# CNN&RNN

人工智能实验

小组成员1：　仲　逊 16327143

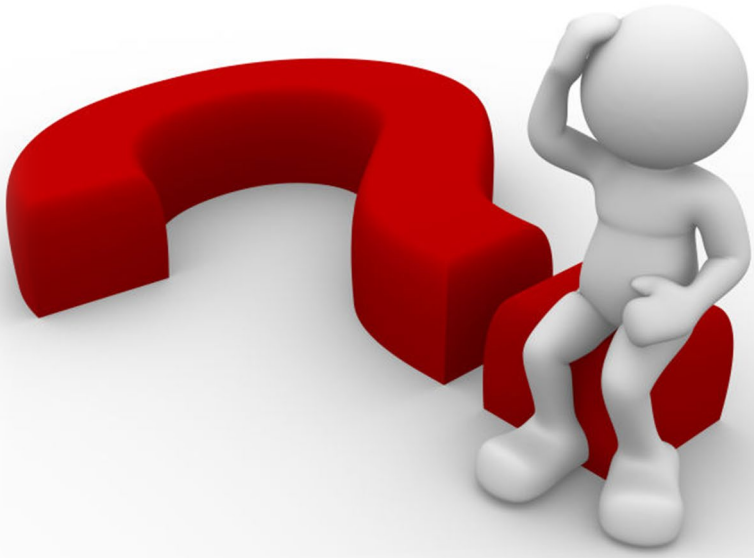
小组成员2：　江金昱 17341067

# 目录

# RNN

# 问题简述

- 给定句子A，句子B，计算两个句子的语义情感相似度
- 并根据相似度高低，给出0-5的打分
- 5最高，0最低



# 实现思路

1. 将句子表示成矩阵的形式，单词是矩阵中的一行向量

单词的表示可以使用one-hot，即构建字典

也可以使用word embedding，调用与训练好的模型将单词映射到指定的维度

2. 定义RNN/LSTM模型

3. 将两个句子中的单词分别循环输入模型（共享参数），得到最终的output

4. 计算两个output的相似度

5. 计算误差，更新梯度

6. 重复3、4、5直到模型收敛

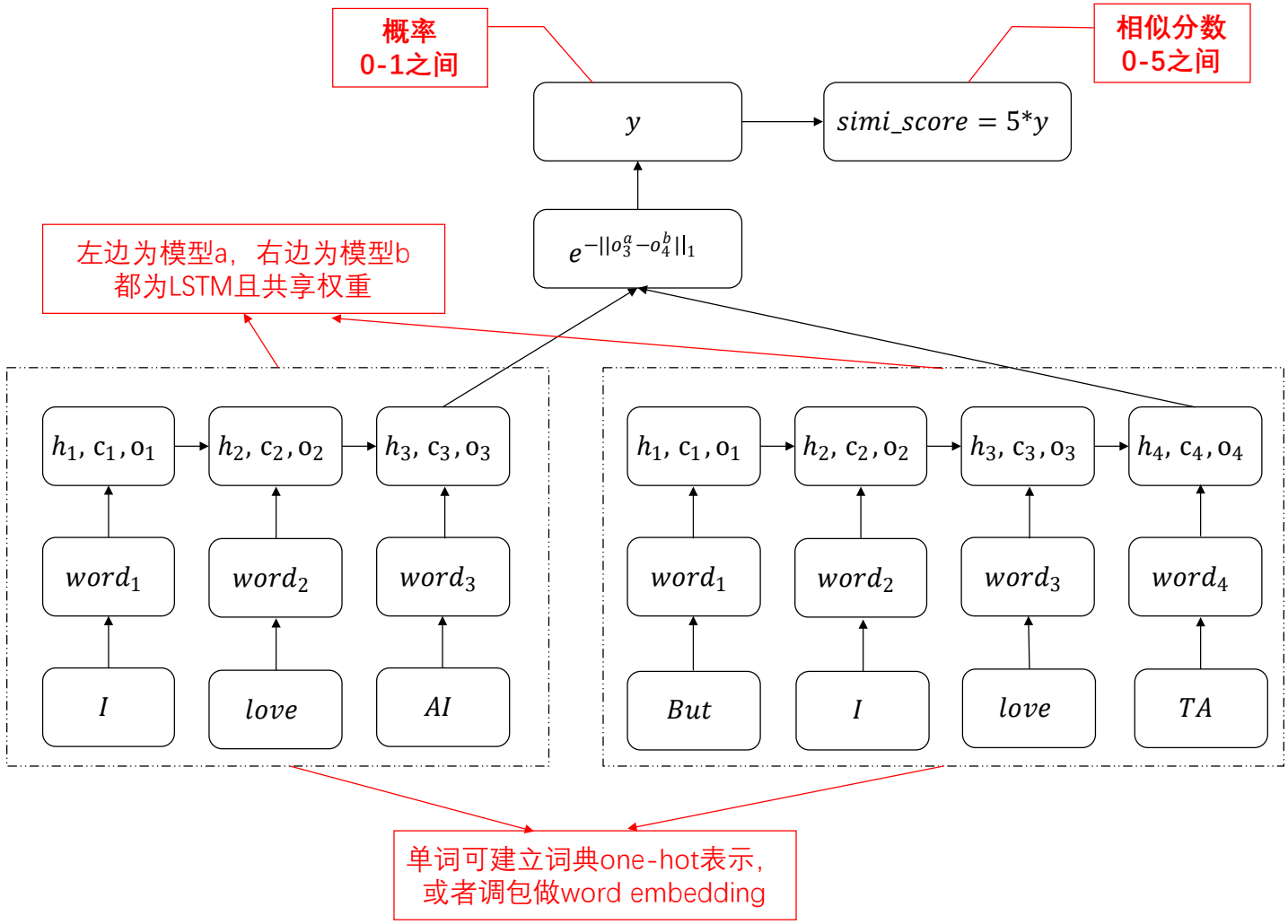
# 创新

1. 动态调整学习率

2. 引入LSTM机制

3. 当LSTM的num_layers（堆叠层数大于1）的时候，引入dropout机制，使神经元以一定概率失活

# 网络结构


概率
0-1之间
相似分数
0-5之间
$y$
$simi\_score = 5*y$
$e^{-||o_3^a - o_4^b||_1}$
左边为模型a，右边为模型b
都为LSTM且共享权重
$h_1, c_1, o_1$
$h_2, c_2, o_2$
$h_3, c_3, o_3$
$h_4, c_4, o_4$
$word_1$
$word_2$
$word_3$
$word_4$
$I$
$love$
$AI$
$But$
$I$
$love$
$TA$
单词可建立词典one-hot表示，
或者调包做word embedding

# 结果分析

| 基础参数 | | |
|---|---|---|
| **参数名** | **解释** | **值** |
| Num_layers | Rnn每一时刻堆叠的层数 | 1 |
| Drop_out_prob | 神经元失活的概率，仅当num_layers>1的时候有效 | 0 |
| Hidden_size | 隐藏状态向量的特征的维度 | 128 |
| Embed_size | 输入的特征向量的特征的维数 | 字典大小/embedding后的大小 |
| Learning_rate | 学习率 | 初始化为0.1，后经动态调整 |

# 结果分析–RNN模型

| Epoch（训练轮数） | 训练集MSE | DEV验证集MSE |
|---|---|---|
| 1 | 4.470229 | 3.113677 |
| 5 | 2.682181 | 2.482660 |
| 10 | 2.452715 | 2.505848 |
| 20 | 2.505528 | 2.612407 |
| 30 | 2.560050 | 2.485946 |

可以看到模型在训练集上在20-30epoch时已经拟合了，而验证集的误差在5个epoch后变化已经不大，不论是偏差和方差都较大。

皮尔逊系数：0.2371

# 结果分析-纯LSTM模型

| Epoch（训练轮数） | 训练集MSE | DEV验证集MSE |
|---|---|---|
| 1 | 2.788375 | 2.363721 |
| 5 | 1.943528 | 2.229069 |
| 10 | 1.657909 | 2.027339 |
| 20 | 1.248842 | 1.880080 |
| 30 | 0.947972 | 1.793984 |
| 85 | 0.475499 | 1.683241 |

可以看到引入LSTM后，在训练集上的拟合能力增强，在验证集上的泛化能力也增强，

此时模型相对于RNN具有较低的方差和偏差。

最终在测试集上的Pearson相关系数为：0.48066

# 结果分析-LSTM增加堆叠层数，引入dropout

| Epoch（训练轮数） | 训练集MSE | DEV验证集MSE |
|---|---|---|
| 1 | 2.966341 | 2.270921 |
| 5 | 1.793359 | 1.945135 |
| 10 | 1.557647 | 1.777588 |
| 20 | 1.334712 | 1.706778 |
| 30 | 1.236174 | 1.660273 |
| 50 | 1.176918 | 1.653160 |
| 100 | 1.173570 | 1.647994 |

num_layers=3（堆叠层数为3），dropu_out=0.1/0.2 结果类似

当加深网络层数，即使加入了dropout，但是只是在训练集上的MSE减少了，验证集上的MSE不降反升，泛化效果较差，模型面临过拟合的问题，具有低偏差，高方差

# 结果分析-LSTM，数据采用word embedding

| Epoch（训练轮数） | 训练集MSE | DEV验证集MSE |
|---|---|---|
| 1 | 2.551955 | 2.540359 |
| 5 | 2.095119 | 2.560954 |
| 10 | 1.991081 | 2.954000 |
| 20 | 1.787526 | 5.105361 |
| 30 | 1.5044667 | 3.979506 |
| 50 | 0.921095 | 3.580214 |
| 100 | 0.540754 | 2.595597 |

加入embedding后模型拟合较快，因为embedding在一定程度上完成了一部分的单词编码工作，因此能加速模型的训练。另外在最终结果上，加入了embedding的皮尔逊系数也是最高的。模型具有较低的偏差和方差。

最终在测试集上的Pearson相关系数为：0.50225


CNN

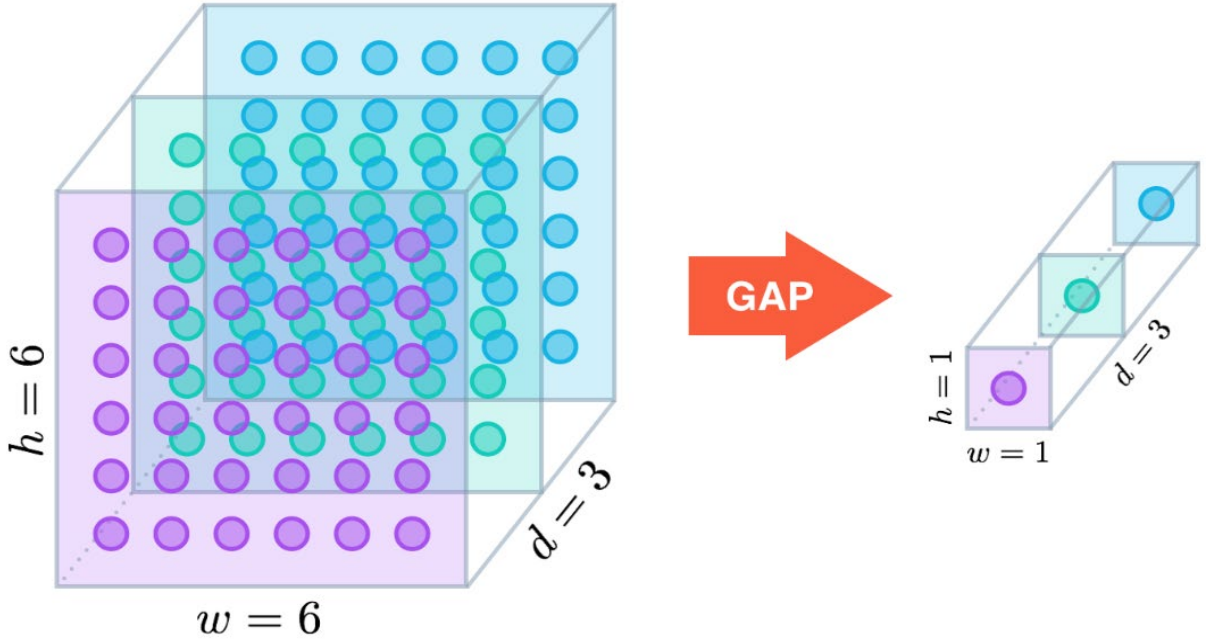
# 实现思路

1. 加载读取数据集并划分，pytorch官方的Dataloader带来了许多便利
2. 设计CNN结构
3. 将数据批量输入网络得到预测结果，计算和训练集的真实tag之间的误差，使用交叉熵损失函数，更新梯度
4. 模型收敛后得到最终网络参数，统计测试集上的准确率
5. 调整模型/参数/优化器，重复3和4过程，期间记录不同模型的效果并作图对比，直到最终模型效果较为理想

# 创新

## 1. 全局平均池化（GAP）

- Network in Network使用GAP来取代了最后的全连接层，直接实现了降维，更重要的是极大地减少了网络的参数(CNN网络中占比最大的参数其实后面的全连接层)。Global average pooling的结构如下图所示：



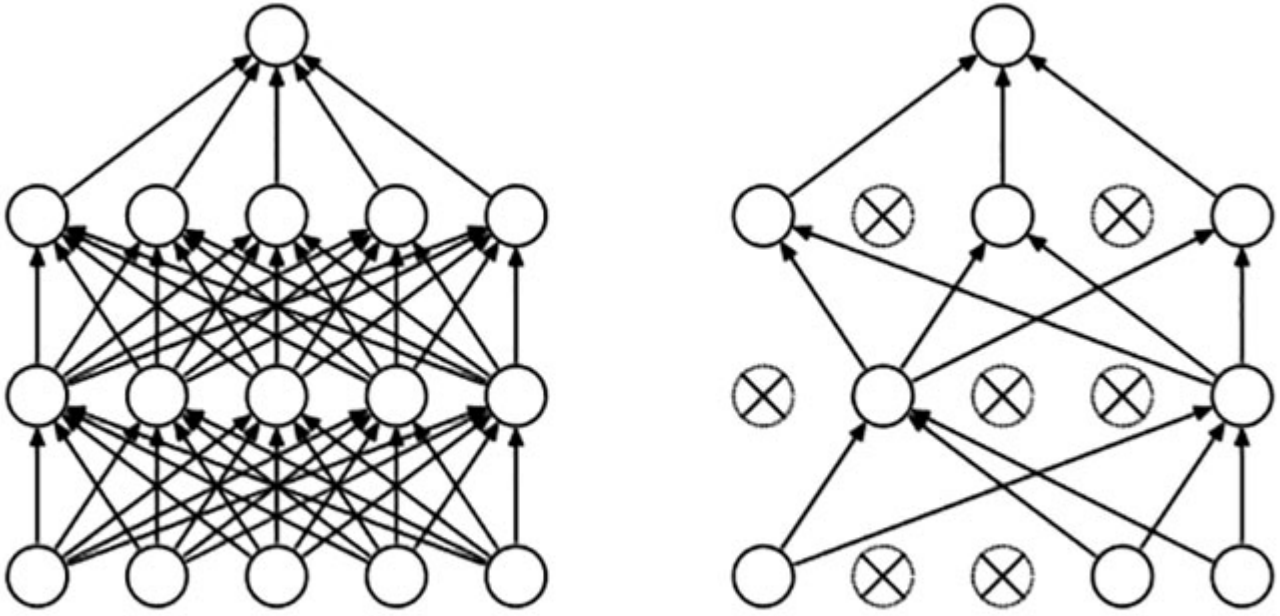
# 创新

## 2. Dropout

- dropout机制即训练过程中神经元有一定机率失活，不再参与训练的过程。通俗地可以解释为网络训练时用较少的神经元，而在实际预测时有更多的神经元工作（训练时增加难度而考试时却比较容易），如此能够得到更准确的结果。

# 创新

## 2. Dropout



(a) Standard Neural Net

(b) After applying dropout.

# 创新

## 3. Batch Normalization

Batch Normalization（批量标准化，简称BN）是近些年来深度学习优化中一个重要的手段。BN的优点主要有：加速训练过程、可以使用较大的学习率、允许在深层网络中使用易导致梯度消失的激活函数和具有一定正则化效果四点

# 创新

## 3. Batch Normalization

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

https://arxiv.org/pdf/1502.03167.pdf

# 网络结构

**ResNet风格网络图**
**描述性**

Input 3 channels -> output 64 channels

Input 64 channels -> output 64 channels

Max pooling 2×2

Input 64 channels -> output 128 channels

Input 128 channels -> output 128 channels

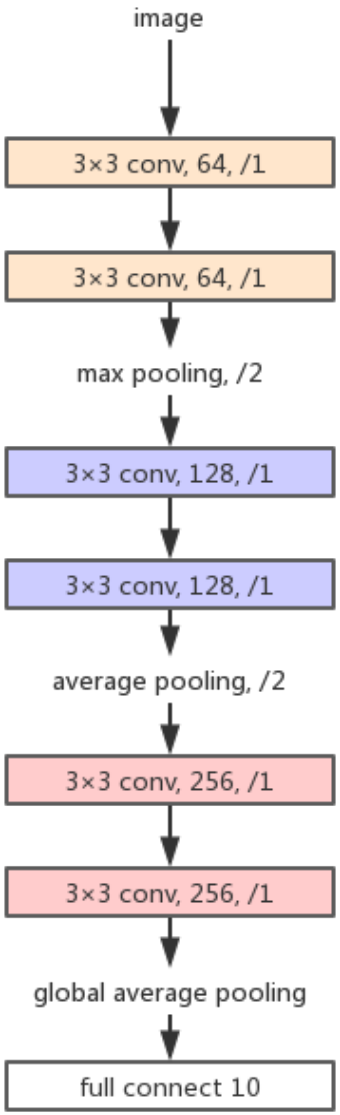
Average pooling 2×2

Input 128 channels -> output 256 channels

Input 256 channels -> output 256 channels

Global average pooling

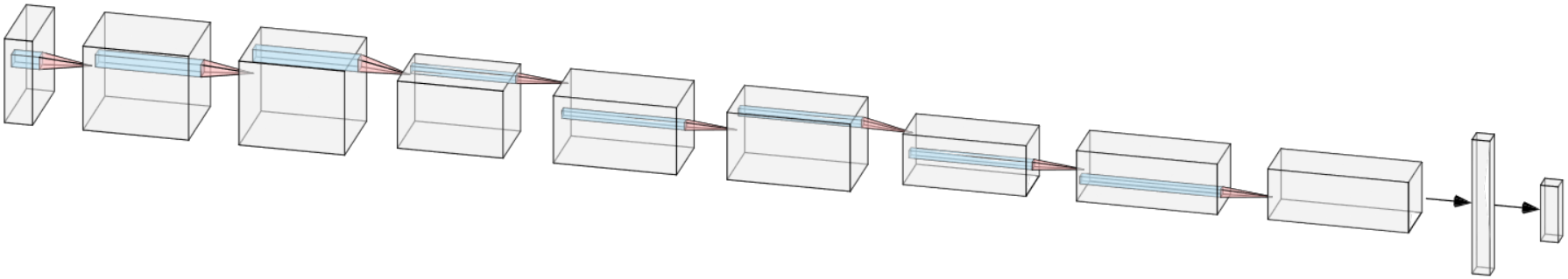
Full connect output 10



# 网络结构

**AlexNet风格的网络图**

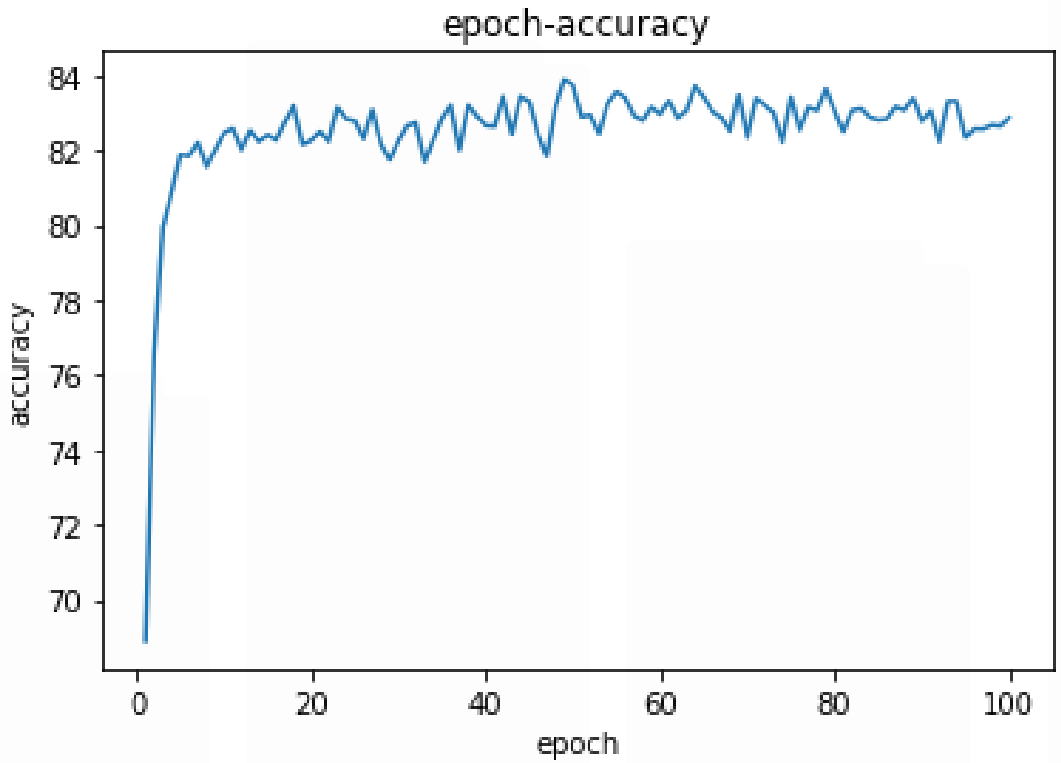
从中能够较为直观地看出特征图
经过每一层后的变化情况



# 实现流程与结果

## 经典模型（CNN+全连接层）

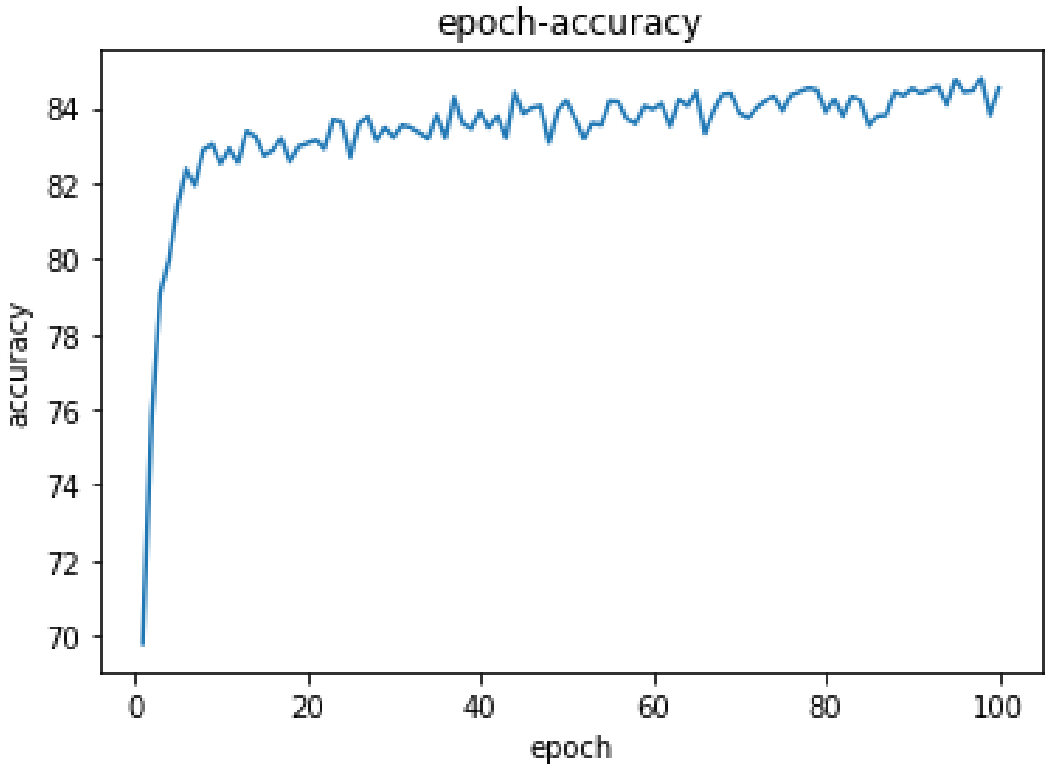
```
Accuracy of the network
on test set: 82 %
Training Time: 2511.1305
Accuracy of plane : 86 %
Accuracy of   car : 96 %
Accuracy of  bird : 73 %
Accuracy of   cat : 65 %
Accuracy of  deer : 77 %
Accuracy of   dog : 71 %
Accuracy of  frog : 90 %
Accuracy of horse : 84 %
Accuracy of  ship : 95 %
Accuracy of truck : 93 %
```



# 实现流程与结果

## 加入dropout机制

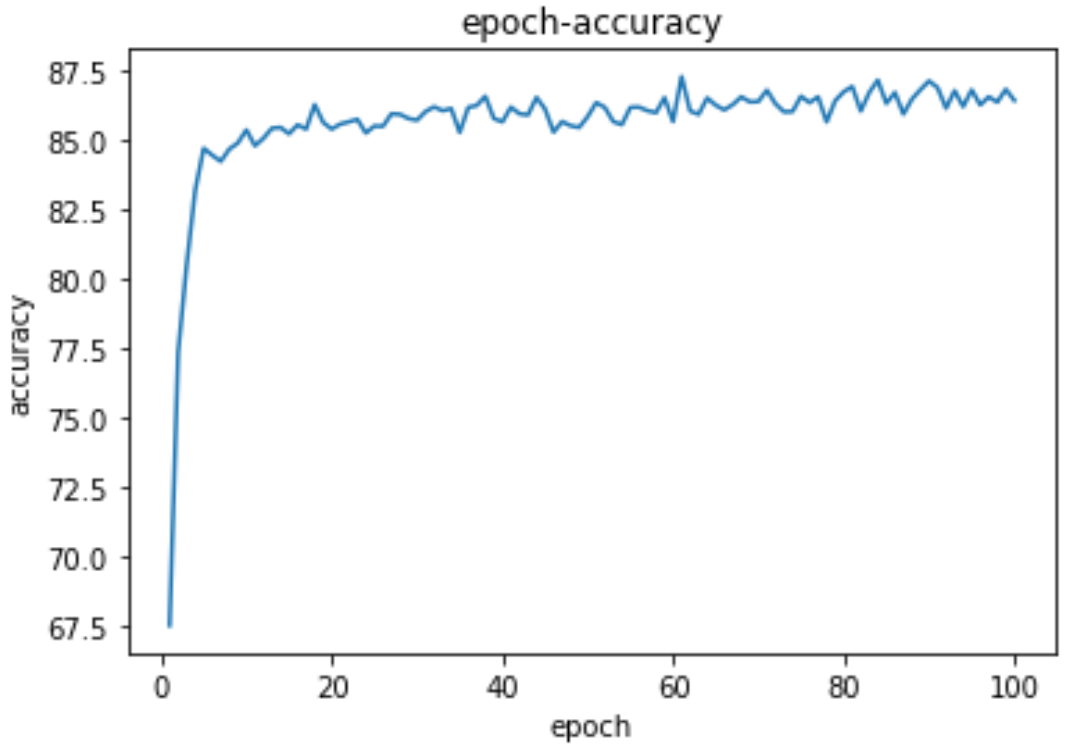
```
Accuracy of the network
on test set: 84 %
Training Time: 2746.8287
Accuracy of plane : 88 %
Accuracy of   car : 93 %
Accuracy of  bird : 89 %
Accuracy of   cat : 72 %
Accuracy of  deer : 81 %
Accuracy of   dog : 75 %
Accuracy of  frog : 92 %
Accuracy of horse : 87 %
Accuracy of  ship : 85 %
Accuracy of truck : 96 %
```



# 实现流程与结果

## 加入全局平均池化（GAP）

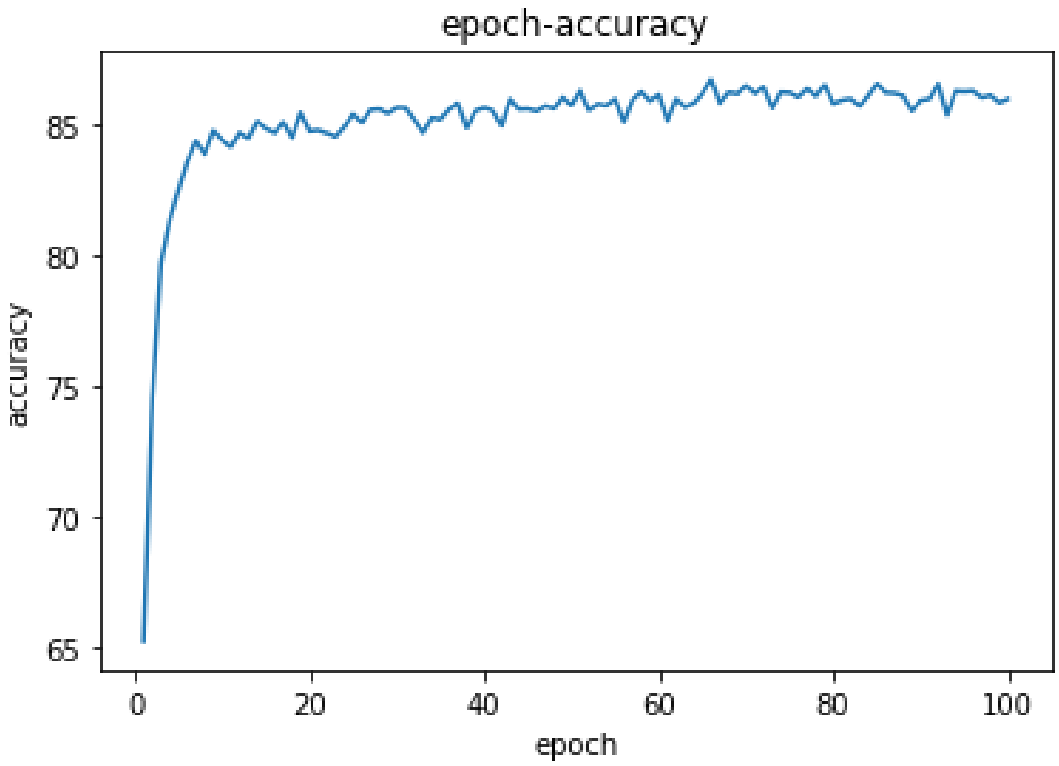
```
Accuracy of the network
on test set: 87 %
Training Time: 2495.2561
Accuracy of plane : 90 %
Accuracy of  car : 100 %
Accuracy of  bird : 76 %
Accuracy of  cat : 67 %
Accuracy of  deer : 87 %
Accuracy of  dog : 75 %
Accuracy of  frog : 83 %
Accuracy of horse : 87 %
Accuracy of  ship : 93 %
Accuracy of truck : 98 %
```



# 实现流程与结果

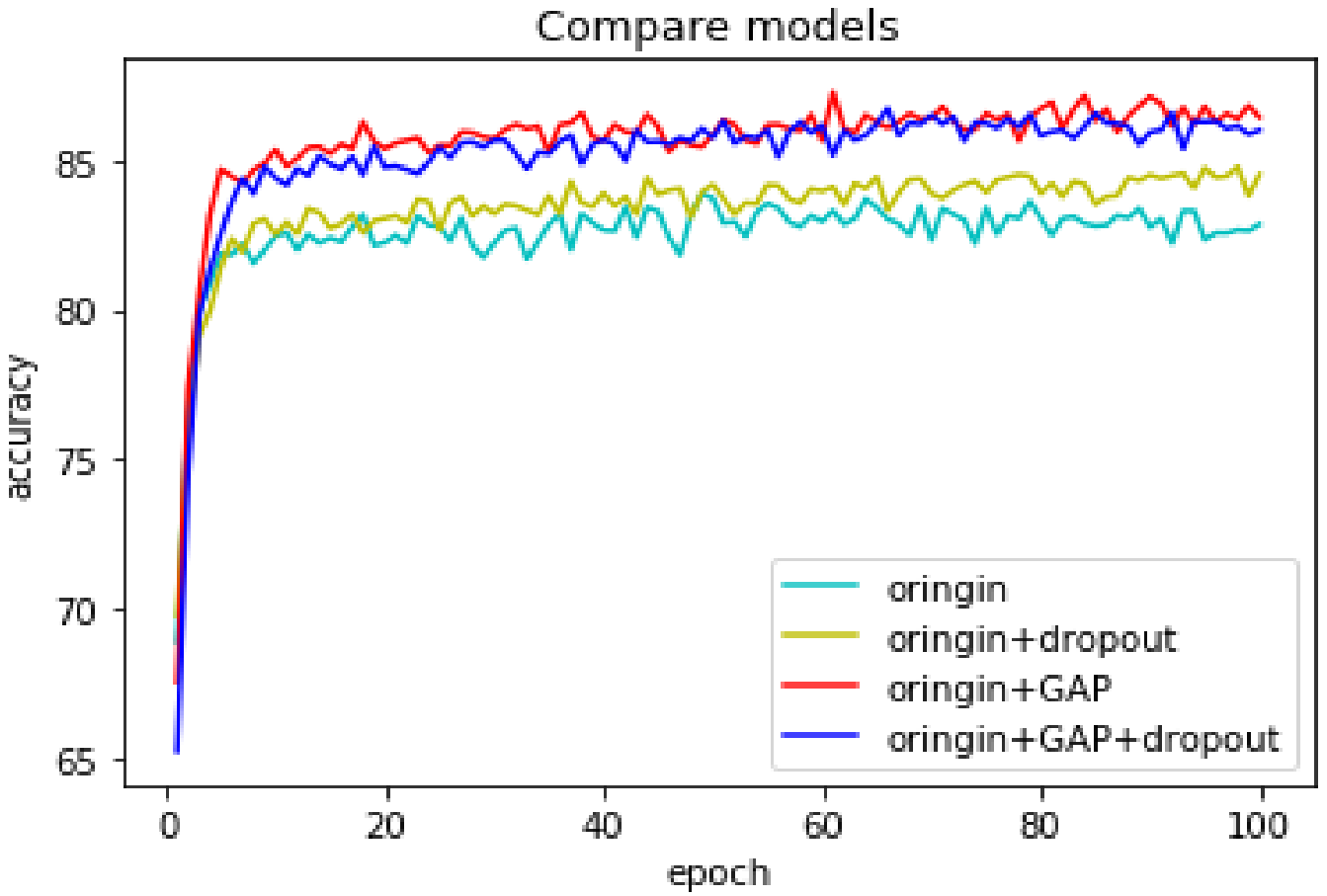
## 加入全局平均池化（GAP）和dropout

```
Accuracy of the network
on test set : 86 %
Training Time: 2606.0361
Accuracy of plane : 93 %
Accuracy of   car : 93 %
Accuracy of  bird : 73 %
Accuracy of   cat : 74 %
Accuracy of  deer : 79 %
Accuracy of   dog : 81 %
Accuracy of  frog : 88 %
Accuracy of horse : 93 %
Accuracy of  ship : 85 %
Accuracy of truck : 93 %
```



# 实现流程与结果

四种结构对比



# 参考文献

**[1]** Siamese Recurrent Architecture for learning Sentence Similarity

[https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12195/12023]

**[2]** Deep Residual Learning for Image Recognition

[https://arxiv.org/pdf/1512.03385.pdf]

**[3]** Network In Network

[https://arxiv.org/abs/1312.4400]

**[4]** Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[https://arxiv.org/pdf/1502.03167.pdf]


# THANKS!