**实验报告**

**06117113 吴中行**

搭建《Arrhythmia detection using deep convolutional neural network with long duration ECG signals》中神经网络并完成网络训练。使用Qkeras对1中训练后网络进行量化.

目录

[一、 论文 1](#_Toc59732054)

[二、 数据集 2](#_Toc59732055)

[三、 实验环境 3](#_Toc59732056)

[四、 实验思路 3](#_Toc59732057)

[五、 神经网络结构 4](#_Toc59732058)

[六、 参数选择 6](#_Toc59732059)

[七、 实验结果 7](#_Toc59732060)

[八、 量化 8](#_Toc59732061)

[九、 调参 9](#_Toc59732062)

[十、 量化后的结果 11](#_Toc59732063)

[附录 12](#_Toc59732064)

## 论文

《Arrhythmia detection using deep convolutional neural network with long duration ECG signals》

*Abstract：*

本文提出了一种基于长期心电图（ECG）信号分析的心律失常（17类）检测新的深度学习方法。我们研究的目的是设计一种基于深度学习的新方法，以高效，快速地对心律不齐进行分类。

研究基于MIT-BIH心律失常数据库中的1000条ECG信号片段。应用基于10 s ECG信号片段分析的方法（不是单个QRS复合信号）（平均而言，分类/分析少13倍）。设计了完整的end to end结构，而不是传统方法中使用的手工特征提取和选择。

我们的主要贡献是设计一种新的1D卷积神经网络模型（1D-CNN）。提出的方法是1）高效，2）快速（实时分类）3）非复杂和4）使用简单（组合特征提取和选择以及在一个阶段进行分类）。

Deep 1D-CNN在17个心律不齐症疾病（分类）中的总体识别准确度为91.33％，每个样本的分类时间为0.015秒。与当前的研究相比，我们的结果是迄今为止最好的结果之一，我们的解决方案可以在移动设备和云计算中实现。

## 数据集

**数据集*MIT-BIH***

数据集获取：

1 https://archive.physionet.org/physiobank/database/html/mitdbdir/records.htm

2

https://archive.physionet.org/cgi-bin/atm/ATM

图片包含 屏幕截图

描述已自动生成

按照论文选择360Hz采样频率，10s，标准格式

**数据集介绍：**

MIT —— Massachusetts Institute of Technology

BIH —— Beth Israel Hospital

从这能整个地获取这个 含48 个充分标注的30分钟长度的ECG  数据集。MIT-BIH 心律失常数据库目录也可以在线使用

自1975年以来，Beth Israel Deaconess和MIT一直支持我们自己的心律失常分析和相关课题的研究。研究取得的最大的成果之一是MIT-BIH心律失常数据库，我们在1980年完成数据库创建并开始发布。

MIT-BIH 心律失常数据库包含48个30分钟长度双通道动态心电图记录片段，这些片段是从 1975 - 1979年间BIH心律失常实验室的47名研究对象那里获得的。

论文中和本实验中ECG信号长度为10s，采样频率是360Hz，也就是也有每个样本有3600个数据.论文中做了三次实验包括13Classes 15Classes 17Classes三种，我只做了17Classes.

## 实验环境

VMware虚拟机Ubuntu18.04

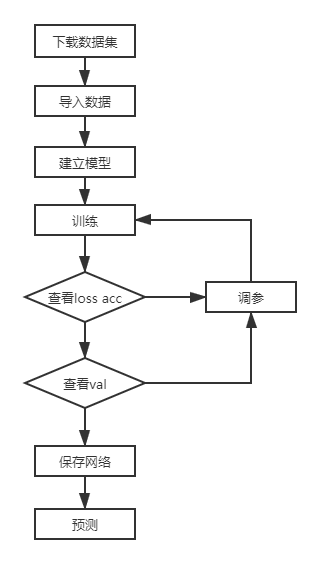
I7-9750H 2 Core 4 TH

RAM 4GB

TensorFlow 2.0 Keras 2.3.0

IDE jupyter notebook

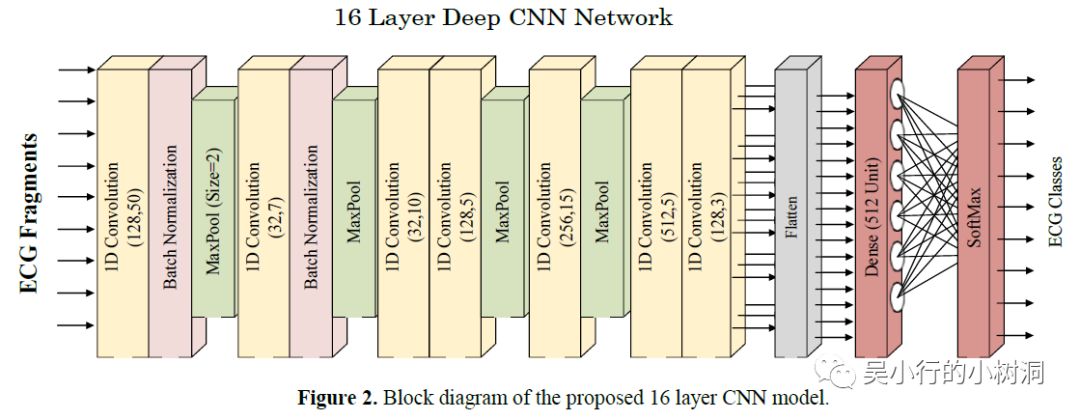
## 实验思路



## 神经网络结构

根据论文神经网络框架，搭建如下网络：

由7个一维卷积层，4个最大化池，2个标准化层，1个平展层，1个全连接层以及最后的多分类层构成。

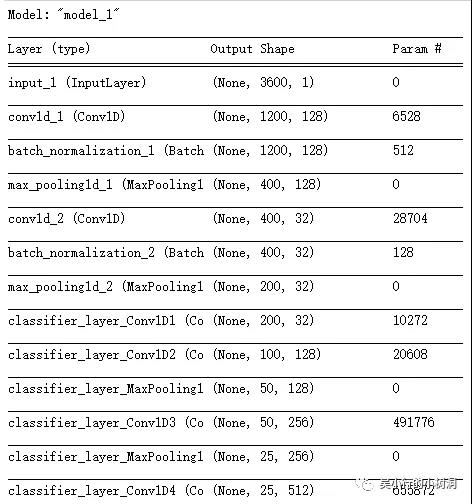


图片包含 文字, 收据

描述已自动生成

padding 默认 =  valid，但这个填充边界会导致负维度的产生，论文里也没写是same还是valid，我改成了same才通过。还有就是注意channal\_first 还是channal\_last。

我分别用Sequence和Model构建了上述模型。

图片包含 屏幕截图

描述已自动生成

## 参数选择

**关于优化器的选择：**

        对于多分类问题，SGD应该是可以的，但是我按照论文选择了更优的Adam，因为Adam下降更快。

**学习率lr：**

        越大训练时间越短。但是过大loss会跑飞或者直接nan，过小又会陷入局部最小值，loss也降不下来。一般lr的值在0.001~0.01之间。可以乘除2，5进行调整。一开始用lr = 0.001，发现loss降低到0.01xx降不下去了，应该是陷入了局部最小值，我改成0.002后就能下降到1e-05了。

**BATCH\_SIZE ：**

        和训练时间没关系，一般选择**8/16/32**。

        Batch\_Size 太小，算法不收敛。

        随着 Batch\_Size 增大，处理相同数据量的速度越快。

        随着 Batch\_Size 增大，达到相同精度所需要的 epoch 数量越来越多。

        由于上述两种因素的矛盾， Batch\_Size 增大到某个时候，达到时间上的最优。

        由于最终收敛精度会陷入不同的局部极值，因此 Batch\_Size 增大到某些时候，达到最终收敛精度上的最优。

我一开始设置的太大了，100，50个epochs精度下降的很慢。后来改成16。

**EPOCHS：**

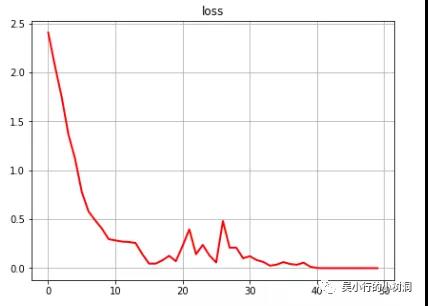
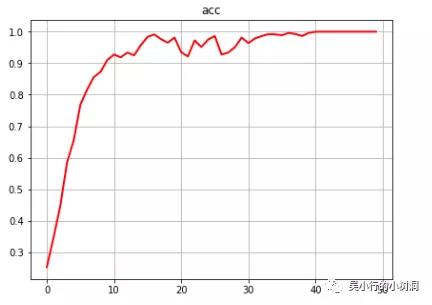
        训练轮数，一般先写个大点的值，看到多少轮，loss和acc基本不变了，然后减小到那个值就行。

参数列表

|  |  |
| --- | --- |
| **类别** | 17个类别 |
| **总样本数** | 1000 |
| **训练集** | 800 |
| **验证集** | 200 |
| **优化器** | Adam |
| **学习率** | 0.002 |
| **目标函数** | categorical\_crossentropy（与softmax一致） |
| **BATCH\_SIZE** | 16 |
| **EPOCHS** | 50 |

## 实验结果

训练集的loss & acc curve



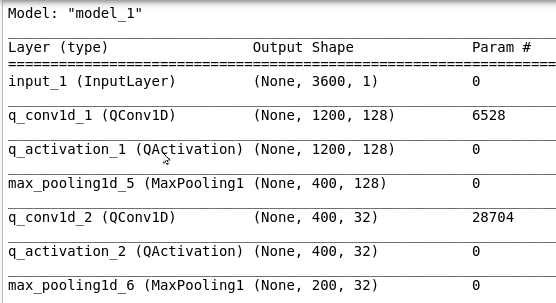
在验证集上的表现：

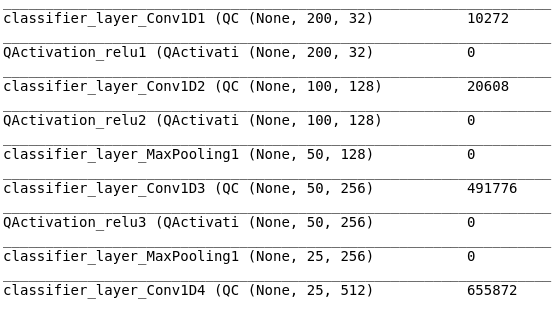


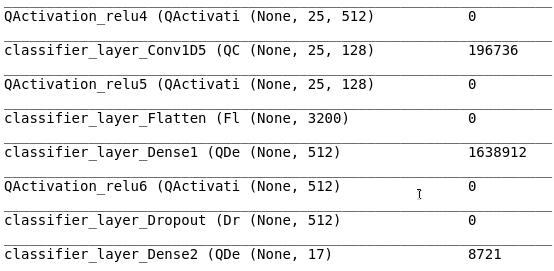
基本与论文中的17Classes的91.33%一致。

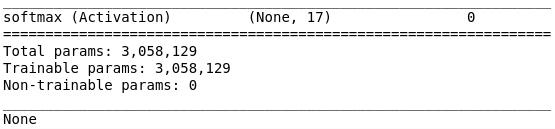
## 量化

量化之后的网络模型：









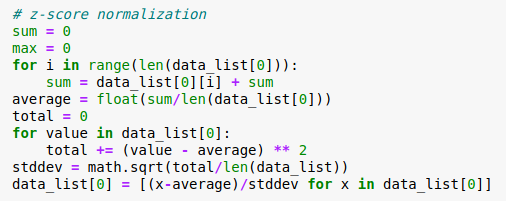
在原有网络的基础上：

Conv1D → QConv1D

Activation → QActivation

Dense → QDense

因为在Qkeras0.5.0中没有BatchNormalization对应的QBatchNormalization，我将网络中的BatchNormalization层去除，对应的在数据预处理阶段加上z-score Normalization



Conv层、Dense层增加：

kernel\_quantizer=quantized\_bits(4,0,1)，

bias\_quantizer=quantized\_bits(4,0,1),

Conv中的Activation删除，后面增加：

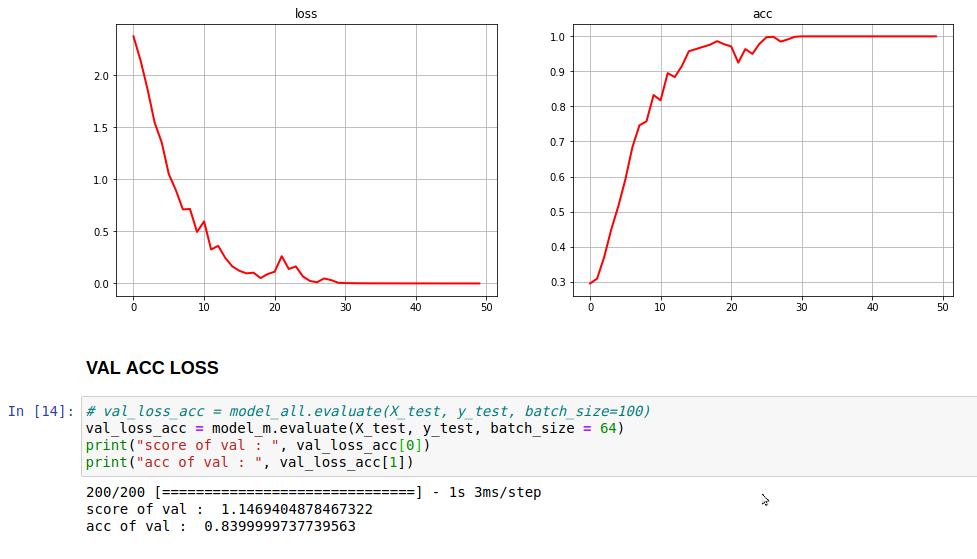
QActivation("quantized\_relu(4,0)", name = 'QActivation\_relu5')(x)

## 调参

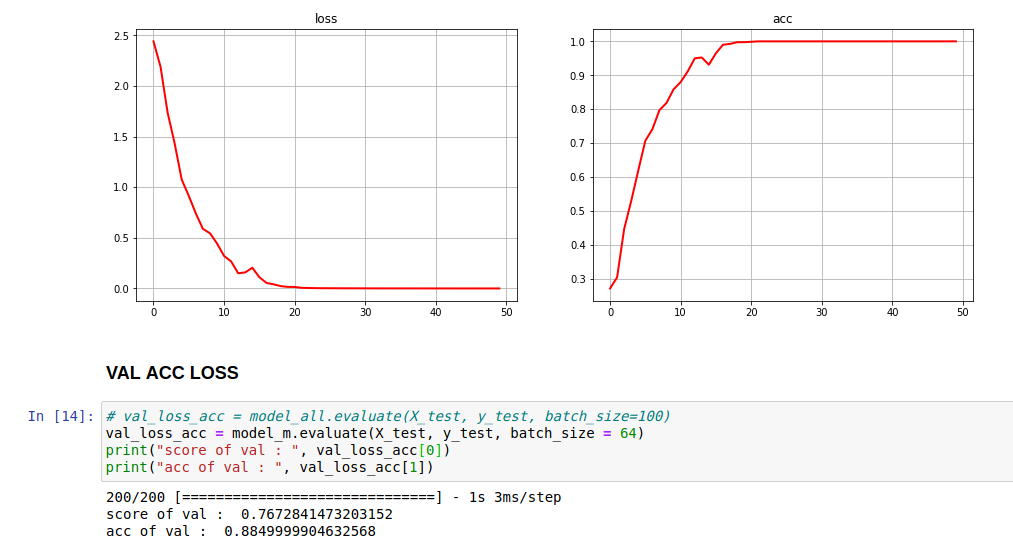
1. 首先，确定位宽，考虑网络大小，优先确定位宽**4bit**
2. 优化器固定“**Adam**”，损失函数“**categorical\_crossentropy**”
3. 然后保持lr = 0.001，调整batch size

|  |  |
| --- | --- |
| Batch\_size | Acc of val(50epochs) |
| 8 | 0.83 |
| 16 | 0.88 |
| 32 | 0.86 |
| 64 | 0.79 |

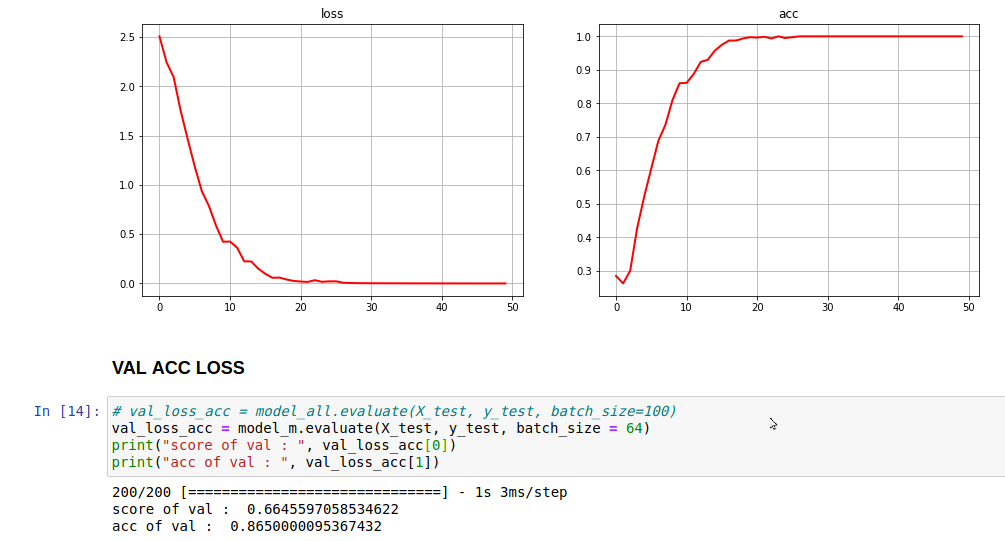
Batch\_size = 8:



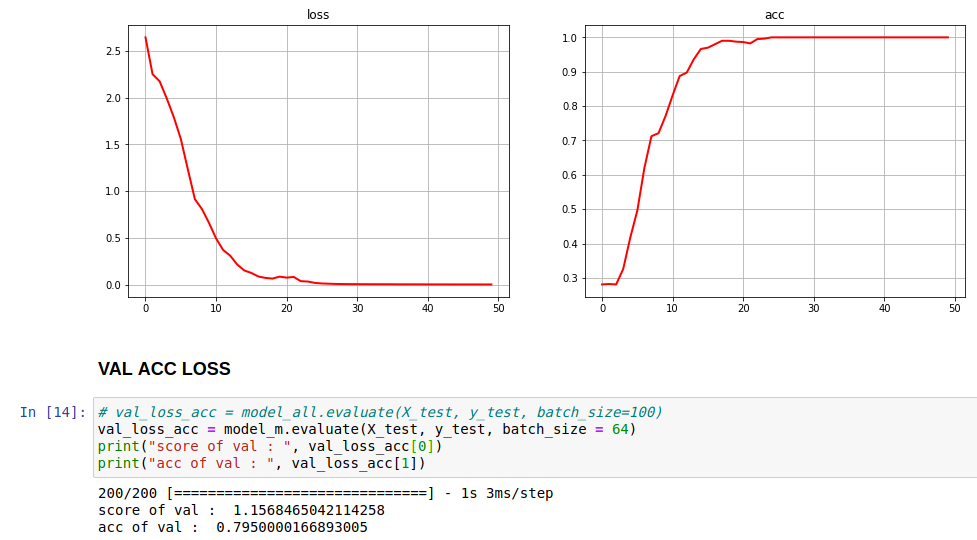
Batch\_size = 16:



Batch\_size = 32:



Batch\_size = 64时：



综上，选择**Batch\_szie=16**

1. 调整lr

|  |  |
| --- | --- |
| lr | Acc of val(50epochs) |
| 0.01 | 没收敛 |
| 0.005 | 0.85 |
| 0.002 | 0.87 |
| 0.001 | 0.88 |

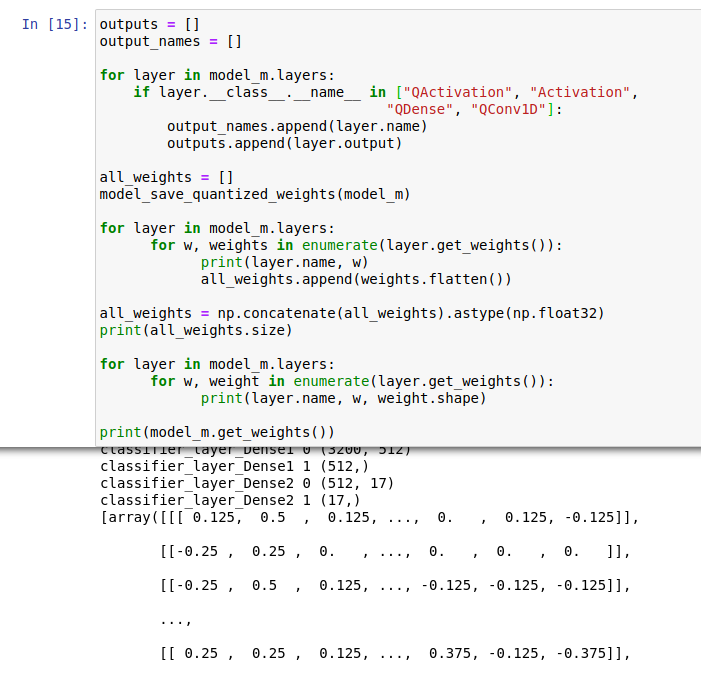
综上，还是选择**lr = 0.001**

1. 选择合适的轮数

观察100epochs的结果，发现50轮以后，几乎不下降，将最终的**epochs = 50**

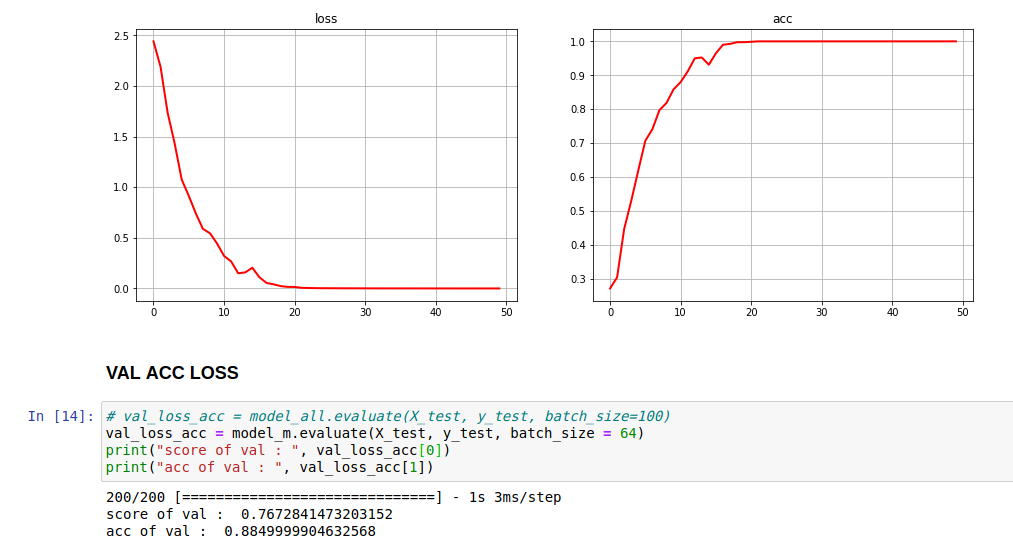
## 量化后的结果

查看参数：



确认参数已被量化，4bits位宽。

最终结果：



最终量化后的验证集的acc为88.49%

与量化前的结果91.33%相比没有下降很多。

## 附录

GitHub：

<https://github.com/SEU-wzx/ECGAI_1D-CNN.git>

