# 语义分割网络贯通设计说明书

## 1 项目开发目的和意义

语义分割是计算机视觉中的基本任务，在语义分割中我们需要将视觉输入分为不同的语义可解释类别，「语义的可解释性」即分类类别在真实世界中是有意义的，与图像分类或目标检测相比，语义分割使我们对图像有更加细致的了解。这种了解在诸如自动驾驶、机器人以及图像搜索引擎等许多领域都是非常重要的。

目前有一些常用于训练语义分割模型的数据集：

Pascal VOC 2012：有 20 类目标，这些目标包括人类、机动车类以及其他类，可用于目标类别或背景的分割

Cityscapes：50 个城市的城市场景语义理解数据集；

Pascal Context：有 400 多类的室内和室外场景

Stanford Background Dataset：至少有一个前景物体的一组户外场景。

## 2 总体设计

语义分割网络系统可以划分为数据预处理、模型推理、数据后处理三个主要子系统，各子系统相对独立，但存在数据关联。其中数据预处理包括读取图片，转换为模型需要的格式和大小；模型推理，使用AIModeManager接口，对输入的图片数据进行推理；数据后处理完成推理结果的展示。语义分割网络的代码参见 https://gitee.com/Atlas200DK/sample-segmentation

为了说明各模块之间的结构关系，细化的整体结构图如2-1所示。

开始

数据预处理

模型推理

数据后处理

结束

图2-1 系统整体功能结构图

## 3、主要工作

### 将语义分割模型转换为om模型，在分类网络样例基础上修改后处理代码，实现图片语义分割的显示。

### 3.1 模型转om

模型转换可参考 ：

<https://gitee.com/Atlas200DK/sample-segmentation/blob/master/Readme.md>

语义分割模型下载路径：

https://github.com/Atlas200dk/models/tree/master/computer\_vision/segmentation/

### 3.2 语义分割网络处理流程

#### 3.2.1图像预处理

分类网络的数据预处理模块，采用Opencv的imread函数读图片，将图片数据发送给推理模块。

#### **3.2.2 模型转换及推理**

推理模块调用DVPP接口，将BGR图片转为YUV格式。将YUV数据送给推理引擎，推理引擎首先调用ai\_model\_manager\_的CreateOutputTensor接口创建输出Tensor， 然后调用ai\_model\_manager\_的Process接口进行推理。

#### **3.2.3 图片后处理**

后处理模块读取推理的结果，输出每个像素点的类别概率。每个像素点选定概率最大的类别作为它的类别。给不同类别设置不同像素颜色，输出图片。

读取推理结果

每个像素点归属类别分析

设置类别的颜色

不同类别输出不同颜色

结束

### 3.3 模型说明

语义分割模型下载路径：<https://gitee.com/HuaweiAscend/models/tree/master/computer_vision/segmentation>

语义分割模型是基于分类网络，一般都用卷积神经网络为每个像素分配一个初始类别标签。模型的输出为每个像素点的置信度。

例如Erfnet 模型，输入为1024\*512，输出为19\*1024\*512， ，表示每个像素点在19个类别的置信度。

## 4 关键代码

语义分割网络贯通软件实现流程和分类网络类似，后处理引擎需要根据语义分割模型做处理。

#### **4.1函数定义**

HIAI\_StatusT GeneralPost::SegmentationNetPostProcess(

const shared\_ptr<EngineTrans> &result) {

#### **4.2函数实现**

string file\_path = result->image\_info.path;

/\*if inference result is null return \*/

if (result->inference\_res.empty()){

ERROR\_LOG("Failed to deal file=%s. Reason: inference result empty.", file\_path.c\_str());

return HIAI\_ERROR;

}

/\* judge categories num \*/

if(MAX\_NUM < result->console\_params.output\_nums)

{

ERROR\_LOG("Failed to deal categories num %d is large than 255.", result->console\_params.output\_nums);

return HIAI\_ERROR;

}

/\* set lut file \*/

uchar lutData[MAX\_NUM \* 3];

uchar label;

for (int i = 0; i<MAX\_NUM; i++)

{

int j =0;

label = i;

while(label >0)

{

lutData[i \* 3] |= (((label >> 0) & 1) << (7 - j));

lutData[i \* 3 + 1] |= (((label >> 1) & 1) << (7 - j));

lutData[i \* 3 + 2] |= (((label >> 2) & 1) << (7 - j));

label >>=3;

j += 1;

}

}

cv::Mat label\_colours(1, MAX\_NUM, CV\_8UC3, lutData);

cv::cvtColor(label\_colours, label\_colours, CV\_RGB2BGR);

/\*read inference result \*/

Output out = result->inference\_res[0];

int32\_t size = out.size / sizeof(float);

int32\_t ModelOutSize = result->image\_info.width\* result->image\_info.height\*result->console\_params.output\_nums;

if ((size <= 0)||(size < ModelOutSize)) {

ERROR\_LOG("Failed to deal file=%s. Reason: inference result size=%d error.", file\_path.c\_str(), size);

return HIAI\_ERROR;

}

/\*copy inference data \*/

cv::Mat class\_each\_row (result->console\_params.output\_nums, result->image\_info.width\* result->image\_info.height, CV\_32FC1, const\_cast<float \*>((float \*)out.data.get()));

/\* transpose to make each row with all probabilities \*/

class\_each\_row = class\_each\_row.t();

cv::Point maxId; // point [x,y] values for index of max

double maxValue; // the holy max value itself

cv::Mat prediction\_map(result->image\_info.height,result->image\_info.width, CV\_8UC1);

/\* select each pixle max \*/

for (int i = 0; i<class\_each\_row.rows;i++){

minMaxLoc(class\_each\_row.row(i),0,&maxValue,0,&maxId);

prediction\_map.at<uchar>(i) = maxId.x;

}

/\* get inference out pic\*/

cv::cvtColor(prediction\_map.clone(), prediction\_map, CV\_GRAY2BGR);

cv::Mat Infer\_output\_image ;

cv::LUT(prediction\_map, label\_colours, Infer\_output\_image);

/\* get origin pic \*/

cv::Mat originImage= cv::imread(result->image\_info.path, CV\_LOAD\_IMAGE\_UNCHANGED);

resize(originImage,originImage,cv::Size( result->image\_info.width,result->image\_info.height),

originImage.cols/result->image\_info.width,originImage.rows/result->image\_info.height);

/\* get display image\*/

cv::Mat displayImage ;

displayImage = 0.5\*Infer\_output\_image +0.5\*originImage;

/\*set output img name \*/

stringstream sstream;

int pos = result->image\_info.path.find\_last\_of(kFileSperator);

string file\_name(result->image\_info.path.substr(pos + 1));

bool save\_ret(true);

sstream.str("");

sstream << result->console\_params.output\_path << kFileSperator << kOutputFilePrefix << file\_name;

string output\_path = sstream.str();

/\*write img \*/

save\_ret = cv::imwrite(output\_path, displayImage);

if (!save\_ret) {

ERROR\_LOG("Failed to deal file=%s. Reason: save image failed.",

result->image\_info.path.c\_str());

return HIAI\_ERROR;

}

// INFO\_LOG("output\_image channels imwrite end \n");

return HIAI\_OK;

}

## 5 推理结果：

## FCN模型（使用Pascal数据集训练）推理：

## 000054out_000054

Erfnet（使用cityscape数据集训练）模型推理：





## 6 后续可扩展性

后续可用数据集训练模型，进一步提高模型的精度。

Pascal数据集：https://pjreddie.com/projects/pascal-voc-dataset-mirror/

Cityscape数据集：https://pan.baidu.com/s/1w3W\_dQBUiHcwkLOtbSJ1Tg 提取码：1bln