实验四 动物图像识别

# 1.程序介绍

该程序为名称为动物图像识别，设计的思路为调用百度智能云平台API接口进行图像处理，本地加载处理识别图片，接受请求后返回的JSON文本参数，进行解析给出识别结果。这里利用Qt搭建界面，c++发送请求和接受应答。整体为用户给出动物图片，点击开始识别，会显示结果。

# 2.操作说明

点击文件选择打开，选择要识别的动物图像，这里以袋鼠照片为例，点击开始识别按钮就可以看到结果，如图1所示。

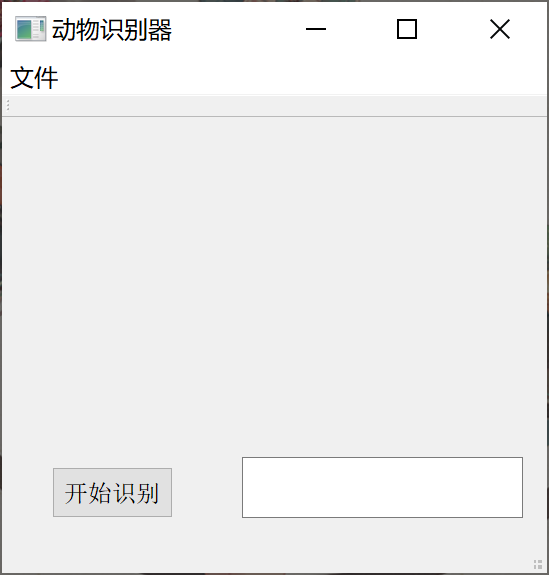
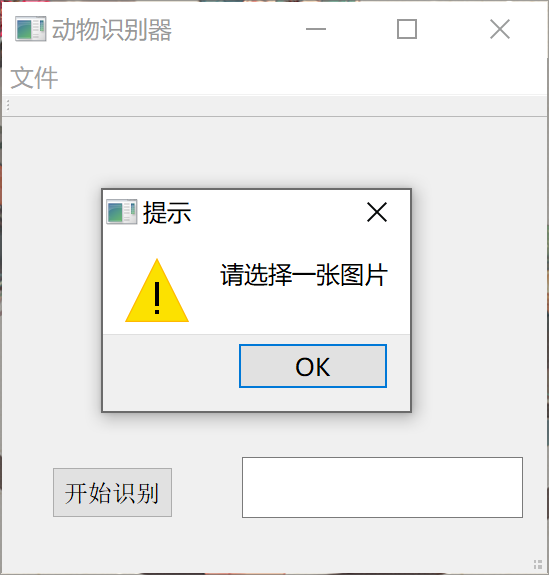
 

图 1操作展示图

# 3.设计理念

## 3.1设计目标

该项目是想设计一个可以自动识别动物图片的软件，对于人们不了解的动物给出其名称，这里采用百度AI平台的接口，利用其云数据学习的结果进行匹配，其运行界面如图2所示。



图 2动物图像识别页面

## 3.2 设计分析和算法分析

### 3.2.1整体思路

搭建一个图片处理的界面，进行图片的读取，这里图像识别主要借助百度云AI平台搭建项目服务器，进行访问请求，按照上传要求，先要将图片进行处理，转化为base64编码。第一次请求获得Access Token，第二次请求获得结果，解析结果输出。这里构建一个Image类和Http类，用来处理图片和发送请求。

### 3.2.2图像处理算法

将图像转化为base64，这里利用QByteArray类和QBuffer类来实现，具体代码如下：

|  |
| --- |
| #ifndef IMAGE\_H  #define IMAGE\_H  #include <QString>  #include <QImage>  #include <QByteArray>  #include <QBuffer>  #include <QTextCodec>  *class* **Image**  {  *public*:  **Image**();  *static* QByteArray **imageToBase64**(QString imgPath);  void **urlEncode**();  };  #endif *//* *IMAGE\_H* |

代码块 1 Image类头文件

|  |
| --- |
| #include "image.h"  Image::**Image**()  {  }  QByteArray Image::**imageToBase64**(QString imgPath)  {  QImage img(imgPath);  QByteArray ba;  *//用QByterArray构造QBuffer*  QBuffer buf(&ba);  buf.*open*(QIODevice::*WriteOnly*);  *//把img写入QBuffer*  img.save(&buf,"JPG");  *//对图片做base64编码，不包含编码头*  QByteArray base64 = ba.toBase64();  QTextCodec\* codec = QTextCodec::codecForName("UTF-8");  QByteArray imgDate = codec->fromUnicode(base64).toPercentEncoding();  *return* imgDate;  } |

代码块 2 Image类实现代码

### 3.3.3百度AI平台搭建

这里在百度AI开发平台注册，创建项目，如图3所示，在开发者文档中有API教程。

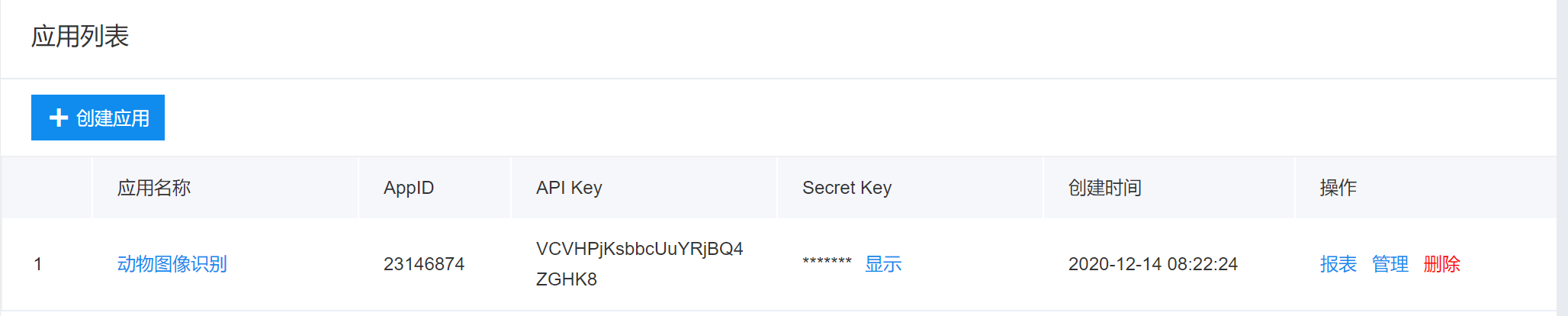


图 3 百度AI项目

以本次动物识别为例，请求方法如下图所示，将相关的网站参数用QString类进行记录。



图 4请求参数

请求代码示例如下：

|  |
| --- |
| curl -i -k 'https://aip.baidubce.com/rest/2.0/image-classify/v1/animal?access\_token=【调用鉴权接口获取的token】' --data 'image=【图片Base64编码，需UrlEncode】' -H 'Content-Type:application/x-www-form-urlencoded' |

对返回的参数进行识别，返回类型如下图所示，对返回的JSON文本进行解析，获得结果，返回示例如下：



图 5返回参数

|  |
| --- |
| HTTP/1.1 200 OK  x-bce-request-id: 73c4e74c-3101-4a00-bf44-fe246959c05e  Cache-Control: no-cache  Server: BWS  Date: Tue, 18 Oct 2016 02:21:01 GMT  Content-Type: application/json;charset=UTF-8  {  "log\_id": 7392482912853822863,  "result": [{  "score": "0.993811",  "name": "叉角羚",  "baike\_info": {  "baike\_url": "http://baike.baidu.com/item/%E5%8F%89%E8%A7%92%E7%BE%9A/801703",  "description": "叉角羚(学名：Antilocapra americana)：在角的中部角鞘有向前伸的分枝，故名。体型中等，体长1-1.5米，尾长7.5-10厘米，肩高81-104厘米，成体重36-60千克，雌体比雄体小；背面为红褐色，颈部有黑色鬃毛，腹部和臀部为白色，颊面部和颈部两侧有黑色块斑；毛被下面为绒毛，上覆以粗糙、质脆的长毛，由于某些皮肤肌的作用，能使其毛被呈不同角度，以利于保暖或散热。植食。叉角羚奔跑速度非常快，最高时速达100千米。一次跳跃可达3.5-6米。善游泳。夏季组成小群活动，冬季则集结成上百只的大群。为寻找食物和水源，一年中常进行几次迁移。性机警，视觉敏锐，能看到数千米外的物体。遇险时，臀部的白色毛能立起，向同伴告警。分布于北美洲。"  }  },  {  "score": "0.000289439",  "name": "印度羚"  },  {  "score": "0.000186248",  "name": "藏羚羊"  },  {  "score": "0.000147176",  "name": "跳羚"  },  {  "score": "0.000134434",  "name": "驯鹿"  },  {  "score": "9.86555e-05",  "name": "高鼻羚羊"  }]  } |

代码块 3 返回示例

### 3.3.4请求类实现

先导入一些网络请求头文件，利用QNetworkReply发送请求，先将请求内容进行拼接组合，确定请求Url和请求data。设置循环请求直到接受返回值。

|  |
| --- |
| #ifndef HTTP\_H  #define HTTP\_H  #include <QString>  #include <qnetwork.h>  #include <QtNetwork/QNetworkAccessManager>  #include <QtNetwork/QNetworkReply>  #include <QtNetwork/QNetworkRequest>  #include <QEventLoop>  #include <QJsonObject>  #include <QNetworkReply>  #include <QJsonDocument>  #include <QObject>  *const* QString BaiduTokenUrl = "https://aip.baidubce.com/oauth/2.0/token?grant\_type=client\_credentials&client\_id=%1&client\_secret=%2";  *const* QString client\_id = "VCVHPjKsbbcUuYRjBQ4ZGHK8";  *const* QString secret\_id = "lGK0ocvMR5nhhj6DMsB53vjMSFU6HrZ0";  *const* QString baiduImageUrl = "https://aip.baidubce.com/rest/2.0/image-classify/v1/animal?access\_token=";  *class* **Http** : *public* QObject  {  Q\_OBJECT  *public*:  **Http**();  *static* bool **post\_sync**(QString Url, QMap<QString, QString> header, QByteArray &requestData, QByteArray &replyData);  };  #endif *//* *HTTP\_H* |

代码块 4 Http头文件

|  |
| --- |
| #include "http.h"  Http::**Http**()  {  }  bool Http::**post\_sync**(QString Url, QMap<QString, QString> header, QByteArray &requestData, QByteArray &replyData)  {  QNetworkAccessManager manager; *//发送请求的动作*  QNetworkRequest request; *//请求内容（包含url和头）*  request.setUrl(Url);  QMapIterator<QString, QString> it(header);  *while* (it.hasNext())  {  it.next();  request.setRawHeader(it.key().toLatin1(),it.value().toLatin1());  }  *//发起请求*  QNetworkReply \*replay = manager.post(request, requestData);  QEventLoop l;  connect(replay,&QNetworkReply::finished, &l, &QEventLoop::quit);  l.exec();  *if*(replay != *nullptr* && replay->error() == QNetworkReply::*NoError*)  {  replyData = replay->readAll();  *return* *true*;  }  *else*  {  *return* *false*;  }  } |

代码块 5 Http实现代码

### 3.3.5打开文件函数

打开文件，选择文集路径，并保存，在label中显示。代码如下：

|  |
| --- |
| void MainWindow::**openImageSlot**()  {  imgPath = QFileDialog::getOpenFileName(*this*,"选择图片",QCoreApplication::applicationFilePath());  *if*(imgPath.isEmpty())  {  QMessageBox::warning(*this*,"提示","请选择一张图片");  }  QPixmap pix(imgPath);  ui->label->setPixmap(pix);  } |

代码块 6打开文件代码

### 3.3.6开始识别按钮

点击开始识别，将图片进行处理，拼接获取access\_token，如何发送请求获得结果。代码如下：

|  |
| --- |
| void MainWindow::**on\_pushButton\_clicked**()  {  QByteArray img = Image::imageToBase64(imgPath);  QByteArray imgData = "image=" + img;  *//获取access\_token*  QByteArray replyData;  QString url = QString(BaiduTokenUrl).arg(client\_id).arg(secret\_id);  QMap<QString, QString> header;  header.insert(QString("Content-Type"), QString("application/x-www-form-urlencoded"));  QString accessToken;  bool result;  result = Http::post\_sync(url, header, *imgData*, *replyData*);  *if*(result)  {  QJsonObject obj = QJsonDocument::fromJson(replyData).object();  qDebug() << obj.value("access\_token").toString();  accessToken = obj.value("access\_token").toString();  }  char\* ch;  QByteArray ba = accessToken.toLatin1();  ch=ba.data();  replyData.clear();  QString imgUrl = baiduImageUrl + accessToken;  qDebug() << imgUrl;  *//发起第二次请求*  result = Http::post\_sync(imgUrl,header,*imgData*,*replyData*);  *if*(result)  {  *//没办法解析？？？*  QJsonObject obj = QJsonDocument::fromJson(replyData).object();  qDebug() << obj.value("log\_id").toString();  QJsonValue val = obj.value("result");  *if*(val.isArray())  {  QJsonValue first = val.toArray().at(0);  *if*(first.isObject())  {  QString name = first.toObject().value("name").toString();  qDebug() << first.toObject().value("name").toString();  ui->lineEdit->setText(name);  *return*;  }  }  }  ui->lineEdit->setText("识别错误，这是什么呢？");  *return*;  } |

代码块 7开始识别代码

## 3.3类图关系

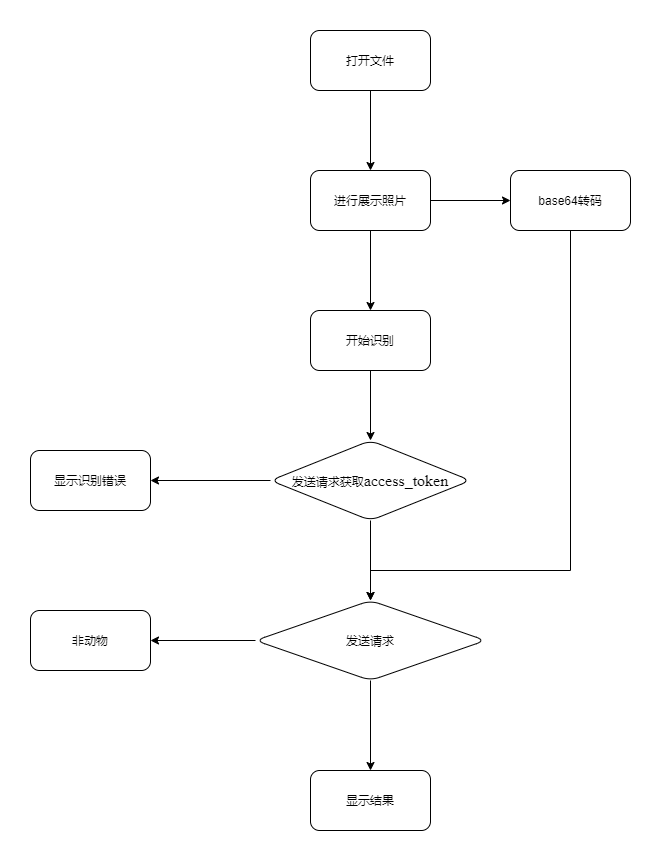


图 6类关系流程图

# 4.程序展示

一些操作实际展示，展示效果如下图：

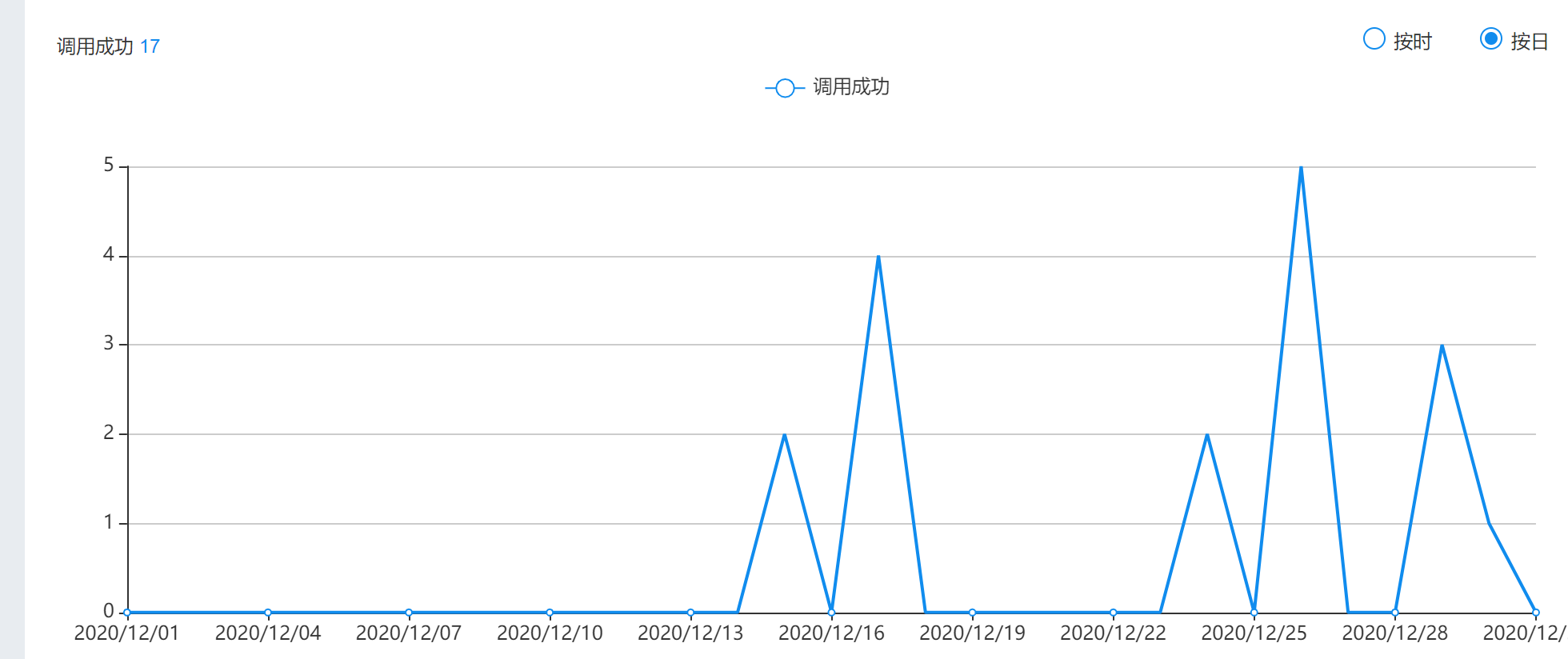
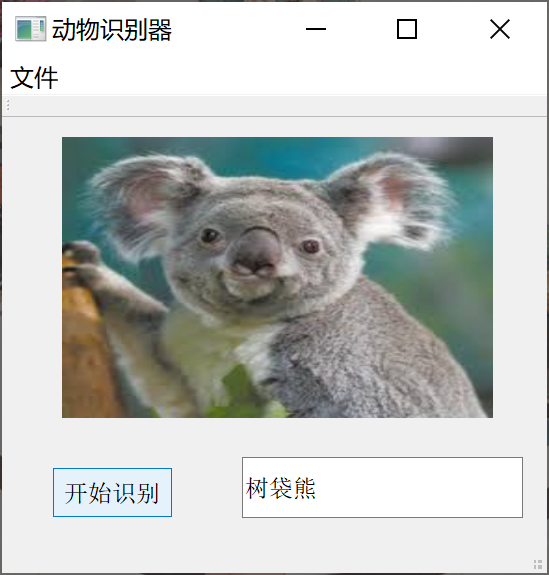
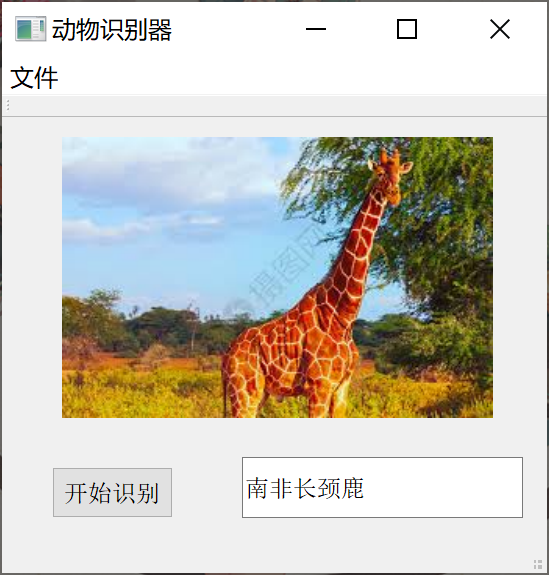


图 7 调用流量显示

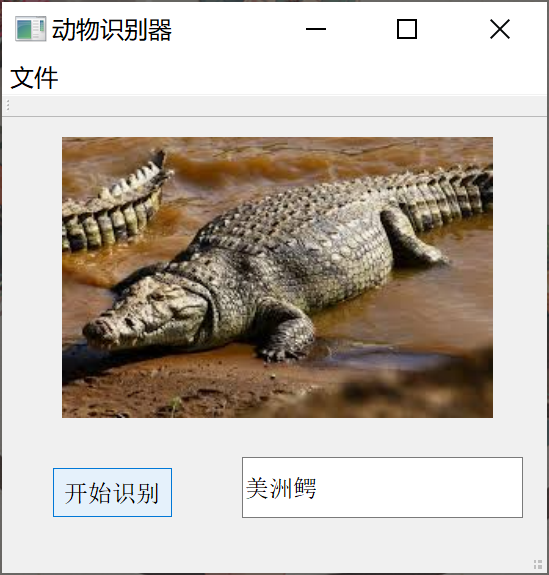
 

图 8成果展示图

# 5.总结思考

通过使用Qt应用框架实现了人机交互界面的计算器，采用Qt实现页面和发送请求，通过使用百度AI的API接口，实现了图像识别功能。该程序是一个集继承、图形界面、事件处理等面向对象编程知识的综合应用的实例程序。