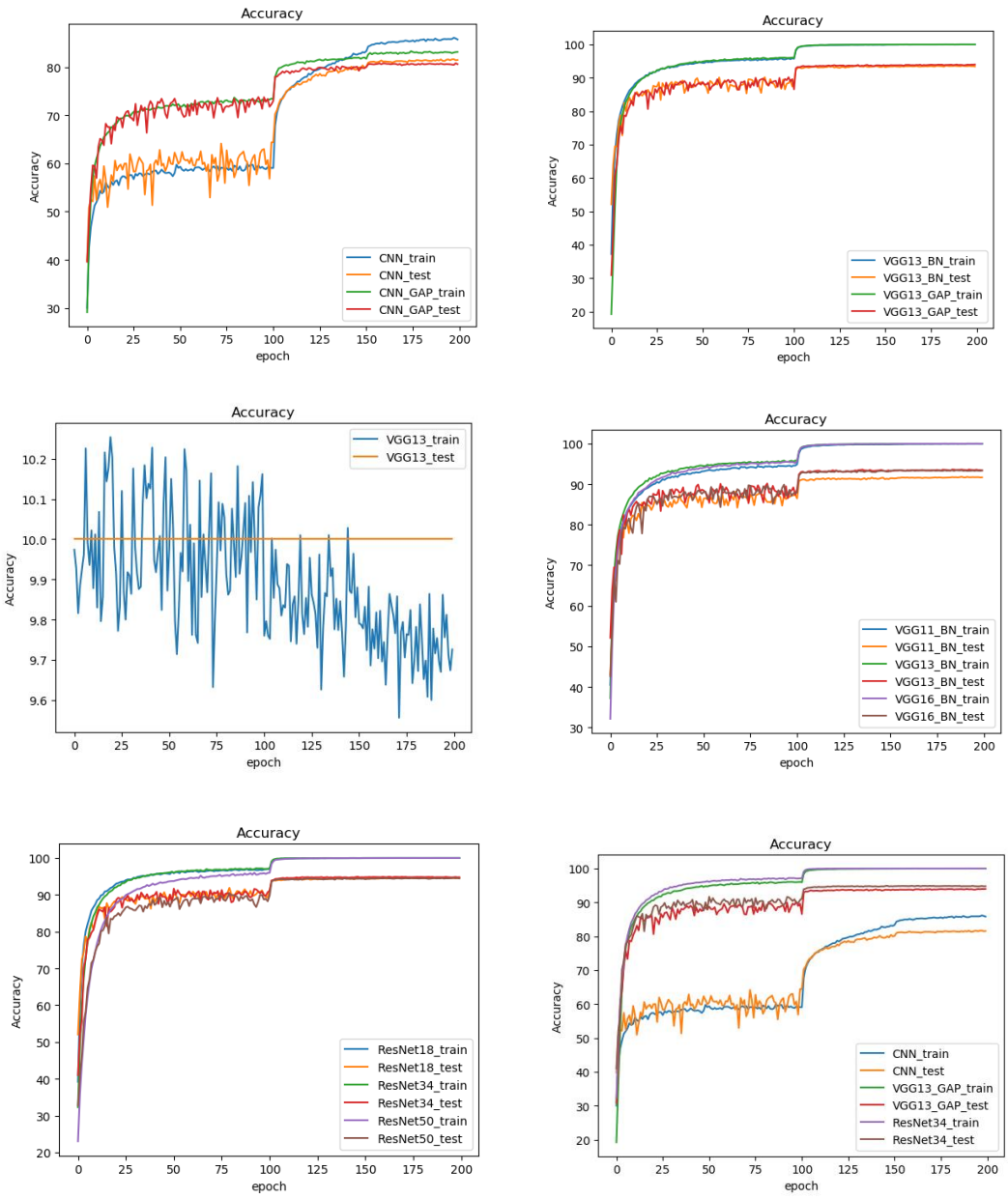
我按照 2.1 的结构设计一个简单的 CNN 网络，这个网络在 CIFAR-10 数据集上取得了 81.69%的平均准确率。

3.1.3 更深的网络

我尝试加大网络深度，使用了 2.2 节中提到的 VGGNet 结构。根据图 2，如果不加入 BatchNormalize 层，无论是在训练集还是在验证集上 VGG 网络都很难收敛。我认为这是因为发生了严重的梯度消失。我尝试在每个卷积层后加入批归一化，模型的结果就正常了很多。

在加入 BatchNormalize 层的情况下，使用 VGG11 可以获得 91.82%的准确率，使用 VGG13 可以获得 93.81%的准确率。不过继续加大深度使用 VGG16 网络，准确率却有小幅下降，只

有 93.44%。无论是训练集还是验证集，VGG13 的表现都优于 VGG16，我认为这是因为深度过大，参数过多，出现梯度消失导致的结果。



（图 2 CNN、VGG、ResNet 的训练日志）

3.1.4 GAP 层与全连接层

我尝试在 CNN 和 VGG 网络上进行实验，在卷积输出后去掉全连接层，直接加 GAP 层后接一个 softmax 层用于分类。如表 4 可以看到，把 VGG 网络上的全连接层去掉甚至可以提升分类准确率。但对于简单的 CNN 网络，将全连接层换成 GAP 层后准确率却有小幅下降。这主要是因为 CNN 网络卷积层较少，学来的特征不够 discriminative，出现了欠拟合，需要全连接层的加强，而 VGG 网络学习的特征足够充分，可以直接用来分类。

表 5 也佐证了这一观点，去掉全连接层的 CNN 由于参数数量太少，学习能力不足，出现欠拟合，训练集准确率远低于加了全连接层的 CNN。对于 VGG 网络，去掉全连接层后网络训练

集的准确率小幅降低，但测试集准确率更高。由此可以得出结论，对于特征比较完备的卷积层，后接全连接层会造成模型冗余，并不会提升准确率，简单地使用 GAP 层一定程度上可以减少了过拟合。

Model	train	test
CNN	88.52	81.69
CNN_GAP	83.17	80.85
VGG11_BN	99.99	91.82
VGG11_BN _GAP	99.93	92.09
VGG13_BN	99.99	93.81
VGG13_BN _GAP	99.98	93.97
VGG16_BN	99.99	93.44
VGG16_BN _GAP	99.97	93.61

（表 5 GAP 层对训练/测试准确率的影响）

3.1.5 残差神经网络

我使用了 ResNet18，ResNet34，ResNet50 三种网络结构。如表 4，ResNet34 的准确率最高，达到了 94.90%。ResNet50 的深度更大，准确率却略低于 ResNet34，我认为这主要是因为，ResNet50 和 ResNet34 在卷积块数量上完全一致，它更深是因为采用了 bottleneck 结构，这种结构中有下采样作用的卷积层较少，导致提取的特征不如 ResNet34 全面。

3.1.6 数据增强

我尝试将数据增强去掉，只对图像进行归一化，采用 ResNet18 的结构进行训练。如表 4，去掉数据增强的准确率会稍微下降。由此可见数据增强对模型鲁棒性的提升。
因为 CIFAR-10 数据集图像大小基本相同，不存在不同尺度的问题，所以我认为多尺度的图像并不会对模型准确率产生提升，也就没有对图像尺度进行增强。

3.1.7 类别准确率分析

表 4 为各种模型在 10 种类别上的准确率。可以看到对于以下各种模型，猫和狗的分类准确率最差。根据表 6 中 ResNet34 模型的混淆矩阵和表 7 也可以看到，把猫识别成狗和把狗识别成猫是错误识别中频率最高的两种，即使对于准确率超过 94%的网络也有 5%以上的错误率。这其实是可以理解的，因为猫和够的轮廓特征比较相似，主要区别在细节特征，而 CIFAT-10 数据集的图像较小，细节特征也不够全面。

predict\label	aero	auto	bird	cat	deer	dog	frog	horse	ship	truck
Aero	**95.50**	0.40	1.40	0.60	0.10	0.30	0.40	0.40	1.10	0.20
Automobile	0.10	**97.50**	0.00	0.00	0.00	0.10	0.00	0.00	0.50	2.20
Bird	1.60	0.00	**93.30**	1.70	0.80	0.40	1.30	0.40	0.30	0.10
Cat	0.60	0.00	1.10	**89.20**	1.10	5.80	1.00	0.90	0.40	0.10
Deer	0.40	0.20	1.50	1.20	**96.40**	1.50	0.30	1.20	0.00	0.00
Dog	0.00	0.00	0.90	5.10	0.80	**91.10**	0.30	1.00	0.00	0.00
Frog	0.10	0.00	1.30	1.20	0.30	0.10	**96.40**	0.00	0.10	0.00

Horse	0.00	0.00	0.30	0.50	0.50	0.40	0.00	**96.00**	0.00	0.00
Ship	1.50	0.40	0.20	0.30	0.00	0.00	0.10	0.00	**97.00**	0.80
truck	0.20	1.50	0.00	0.20	0.00	0.30	0.20	0.10	0.60	**96.60**

（表 6 ResNet34 的混淆矩阵）

Model	Dog→cat	Cat→dog
CNN	12.90	14.60
VGG13_BN	5.90	6.50
VGG13_BN_GAP	5.50	7.30
ResNet18	5.80	6.00
ResNet34	5.80	5.10
ResNet50	5.30	5.80

（表 7 部分模型猫狗误分类统计）

3.2 循环神经网络

3.2.1 循环神经网络数据集

本次实验用到的循环神经网络数据集为 STSBenchmark，训练集测试集以及验证集中每一行数据都是以’\t’划分，而有些句子中会有’,’读取出会多出一项,所以在读取时要先合并’,’,然后将每一行数据以’\t 划分’,然后提取其中的我们需要的 Sentence1,Sentence2 以及他们的相似度 k。

然后经过统计发现，最长的句子有 58 个词而最短的句子只有 2 个词，所以这里我们需要对句子进行预处理一下。首先是将句子统一为小写字母且去掉缩写，这样将句子格式稍微统一一下之后，再利用一个正则表达式去掉标点符号，再利用 nltk.stem 包中的 SnowballStemmer 来去除句子中的停止词(Stopwords)。此时句子对机器的可读性已经大大提高了。

然后对每个句子进行编码，考虑到使用 one-hot 矩阵可能会使数据集过大，这里决定使用 Word2Vec 方法。

Word2Vec 原理：

统计语言模型给出了这一类问题的一个基本解决框架。对于一段文本序列 S=w1,w2,…,wT，它的概率可以表示为：

p(S)=p(w1,w2,w3,w4,w5,…,wt)=p(w1)p(w2|w1)p(w3|w1,w2)…p(wt|w1,w2,…,wt-1)

基于马尔科夫假设（Markov Assumption）：下一个词的出现仅依赖于它前面的一个或几个词。假设下一个词的出现依赖它前面的一个词，则有：

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)…p(wn|w1,w2,…,wn-1)

=p(w1)p(w2|w1)p(w3|w2)…p(wn|wn-1) # bigram

假设下一个词的出现依赖它前面的两个词，则有：

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)…p(wn|w1,w2,…,wn-1)

=p(w1)p(w2|w1)p(w3|w1,w2)…p(wn|wn-1,wn-2) # trigram

那么，我们在面临实际问题时，如何选择依赖词的个数，即 n。

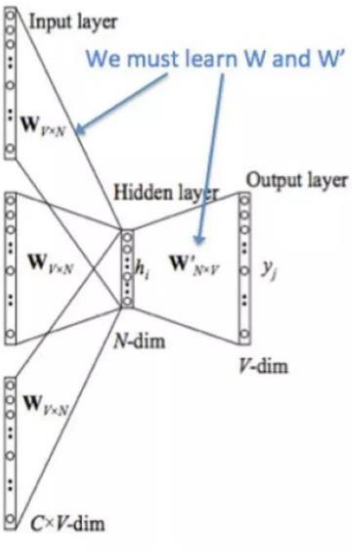
更大的 n：对下一个词出现的约束信息更多，具有更大的辨别力；

更小的 n：在训练语料库中出现的次数更多，具有更可靠的统计信息，具有更高的可靠性。

理论上，n 越大越好，经验上，trigram 用的最多，尽管如此，原则上，能用 bigram 解决，

绝不使用 trigram。
Word2Vec 分为两种:CBOW 和 Skip-gram

CBOW:



:

CBOW（Continuous Bag-of-Word Model）又称连续词袋模型，是一个三层神经网络。如上图所示，该模型的特点是输入已知上下文，输出对当前单词的预测。输入为上下文的 one-hot 向量(V-dim)，然后所有的 one-hot 分别乘以共享的权重矩阵 W 的结果相加求平均得到隐藏层的向量h_i(N-dim)，再通过权重矩阵 W'得到输出结果向量y_i(V-dim),最后再与标准的 one-hot 比较的到误差。制定一个损失函数然后通过梯度下降来更新权重矩阵。训练完毕后,输入层的每个单词与矩阵 W 相乘的得到的向量就是我们要的词向量(Word embedding)
这里我们调用的算法为 CBOW，就不对 Skip-gram 多加赘述。
对于每个词我们得到一个 50 维的词向量，然后对每个句子做 padding 操作，我们采用的使补 0 和删去尾部过长的词向量，控制每个句子为一个 15x50 维的矩阵。

3.2.2 LSTM 模型对比

设定一个 normal 模型为我们最初的 LSTM 模型，其参数如下表:

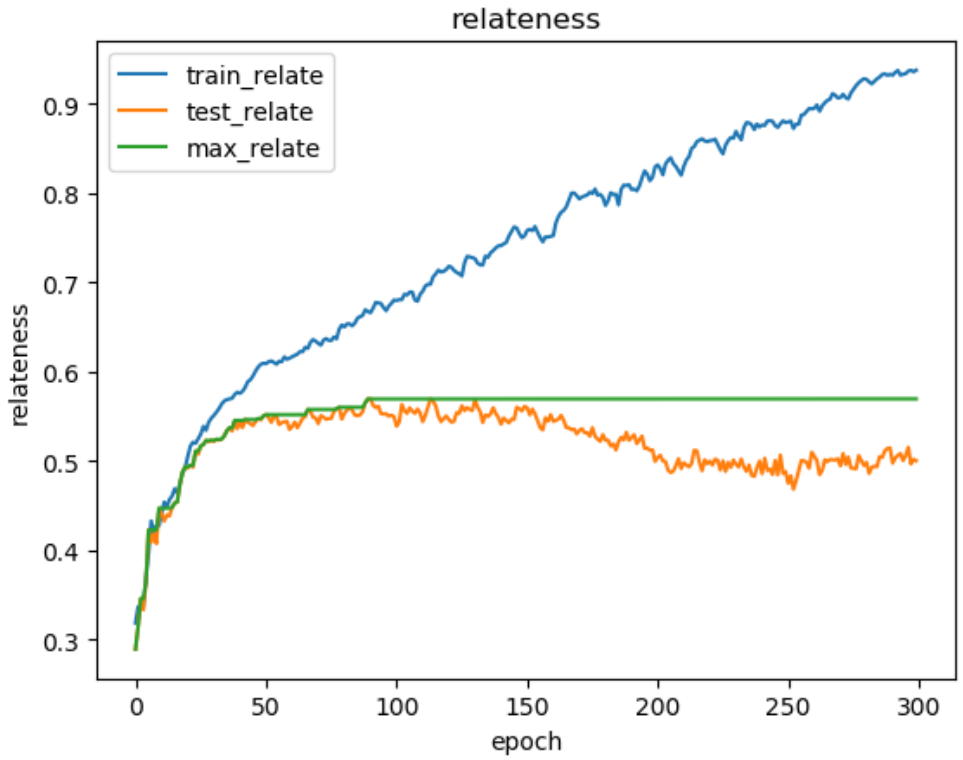
LSTM 结构参数	参数值	参数解释
epoch	300	遍历训练集的次数
batch_size	128	每次投入网络中句子的数量
input_size	50	每个 cell 的输入的向量维度
lr	0.1	学习率
hidden_size	256	隐藏层维度
metric	L1	匹配度 score 的算法的选择
num_layers	2	LSTM 网络层数
bidrect	Ture	是否为双向 LSTM
output_last	True	输出是否为最后一个 cell 的 h

注：metric 的取值 L1: $score = e^{-\left\|h^{(a)}-h^{(b)}\right\|_1}$,L2: $score = e^{-\left\|h^{(a)}-h^{(b)}\right\|_1}$,

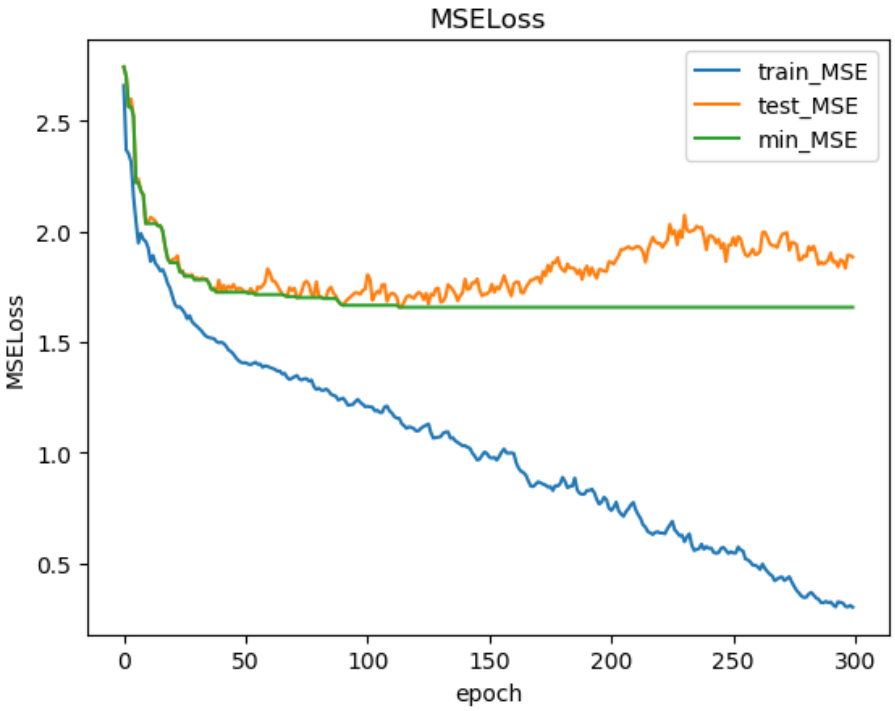
cos: $score = (1 + \frac{\sum_{i=1}^{hidden_{size}} h_i^{(a)} \times h_i^{(b)}}{\sqrt{h^{(a)^2}} \times \sqrt{h^{(b)^2}}}) \times 2.5$

当 output_last 为 True 时，输出为最后一个 cell 的 h，否则为每个 cell 输出的 h 平均值。

3.2.3 normal 模型的皮尔森相关系数



3.2.4 normal 模型的 MSEloss



由上面两张图可以发现，随着训练次数的提升，在训练集上的准确率一直在提高，而在测试集上的准确率逐渐收敛，这说明在后续的训练中，模型出现了过拟合的问题，下面对其他模型的运行结果进行对比。

3.2.5 模型对比

normal 模型上修改参数	测试集最大皮尔森系数	测试集最小均方误差
default	57.15	1.66(train: 93.41/0.31 过拟合)
output_last=False	56.81	1.65
bidirect=False	56.48	1.68
metric=L2	**52.23**	1.80
metric=cos	39.29	2.27
num_layers=1	56.59	1.67(train: 65.06/1.29 欠拟合)
num_layers=3	54.74	1.73(train: 96.24/0.17 过拟合)
添加全连接层 256-128	56.16	1.71
添加全连接层 256-32	55.07	1.72

发现除了利用余弦距离作为比较参数时相关性系数较低，均方误差较大之外，其他的情况在测试集上的皮尔森相关性系数都能落在(50,60)，均方误差落在(1.5,2)，单层的 LSTM 在训练集上欠拟合，其他网络结构在训练集上过拟合

4. 创新点

4.1卷积神经网络

1）使用了批归一化层(BatchNornalize)，尝试克服梯度消失的问题。
2）尝试加大网络深度，测试并分析了不同深度的神经网络的性能。
3）尝试引入 GAP 层代替传统的全连接层，构造全卷积神经网络，发现了全连接层的冗余性。
4）使用了残差神经网络的结构，并取得了不错的性能。

4.2循环神经网络

1）使用 LSTM 解决简单循环神经网络梯度爆炸或消失的问题。
2）利用孪生神经网络来处理成对的数据。
3）尝试双向 LSTM 或多层 LSTM 来提高模型准确率。
4）比较使用所有 cell 输出结果的平均和最后 cell 的输出结果的异同，尝试在 LSTM 模型后加入全链接层提取数据。
5）利用余弦距离，一、二范数比较句向量相似度。

5. 结论

5.1CNN

本文使用了 CNN、VGGNet、ResNet 等多种骨架网络结构，加入了 BatchNormalize 和 GAP 层等 trick，通过在 CIFAR-10 数据集上的测试，得到了比较充分结果。我们验证了批归一化层对网络准确率提升的作用，也验证了全连接层的冗余性。我们尝试引入残差层，取得了不错的结果。最终我们的模型在 CIFAR-10 上可以取得 94.90%的准确率。

5.2 RNN

利用 Word2Vec 来对数据集进行编码，然后利用孪生 LSTM 网络，并对其输出结果进行了不同方式的处理，尝试了 BiLSTM，Stack LSTM 等网络结构来寻求更好的实验结果。还在网络输出的结果上尝试加入了全连接层来调试结果。发现单层的 LSTM 在训练集上欠拟合，其他网络结构在训练集上过拟合。实现效果除了利用余弦距离作为比较参数时相关性系数较低，均方误差较大之外，其他的情况在测试集上的皮尔森相关性系数都能落在(50,60)，均方误差落在(1.5,2)

6. 实验分工

王程钥：CNN 相关内容以及部分 RNN 调参
朱绪思：RNN 相关内容

参考文献

[1] Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998
[2] K.Simonyan and A.Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015
[3] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. Deep Residual Learning for Image Recognition. In CVPR, 2016
[4] S.Ioffe and C.Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In ICML 2015
[5] Min Lin and Qiang Chen and Shuicheng Yan. Network In Network. In ICLR, 2014
[6] Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In CVPR, 2017
[7] Jonas Mueller and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)
[8] Xin Rong. word2vec Parameter Learning Explained
[9] FantDing. https://www.jianshu.com/p/043083d114d4
[10]宇过天晴 li. https://blog.csdn.net/yu5064/article/details/79601683