**基于Lire的分析与扩展**

**扩展与展示实现方案**

Version 1.3

小组成员：

刘少凡

宋昱材

吴沂楠

黄飞

**版本变更记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 变更时间 | 修改人 | 审核人 | 备注 |
| 1.0 | 20170426 | 宋昱材 | 刘少凡 吴沂楠 黄飞 | 初稿 |
| 1.1 | 20170502 | 宋昱材 | 刘少凡 吴沂楠  黄飞 | 增加具体分工 |
| 1.2 | 20170505 | 宋昱材 | 刘少凡 吴沂楠 黄飞 | 更新第9周实现进度 |
| 1.3 | 20170510 | 吴沂楠 刘少凡  黄飞 宋昱材 |  | 增添类图、顺序图描述系统静态结构与人机交互过程，更新第10周进度 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1改进目标 4](#_Toc482213232)

[2工作内容 4](#_Toc482213233)

[2.1 Lire源码修改 4](#_Toc482213234)

[2.2 Caffe环境配置和模型调用 4](#_Toc482213235)

[2.4 JNI实现 5](#_Toc482213236)

[2.3界面实现 5](#_Toc482213237)

[3实现方案 5](#_Toc482213238)

[3.1 Lire源码修改 5](#_Toc482213239)

[3.2 Caffe环境配置和模型调用 6](#_Toc482213240)

[3.3 JNI实现 6](#_Toc482213241)

[3.3.1 Java端 6](#_Toc482213242)

[3.3.2 C++端 6](#_Toc482213243)

[3.4 界面实现 7](#_Toc482213244)

[3.5 系统类图 8](#_Toc482213245)

[3.6 人机交互过程 8](#_Toc482213246)

[3.6.1 图像入库过程 8](#_Toc482213247)

[3.6.2 图像检索过程 9](#_Toc482213248)

[4 具体分工 10](#_Toc482213249)

[4.1 第9周开发任务分配 10](#_Toc482213250)

[4.2 第10周开发任务分配 10](#_Toc482213251)

[5实现进度 11](#_Toc482213252)

[5.1 第9周实现进度 11](#_Toc482213253)

# 1改进目标

作为一种开源框架，Lire的代码对开发者完全透明，程序代码具备简明、方便和清晰的构架设计与函数接口来方便用户的使用。而且，Lire的实现框架使得开发者可以方便地引入新的技术、算法或模块，以满足开发人员的不同需求。因此Lire具备良好的可修改性或可扩展性。

项目的改进目标基于上述的可修改性。

项目计划在Lire中增加一种新的特征提取方法——CNN特征。CNN（Convolutional Neural Network），即卷积神经网络。项目计划利用一个已训练好的面向图像分类任务的CNN模型，将其作为特征提取工具，从模型中提取某一层输出作为图像特征。

这种尝试的出发点基于实际开发中时常会出现的场景，即CBIR系统开发者计划使用Lire工具包进行系统开发，但Lire工具包中并未实现开发者所希望使用的图像特征，因此需要向工具包中扩展该图像特征。项目站在CBIR系统开发者角度，对Lire针对特定开发目标进行扩展。

# 2工作内容

## 2.1 Lire源码修改

Lire的特征提取方法的具体实现在imageanalysis包中，通过LireFeature接口定义了特征提取类需要实现的方法，具体的特征提取类如CEDD等通过继承该接口进行具体的实现。因此需要通过继承LireFeature实现新的CNN类。

另外，要在图像入库和图像检索模块增加CNN特征类的接口。

## 2.2 Caffe环境配置和模型调用

深度学习框架Caffe的运行环境需要依赖多种工具和库的支持，因此需要对操作系统的环境进行配置。

编写C++代码，实现对模型的调用。C++代码主要实现两个功能，一是读取模型配置文件和参数文件将模型加载到内存中；二是将需要提取特征的图片输入模型，获得模型输出结果。

## 2.4 JNI实现

由于Lire基于Java实现，Caffe模型调用基于C++实现，因此需要利用java的JNI机制实现Lire对Caffe的调用。

## 2.3界面实现

前端界面主要包括图像主界面、图像入库界面、图像检索界面和检索结果界面。

# 3实现方案

## 3.1 Lire源码修改

imageamalysis包中的LireFeature接口定义了图像特征类需要实现的方法，具体内容如下所示：

|  |
| --- |
| **package** net.semanticmetadata.lire.imageanalysis;  **import** java.awt.image.BufferedImage;  **public** **interface** LireFeature **extends** Histogram {  //获取特征名称  **public** String getFeatureName();  //获取字段名  **public** String getFieldName();  //提取特征  **public** **void** extract(BufferedImage image);  //获取图像特征的Byte表示  **public** **byte**[] getByteArrayRepresentation();  //重置图像特征  **public** **void** setByteArrayRepresentation(**byte**[] featureData);  //有位移和长度的重置图像特征  **public** **void** setByteArrayRepresentation(**byte**[] featureData, **int** offset, **int** length);  //获取图像特征  **public** **double**[] getDoubleHistogram();  //获取该特征与输入特征的距离  **float** getDistance(LireFeature feature);  //获取特征的字符串表示  java.lang.String getStringRepresentation();  //以字符串重置图像特征  **void** setStringRepresentation(java.lang.String featureVector);} |

通过继承LireFeature接口，实现CNN类。

另外，在Lire工具包的DocumentBuilder.java、DocumentBuilderFactory.java、ImageSearcherFactory.java和ImageSearcher.java中增加相应的构造方法实现和字段添加，实现对索引构造、特征存储和特征搜索代码的调用。

## 3.2 Caffe环境配置和模型调用

项目计划在Ubuntu 14.04操作系统上配置Caffe，依据Caffe文档依次安装依赖库，实现Caffe的安装。

通过C++代码调用Caffe的相关接口实现加载模型和调用模型的相关功能。

## 3.3 JNI实现

### 3.3.1 Java端

Java端实现GetCNN类，通过调用native的C++方法实现对Caffe模型的调用。GetCNN类应当包含以下成员变量和方法：

**表 3. 1 GetCNN类成员变量与方法作用**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 变量/方法 | 类型 | 作用 |
| NetTxt | 成员变量 | String | 记录CNNs模型定义文件路径 |
| NetPara | 成员变量 | String | 记录CNNs模型参数文件路径 |
| LayerName | 成员变量 | String | 记录CNNs模型层名 |
| loadCNN | 私有方法 | void | JNI调用本地Caffe代码实现CNNs模型加载 |
| getCNN | 私有方法 | float[] | JNI调用本地Caffe代码实现特征提取 |
| getFeature | 公共方法 | flaot[] | 共外部调用的图像特征提取接口 |

其中loadCNN和getCNN为native方法，通过调用本地C++代码实现。通过javah命令生成C++所需的.h头文，供C++代码实现时包含。

### 3.3.2 C++端

C++代码包含java生成的头文件，实现头文件中定义的具体方法，将代码生成动态链接库，供java代码调用。

## 3.4 界面实现

前端界面主要包括程序主界面、入库界面、图像检索界面和检索结果界面。

用户操作程序界面的流程如图3.1所示。



图 3.1 用户操作界面流程图

界面相关类信息如表3.2所示。

**表 3.2 界面相关类信息**

|  |  |  |
| --- | --- | --- |
| 类 | 界面 | 功能描述 |
| StartFrame | 主界面 | 提供入库与检索不同功能的选择 |
| StorageFrame | 入库界面 | 选择入库图片路径，进行图片入库 |
| MainFeame | 检索界面 | 选择检索图片，进行图像检索 |
| ImageResultFrame | 检索结果界面 | 检索结果显示 |

## 系统类图

* + 1. **特征提取模块**

主要由CNN.java和GetCNN.java实现，两个类的类图如图3.2所示。

图 3.2 特征提取模块类图

GetCNN类实现对VGG-F模型的调用，提取原始VGG-F特征，其成员变量意义与方法作用如表3.1所示。

* + 1. **图像入库与图像检索实现**

图像入库与图像检索相关类图如图3.3所示：



图 3.3 图像入库与图像检索相关类图

图像入库时通过DocumentBuilderFactory类生成此次入库使用的DocunmentBuilder，DocumentBuilder调用CNN进行特征提取，完成特征存储和索引生成。

图像检索时通过ImageSearcherFactory类生成此次检索的ImageSearcher，ImageSearcher通过调用CNN进行特征提取、特征距离计算，完成检索。

最终，完成了对Lire的扩展工作，将VGG-F特征加入到Lire中，并实现了图像入库模块和图像检索模块。

## 3.6 人机交互过程

系统的功能主要为图像入库与图像检索，下面从这两个方面来描述人机交互过程。

### 3.6.1 图像入库过程



图 3.4 图像入库过程顺序图

图像入库过程的用户输入是一个文件夹路径，该文件夹中包含此次入库的所有图像，最终系统将图像入库之后通知用户入库完成。

过程如图3.4，首先用户打开系统主界面，然后选择图像入库，主界面会再启动一个入库主界面，然后用户在入库主界面选择图片所在文件夹的路径，并点击开始按钮，系统进行图像入库，建立图片索引，索引构建完成之后，系统告知用户入库完成。

### 3.6.2 图像检索过程

图 3.5 图像检索过程顺序图

图像检索过程的用户输入是一张图片，最终系统将检索库中相似图片，按照相似度返回检索结果。

过程如图3.5，首先用户打开系统主界面，然后选择图像检索，主界面会再启动一个检索主界面，然后用户在检索主界面上传图片，并点击搜索按钮，系统进行图像检索，并启动一个检索结果页面用以展示检索结果。

# 4 具体分工

## 4.1 第9周开发任务分配

（1）Caffe环境配置

负责人：宋昱材

工作内容：在Ubuntu 14.04操作系统上完成对Caffe框架的搭建。

（2）Lire代码修改

负责人：刘少凡、黄飞

工作内容：完成继承LireFeature接口的CNN类的具体方法编写。

（3）界面编写

负责人：吴沂楠

工作内容：完成主界面、入库、检索界面的编写

## 4.2 第10周开发任务分配

（1）CNN模型调用代码编写

负责人：宋昱材

工作内容：完成调用CNN模型的C++代码编写，实现JNI机制。与其他组员配合实现系统整合和调试。

（2）Lire代码修改与系统整合

负责人：刘少凡、黄飞

工作内容：完成GetCNN类的相关代码编写，完成入库和检索模块相关代码的修改。与其他组员配合实现系统整合和调试。

（2）界面编写

负责人：吴沂楠

工作内容：完成检索结果展示界面的编写。与其他组员配合实现系统整合和调试。

# 5实现进度

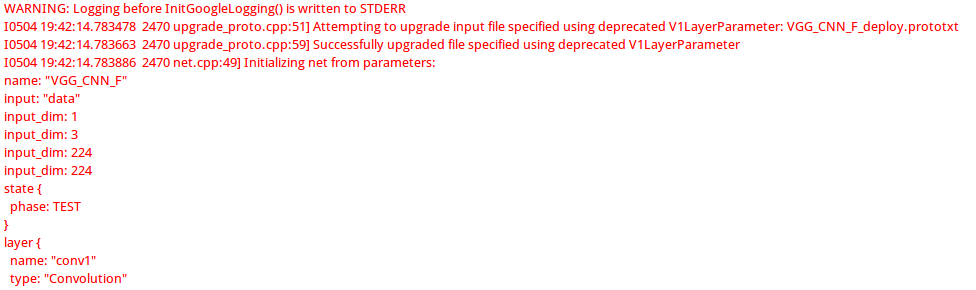
## 5.1 第9周实现进度

（1）Caffe环境配置

负责人：宋昱材

* 实现内容
* 完成Caffe框架的搭建

在Ubuntu 14.04上进行了Caffe框架的搭建，并进行了调用模型的测试，如图5.1所示，为Caffe加载VGG-F模型后的日志记录。



* 确定CNN模型

另一个深度学习框架MatConvNet总结了近几年的多种CNNs模型在相同硬件条件下处理ILSVRC 2012测试数据集的准确率与时间效率，其中部分数据如表5.1所示。

表5.1中的alexnet模型是最早提出的CNNs模型。vgg-f模型沿用了alexnet模型的8层基础架构，并通过调整卷积核的尺寸和数量，改变参数数量，使得模型在保持并一定程度上提升分类准确率的情况下，很大程度提高了处理图片的时间效率。

模型在实际运行过程中处理图片的时间复杂度直接影响实际应用系统运行时特征提取的时间效率，因此模型的时间复杂度不能过高。

基于以上原因，确定了基于8层基本CNNs结构的、时间复杂度较低的VGG-F模型作为项目所选择的模型。

**表 5.0‑1 各CNNs模型准确率与时间效率对比**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型名称 | 提出时间 | 模型层数 | Top-1错误率 | Top-5错误率 | 每秒处理图片量 |
| resnet-50[[26](#_ENREF_26)] | 2015 | 50 | 24.6 | 7.7 | 315.3 |
| resnet-101[[26](#_ENREF_26)] | 2015 | 101 | 23.4 | 7.0 | 212.7 |
| resnet-152[[26](#_ENREF_26)] | 2015 | 152 | 23.0 | 6.7 | 156.6 |
| vgg-verydeep-19[[14](#_ENREF_14)] | 2014 | 19 | 28.7 | 9.9 | 154.5 |
| vgg-verydeep-16[[14](#_ENREF_14)] | 2014 | 16 | 28.5 | 9.9 | 183.1 |
| googlenet[[27](#_ENREF_27)] | 2014 | 22 | 34.2 | 12.9 | 501.8 |
| vgg-f[[28](#_ENREF_28)] | 2013 | 8 | 41.1 | 18.8 | 793.1 |
| alexnet[[13](#_ENREF_13)] | 2012 | 8 | 42.6 | 19.6 | 382.4 |

注：表中数据来源为<http://www.vlfeat.org/matconvnet/pretrained/>。

* 下周工作

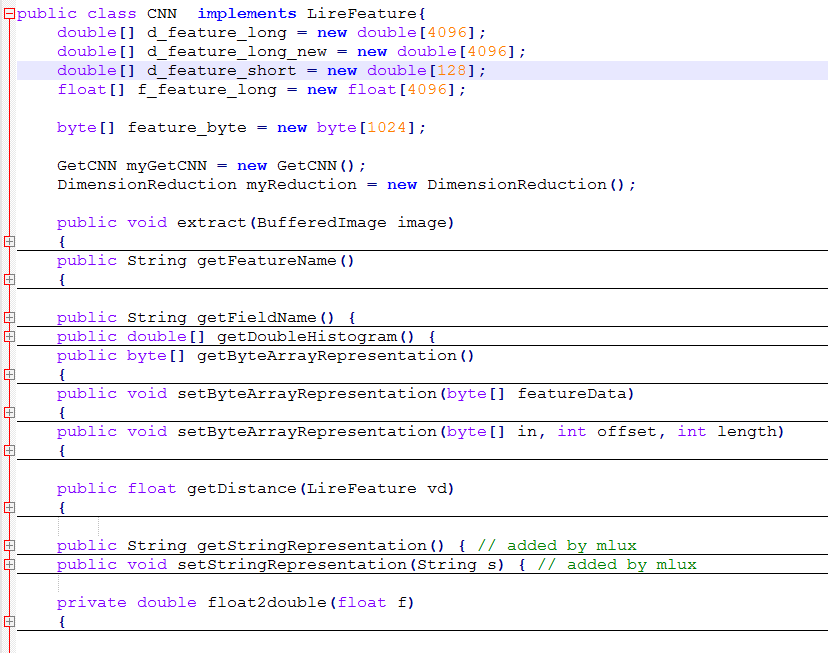
完成调用CNN模型的C++代码编写，实现JNI机制。与其他组员配合实现系统整合和调试。

（2）Lire代码修改

负责人：刘少凡、黄飞

* 实现内容

通过继承LireFeature接口，实现了CNN类，主要工作为实现如图所示的相关方法。



* 下周工作

完成GetCNN类的相关代码编写，完成入库和检索模块相关代码的修改。与其他组员配合实现系统整合和调试。

（3）界面编写

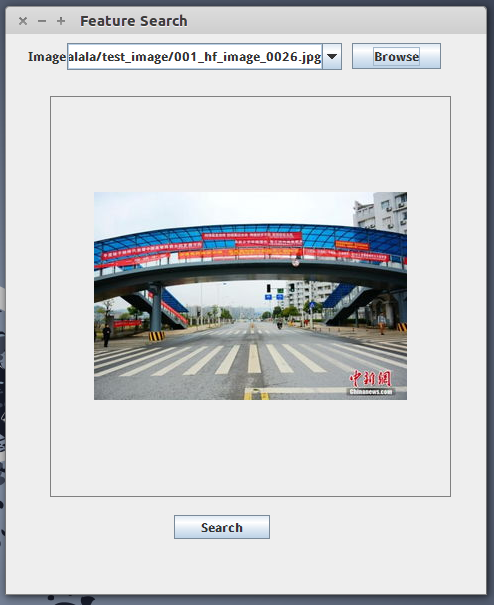
负责人：吴沂楠

* 实现内容
* 主界面



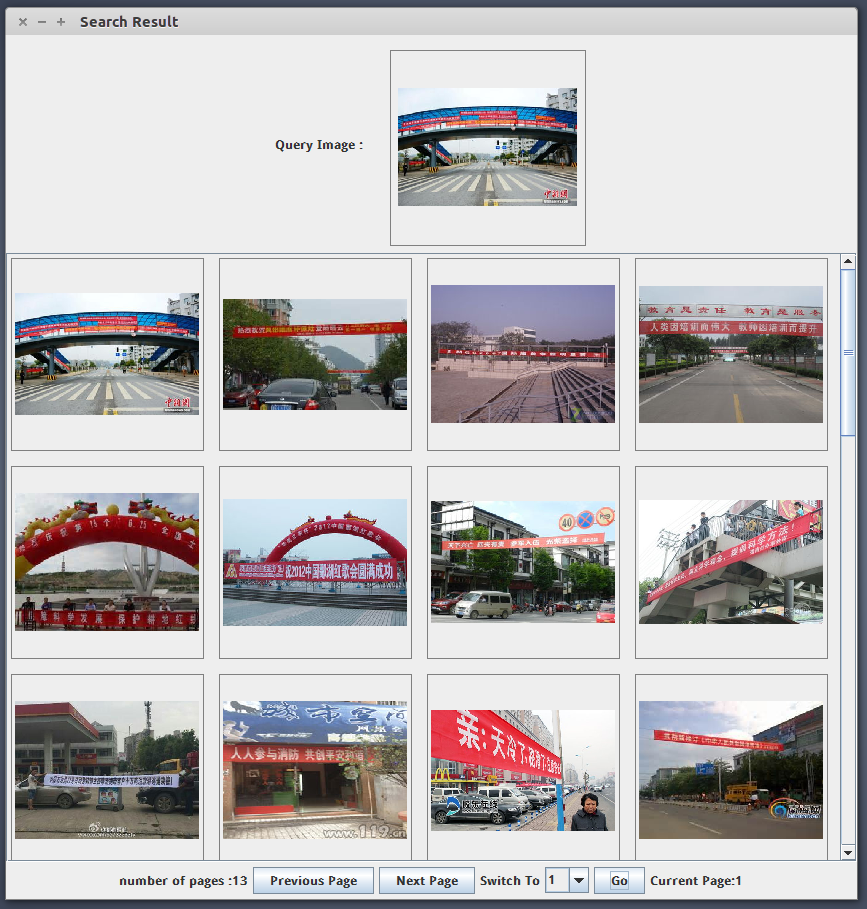
通过点击检索和入库按钮，分别进入检索界面和入库界面

* 检索界面



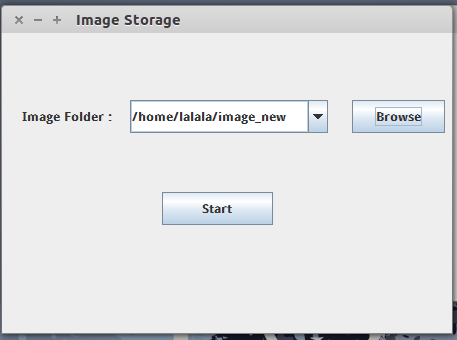
选择检索示例图片，点击检索按钮，检索结果由结果展示结果显示。

* 检索结果展示界面



界面上方显示用户输入的实例图片，下方显示检索结果。

* 入库界面



用户输入入库图片文件夹的路径，点击开始，进行入库。

* 下周工作

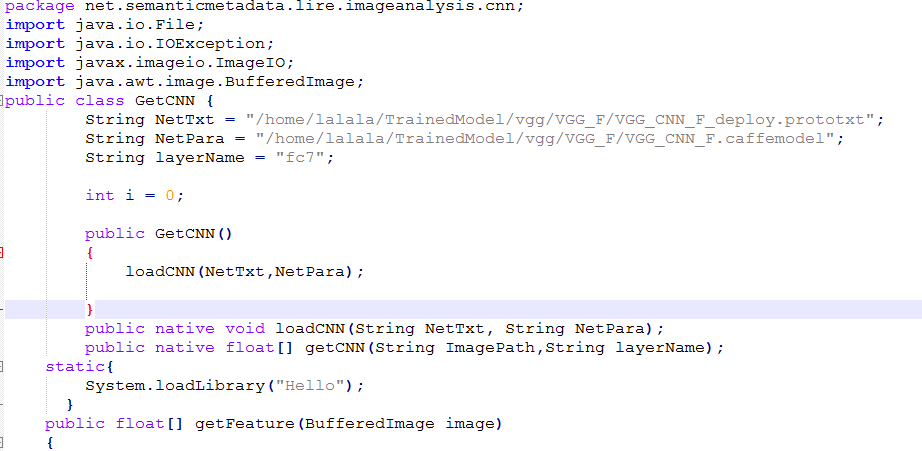
协助宋昱材完成调用CNN模型的C++代码编写，实现JNI机制。与其他组员配合实现系统整合和调试。

## 5.2 第10周实现进度

（1）GetCNN类实现

负责人：刘少凡、黄飞

实现GetCNN类，GetCNN类主要实现两个native方法，loadCNN和getCNN方法。相关代码如图所示。



loadCNN方法输入模型定义文件路径和模型参数文件路径，通过C++代码，将CNN模型加载到内存中。

getCNN方法输入图片路径和层数，通过C++代码，调用内存中的CNN模型提取图像特征。

getFeature方法为提供给CNN类的外部接口，CNN类通过getFeature方法调用getCNN方法提取特征。

调用javah命令，生成GetCNN类对应的C++头文件。

修改图像入库和图像检索的相关代码，增加CNN特征相关的接口。

（2）C++代码实现

负责人：宋昱材、吴沂楠

通过包含GetCNN类生成的头文件，实现C++的函数，主要为两个函数，即GetCNN类中loadCNN方法和getCNN方法对应的C++函数。

Java\_net\_semanticmetadata\_lire\_imageanalysis\_cnn\_GetCNN\_loadCNN方法通过调用Caffe的模型操作相关接口实现对模型配置文件和模型参数文件的读取和加载。

Java\_net\_semanticmetadata\_lire\_imageanalysis\_cnn\_GetCNN\_getCNN方法通过调用Caffe的模型测试相关接口，输入图片进行模型运算，从内存中读取运算结果，得到图像特征。



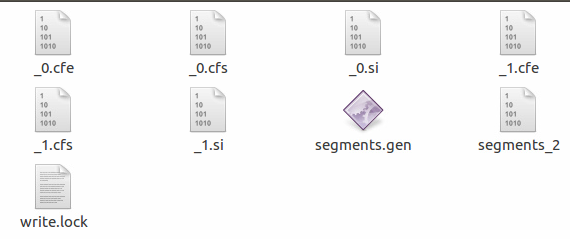
C++代码完成后，将C++代码编译生成动态链接库，供java调用。

（3）系统整合

负责人：刘少凡、宋昱材、吴沂楠、黄飞

将Lire代码与C++代码生成的动态链接库进行整合，实现系统的JNI机制。并对系统进行简单的测试，确保运行成功。

入库成功后，设置的保存索引的文件夹目录下会出现新的索引文件，如图，其中\_0开头的是第一次入库生成的索引，\_1开头的是第二次入库生成的索引。



检索完成后，会依序显示检索结果，如图。

