2019 年上海市青少年人工智能创新大赛

AI 海洋科技挑战赛 赛事介绍及高校组竞赛规则

一. 赛事背景

为认真贯彻落实党的十九大精神及习近平新时代中国特色社会主义思想，深入学习习近平总书记关于科技创新和青少年工作的重要指示精神，全面落实党中央、国务院《新一代人工智能发展规划》及市委、团中央的各项部署，进一步推动上海共青团聚焦重点、集中力量，助力青少年人工智能创新领域的发展，营造青少年人工智能创新的热烈氛围，共青团上海市委员会联合上海市经济和信息化委员会、上海市教育委员会、上海市科学技术委员会共同举办第二届上海市青少年人工智能创新大赛。

二. 组织机构

指导单位：共青团上海市委员会、上海市经济和信息化委员会、上海市教育委员会、上海市科学技术委员会

主办单位：上海市青少年活动中心、青年报社

协办单位：上海少年科学院、黄浦区青少年科技中心

支持单位：格顿教育科技

媒体支持：人民日报社民生网文化教育强国论坛、青年报·青春上海、萌动上海、东方网、人民网等

三. 赛事主题

人工智能与海洋生态

四. 参赛对象

(一) 参赛者

面向全市高校在校学生，每支参赛队伍只能选择一条赛道。参赛队伍可由个人申报，也可由团队申报，每队不超过 3 人。

(二) 赛道设计

1. 海洋科技算法组
2. 海洋科技工程组

五. 赛事简介

赛事分为海洋生物识别 AI 算法组和海洋航行器 AI 工程组比赛。参赛队伍着眼于水下通讯、海洋污染治理、海难救援等领域，通过比赛的形式将人工智能与海洋生物保护等实际应用结合起来，体现人工智能赋能经济和社会的价值。详见附件。

(一) 算法组

本赛道分为预选和决赛两阶段，赛题为鱼脸识别。鱼脸识别是根据鱼的特征来识别鱼类，从而获取更准确的海洋生物数量，帮助渔业部门精确掌握海洋生物资源的增长、减少或迁移的情况，将人工智能与海洋生物保护实际应用结合起来。

(二) 工程组

本赛道分为预选和决赛两阶段，赛题为完成一个人工智能海洋航行器设计方案。参赛队伍可着眼于水下通讯、海洋污染治理、海难救援等领域，从预防、监控、治理等不同角度来设计人工智能海洋航行器，设计方案应体现功能原理创新或总体布局创新。

六. 赛程安排

1. 大赛启动仪式（5 月 31 日）

第二届上海市青少年人工智能创新大赛启动仪式将在上海市青少年活动中心举行，同时向各区、高校等团组织下发大赛通知。大赛将开通线上报名系统，并在“青春上海”、“萌动上海”等微信公众号发布赛事报名信息。

2. 报名（7 月 12 日-7 月 22 日）

报名表需以电子版的形式在线提交，或以纸质版的形式现场提交，截止时间为 2019 年 7 月 22 日

22:00。

3. 预选作品提交（7 月 24 日-7 月 26 日）

预选作品以网络形式提交申报，截止时间为 2019 年 7 月 26 日 22:00。组委会将会根据预选成绩，于 2019 年 7 月 29 日 20:00 前公布进入决赛答辩的名单。

4. 决赛现场答辩（8 月 6 日-8 月 8 日）

组委会将安排入围决赛的参赛队进行答辩，现场答辩各队总时不超过 20 分钟，每支队伍的作品阐述时间不超过 10 分钟。

5. 展示及颁奖（8 月下旬）

举行颁奖仪式，向各奖项的获奖队伍颁发荣誉，并展示各组获奖作品风采。

七. 评奖规则

按照本项赛事报名总量的 30%评选出各赛道奖项，一等奖（5%）、二等奖（10%）、三等奖（15%），由主办方统一颁发证书。

八. 附件（高校组竞赛规则）

附件 1：算法组参赛指南（高校组）

附件 2：开发环境配置建议

附件 3：常用神经网络模型参考

附件 1

算法组参赛指南（高校组）

一. 赛题：鱼脸识别挑战赛

参赛者根据给定的 50 个类别海洋鱼类数据集——seafishes-50，设计和训练鱼脸识别算法。

(一) 任务说明

近日，我国海洋科考队在一座远离大陆海岛的附近水域中，发现一处神秘的海洋王国，那里生活着种类繁多的海洋生物：有成群嬉戏的海豚，有五彩缤纷的珊瑚，还有追逐海浪的海豹……为了解海洋生物栖息地的情况，科考队派出“小白龙号”水下机器人潜入海底。它的任务是调查海洋生物种群数量与分布，为海洋科考提供科学数据。为了完成水下探测任务，科考队打算给小白龙安装一副智能装备——利用水下摄像机拍摄海洋生物照片，识别出海洋生物种类。

科考队派遣小白龙实施第三项识别任务——识别 50 种海洋鱼类。为此，参赛者需要设计神经网络算法，使用 seafishes-50 数据集对算法进行训练，实现海洋鱼类的精准分类。

该项任务的挑战性在于：seafishes-50 数据集各类之间的样本数量不平衡（imbalance），有的类别的样本数几百张，有的类别样本数仅有几十张。参赛者需要针对数据集类间不平衡以及单类样本数较少的特点，从头训练一个神经网络。

(二) 算法设计任务

参赛者设计的算法应具有训练、验证和预测等功能。参赛者可基于常用的开源神经网络架构，例如 LeNet5、AlexNet、VGG 构建模型，也可以自行设计神经网络架构。大赛组委会鼓励参赛者自行设计算法，将对自行设计的算法给与相应的加分奖励。

(三) 提交内容

参赛者向竞赛评审委员会提交以下内容：

1. 项目申报书

2. 算法完整源代码（附在申报书中）

3. 保存完整的神经网络序贯模型及权重的 keras 模型文件(.h5 格式文件)

(四) 算法设计规则

参赛者须遵循以下设计规则：

1. 须使用 python 编程语言实现算法，基于 tensorflow 和 keras 神经网络库，不符合上述要求的视为无效代码；
2. 提交的代码可在大赛指定的评估平台上运行，并给出完整的评测报告；
3. 神经网络模型设计采用序贯模型（Sequential）；
4. 本次竞赛不允许使用迁移学习技术。

(五) 数据集下载

本项竞赛用于训练算法的数据集为 seafishes-50，数据集可从大赛官方平台下载。 下载文件解压后，可看到每种鱼类的样本存放在单独的文件夹下。

免责声明：竞赛数据集的部分图像来自互联网搜索引擎提供的缩略图或 Urls 链接，上述图像数据仅限于本次赛事使用，请勿用于其它商业用途。未经许可的传播行为由当事人承担全部责任，与本赛事和组委会无关。

二. 评分细则

(一) 算法评估（60 分）

1. 评价指标

竞赛评审委员会使用 scikit-learn 机器学习库的 metrics 方法库评估参赛者的算法性能，并给出量化结果。选用 F1-Score 指标即：

$$F_1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

其中，$Precision = \frac{TP}{TP + FP}$，$Recall = \frac{TP}{TP + FN}$

TP：True Positive,被判定为正样本，事实上也是正样本。

TN：True Negative,被判定为负样本，事实上也是负样本。

FP：False Positive,被判定为正样本，但事实上是负样本。

FN：False Negative,被判定为负样本，但事实上是正样本。

2. 计算得分

算法评估成绩满分为 60 分，按以下公式计算参赛者的得分：

评估得分= 60*（F1-score – MinF1-score）/(MaxF1-score - MinF1-score)

- F1-score ----训练及评估参赛者的数据得到的 F1-score
- MinF1-score----所有参赛者中最低的 F1-score
- MaxF1-score----所有参赛者中最高的 F1-score

3. 加分

若不采用开源算法，而是采用自行设计的算法参加竞赛，将给与参赛者 10-20 分的加分。

4. 评估数据集

为保证比赛公正性，采用统一验证数据集评估参赛算法，竞赛期间该数据集不对参赛者公开。竞赛结束后将公开数据集供参赛者研究。

(二) 现场答辩（40 分）

现场答辩环节，评委根据参赛者的表现现场打分。评委对参赛者表现评分的依据为以下四个方面：

1. 机器学习/深度学习原理的理解
2. 神经网络模型与识别算法的理解
3. 参赛者的动手实验能力
4. 非平衡数据集的理解与问题解决思路

附件 2

开发环境配置建议

本项竞赛建议开发环境配置为：

- Windows10
- Python3.7
- Tensorflow1.13.1
- Keras 2.2.4

Python+ Tensorflow+Keras 环境配置指南

https:/www.jianshu.com/p/6c3b1889b358

附件 3

常用神经网络模型参考

（1）AlexNet

```
import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout,Flatten
from keras.layers import Conv2D,MaxPool2D

# AlexNet
model = Sequential()
#第一段
model.add(Conv2D(filters=96, kernel_size=(11,11),
                 strides=(4,4), padding='valid',
                 input_shape=(resize,resize,3),
                 activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(3,3),
                       strides=(2,2),
                       padding='valid'))
#第二段
model.add(Conv2D(filters=256, kernel_size=(5,5),
```

```
                 strides=(1,1), padding='same',
                 activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(3,3),
                       strides=(2,2),
                       padding='valid'))
#第三段
model.add(Conv2D(filters=384, kernel_size=(3,3),
                 strides=(1,1), padding='same',
                 activation='relu'))
model.add(Conv2D(filters=384, kernel_size=(3,3),
                 strides=(1,1), padding='same',
                 activation='relu'))
model.add(Conv2D(filters=256, kernel_size=(3,3),
                 strides=(1,1), padding='same',
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(3,3),
                       strides=(2,2), padding='valid'))
#第四段
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))

# Output Layer
model.add(Dense(2))
model.add(Activation('softmax'))
model.summary()
```

(2) VGG16

```
import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout,Flatten
from keras.layers import Conv2D,MaxPool2D
from keras.utils import plot_model

# 定义输入
```

```
input_shape = (224,224,3) # RGB影像224x224（height,width,channel)

# 使用序贯模型(sequential)来定义
model = Sequential(name='vgg16-sequential')

# 第1个卷积区块(block1)
model.add(Conv2D(64,(3,3),padding='same',activation='relu',input_shape=input_shap
e,name='block1_conv1'))
model.add(Conv2D(64,(3,3),padding='same',activation='relu',name='block1_conv2'))
model.add(MaxPool2D((2,2),strides=(2,2),name='block1_pool'))

# 第2个卷积区块(block2)
model.add(Conv2D(128,(3,3),padding='same',activation='relu',name='block2_conv1'))
model.add(Conv2D(128,(3,3),padding='same',activation='relu',name='block2_conv2'))
model.add(MaxPool2D((2,2),strides=(2,2),name='block2_pool'))

# 第3个区块(block3)
model.add(Conv2D(256,(3,3),padding='same',activation='relu',name='block3_conv1'))
model.add(Conv2D(256,(3,3),padding='same',activation='relu',name='block3_conv2'))
model.add(Conv2D(256,(3,3),padding='same',activation='relu',name='block3_conv3'))
model.add(MaxPool2D((2,2),strides=(2,2),name='block3_pool'))

# 第4个区块(block4)
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block4_conv1'))
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block4_conv2'))
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block4_conv3'))
model.add(MaxPool2D((2,2),strides=(2,2),name='block4_pool'))

# 第5个区块(block5)
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block5_conv1'))
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block5_conv2'))
model.add(Conv2D(512,(3,3),padding='same',activation='relu',name='block5_conv3'))
model.add(MaxPool2D((2,2),strides=(2,2),name='block5_pool'))

# 前馈全连接区块
model.add(Flatten(name='flatten'))
model.add(Dense(4096,activation='relu',name='fc1'))
model.add(Dense(4096,activation='relu',name='fc2'))
model.add(Dense(1000,activation='softmax',name='predictions'))

# 打印网络结构
model.summary()
```

(3) LeNet-5

```
import keras
from keras.datasets import mnist
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense, Flatten
from keras.models import Sequential

model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5), activation='relu', input_shape=(28, 28,
1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```