一 模型说明：



图1 模型诊断流程图

二 模型细节：

2.1 CNN\_model.py (搭建网络架构文件)

网络张量图

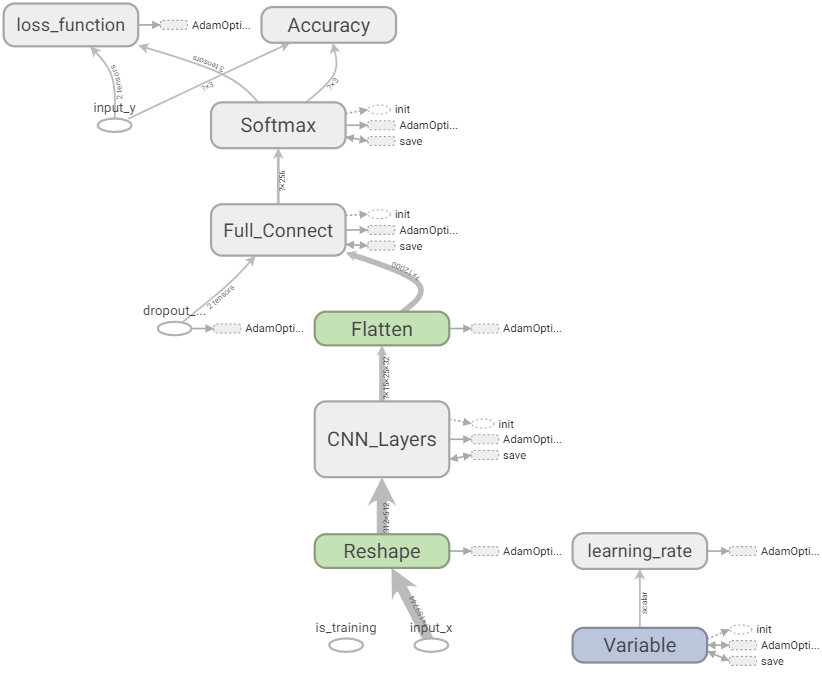


图2 模型CNN网络张量图

2.2 Config.py (配置模型超参数及网络层参数)

repeat = 3 重复训练3次，取最好结果

batch\_size\_list = [64, 128, 256, 512, 1024, 2048] 待选取的batchsize列表

require\_improvement = 20 如果超过20轮，精度没有提升，提前终止该种参数的训练

learning\_rate\_list = [0.005, 0.008, 0.01] 待选取的学习率列表

learning\_rate\_decay = 0.5 学习率指数衰减率

decay\_steps = 100 每训练一百个batchsize，学习率衰减一次

2.2.1 网络尺寸图(batchsize=1)

由于声发射原始信号数据量过大，本项目共采用两种尺寸模型，针对麦克风、振动和声发射有效值处理后的信号，采用如下网络尺寸

表1 振动、麦克风和声发射有效值处理后的信号的网络尺寸说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 网络层 | 特征图height | 特征图weight | 特征图depth | 卷积核height | 卷积核weight |
| 输入层 | 64 | 32 | 1 | 7 | 7 |
| 卷积层1 | 64 | 32 | 8 | 3 | 3 |
| 池化层1 | 31 | 15 | 8 | 3 | 3 |
| 卷积层2 | 31 | 15 | 16 | 3 | 3 |
| 池化层2 | 15 | 7 | 16 | 3 | 3 |
| 卷积层3 | 15 | 7 | 32 | 3 | 3 |
| 池化层3 | 7 | 3 | 32 |  |  |
| 全连接层1 | 672 | 1 |  |  |  |
| 全连接层2 | 64 | 1 |  |  |  |
| 输出层 | 3 | 1 |  |  |  |

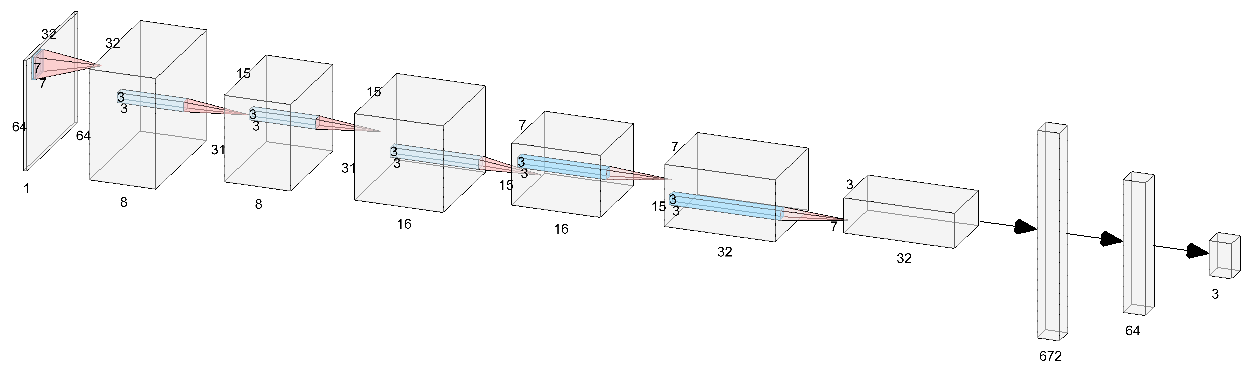


图3 振动、麦克风和声发射有效值处理后的信号的网络尺寸图

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 网络层 | 特征图height | 特征图weight | 特征图depth | 卷积核height | 卷积核weight |
| 输入层 | 512 | 312 | 1 | 25 | 25 |
| 卷积层1 | 512 | 312 | 8 | 5 | 5 |
| 池化层1 | 102 | 62 | 8 | 25 | 25 |
| 卷积层2 | 102 | 62 | 16 | 2 | 2 |
| 池化层2 | 51 | 31 | 16 | 25 | 25 |
| 卷积层3 | 51 | 31 | 32 | 2 | 2 |
| 池化层3 | 25 | 15 | 32 |  |  |
| 全连接层1 | 12000 | 1 |  |  |  |
| 全连接层2 | 256 | 1 |  |  |  |
| 输出层 | 3 | 1 |  |  |  |

表2 MT2声发射原始信号的网络尺寸说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 网络层 | 特征图height | 特征图weight | 特征图depth | 卷积核height | 卷积核weight |
| 输入层 | 512 | 312 | 1 | 25 | 25 |
| 卷积层1 | 512 | 312 | 8 | 5 | 5 |
| 池化层1 | 102 | 62 | 8 | 25 | 25 |
| 卷积层2 | 102 | 62 | 16 | 2 | 2 |
| 池化层2 | 51 | 31 | 16 | 25 | 25 |
| 卷积层3 | 51 | 31 | 32 | 2 | 2 |
| 池化层3 | 25 | 15 | 32 |  |  |
| 全连接层1 | 12000 | 1 |  |  |  |
| 全连接层2 | 256 | 1 |  |  |  |
| 输出层 | 3 | 1 |  |  |  |

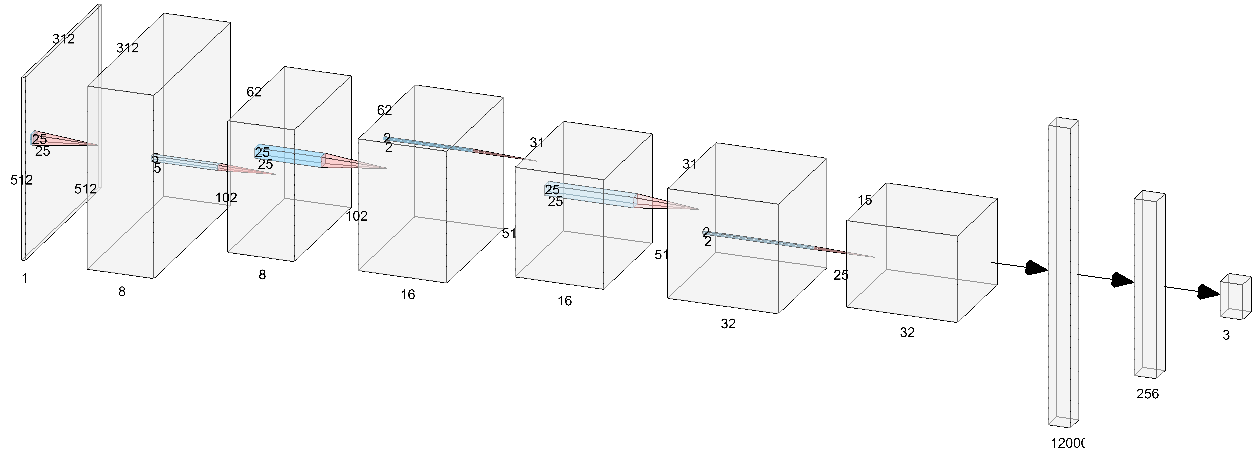


图4声发射原始信号的网络尺寸图

2.3 run\_model\_gpu.py (模型接口文件)

以kafka消费的数据中的传感器标签为唯一标识，模型分为在线测试和离线训练功能，

2.3.1 在线测试接口：

①def run\_test(self, x\_test, checkpoint\_path\_folder):

参数：

x\_test: 从kafka消费端读取每一个topic里的字典数据，依据标签值——传感器号，从相应的数据里截取N个数据点1）对于振动、麦克风、声发射有效值处理后的信号，N=2048 2) 对于声发射原始信号，N=159744

checkpoint\_path\_folder: 模型网络权重参数保存路径

②测试接口执行流程：

1. 加载网络模型参数checkpoint\_path\_folder，
2. 传入对应数据x\_test，
3. 模型输出诊断结果

2.3.2 离线训练接口：

①def run\_train(self, samples\_train, labels\_train, checkpoint\_path\_folder)

参数：

samples\_train: 收集测试结果出错的数据 x\_test, 条数达到500，二维数组形式1）对于振动、麦克风、声发射有效值处理后的信号 (500,2048) 2）对于声发射原始信号，(500,159744)

labels\_train: 收集测试结果出错的数据 x\_test 对应的真实故障标签, 条数达到500，二维数组形式 1） 对于振动、麦克风、声发射有效值处理后的信号 (500,3) 2）对于声发射原始信号，(500,3)

checkpoint\_path\_folder：模型网络权重参数保存路径

②训练接口执行流程：

1. 保存在线测试产生的错误结果
2. 待累积到指定数目，加载对应模型checkpoint\_path\_folder，
3. 传入累积数据（训练样本samples\_train，训练标签labels\_train），
4. 更新模型网络参数

2.4 模型结果文件：

每种模型训练后得到checkpoints(网络权重参数文件)、data\_processing\*(数据预处理文件，得到train\_data, train\_label, test\_data, test\_label), logdir(模型训练日志)、CNN\_Model\_results(保存损失曲线和混淆矩阵\*.png、测试结果test\_result.json、训练与验证结果train\_and\_valid\_result.json)

2.5 模型训练结果



2.6 模型说明

表3 模型说明



2.7 UI界面部分结果

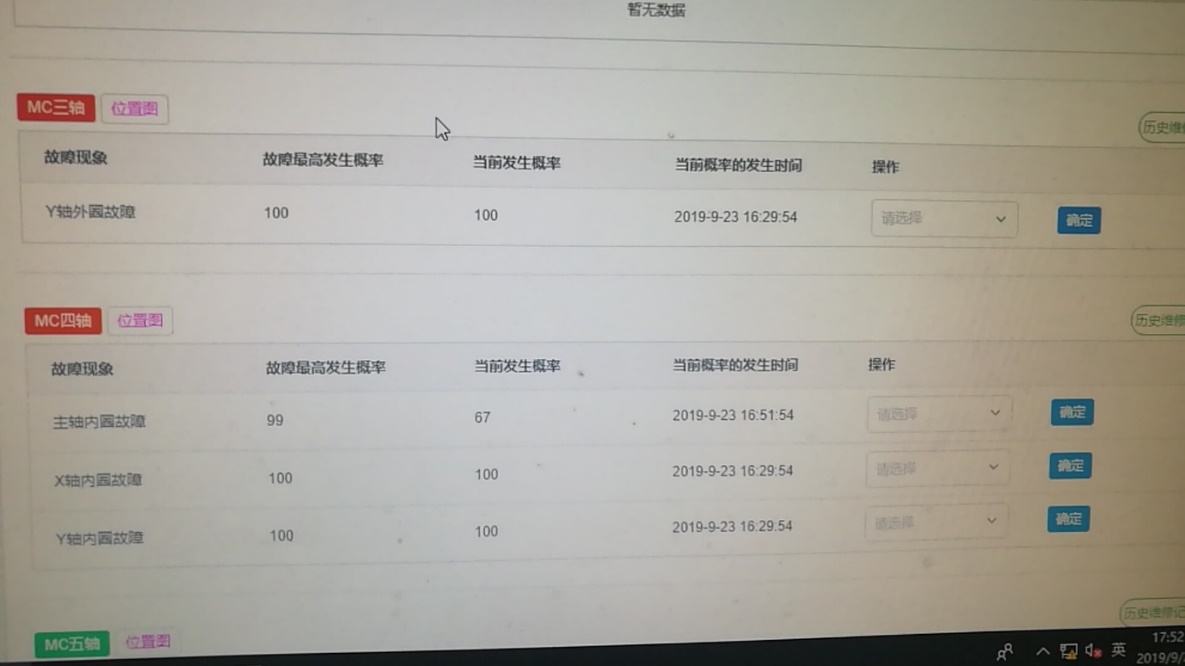


图5 模型预测结果输出UI端

三 算法交互three\_model文件夹说明：

统一口径：

1：在线、online、在线学习、测试模型 、数据源是kafka、 模型方法是run\_test

2：离线、offline、离线学习、训练模型 、数据源是数据库、 模型方法是run\_train

需要注意的几个文件

checkpoints 模型对应参数存放文件夹

models 模型依赖文件存放文件夹

utils/sensor\_checkpoint.json 每个sensor\_id对应的checkpoints相对目录

utils/esgyn\_config.json 数据库配置及kafka集群地址

utils/model\_util.py 根据sensor\_id确定checkpoints参数目录和models模型目录

第25行，因为上面的json文件中放的是相对目录，尽量使用绝对路径，所以加一个home\_path

第26行，self.sensor\_model 每个sensor\_id 对应的模型

--调用的文件路径尽量都要使用绝对路径，不管是模型文件还是其他

--尽量还放到之前的位置，因为web界面会调用dataflow\_remote\_run\_gpu.py文件

--如果想要放到其他位置，所以的路径都要改

dataflow\_online\_run.py 在线学习

参看第19行注释

参看第28行注释

说明：在线学习中从kafka消费数据，如果只启动一个当前文件，只有一个进程一个线程，所以性能可能会差，尤其是声发射的数据量很大

可以复制多个这样的文件，运行多个，只需要注意多个文件中 \*\* kafka的topic一定不能一样 \*\*，尽量做到负载均衡

schedule\_offline\_run.py 离线学习

参看112行注释

四个简单的启停shell脚本，想用就用，不想用根据自己的习惯都可以

补充说明：run\_train方法需要一个版本的返回值，也就是当前的模型属于哪个版本的

其实就是一个字符串，比如说 "v1.1"、 "v1.2" 很简单的这种，不要太复杂就行

四 模型运行环境：

运行模型依赖环境在nadder4主机上的anaconda下的虚拟环境tensorflowGPU1.9中，具体依赖包（requirement.txt文件内容）如下：

absl-py==0.7.1

astor==0.8.0

avro-python3==1.8.0

certifi==2018.8.24

gast==0.2.2

grpcio==1.22.0

kafka-python==1.4.6

Markdown==3.1.1

numpy==1.16.4

protobuf==3.9.0

scikit-learn==0.20.3

scipy==1.2.1

six==1.12.0

tensorboard==1.9.0

tensorflow-gpu==1.9.0

termcolor==1.1.0

Werkzeug==0.15.5