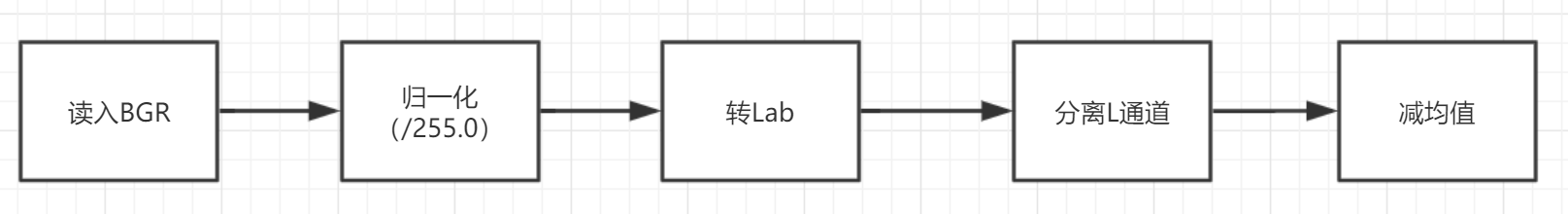
黑白图像上色开发指导

1. 环境准备

本文档基于现有社区样例：通用分类网络应用（项目地址：<https://gitee.com/Atlas200DK/sample-classification>）进行二次开发。本文档基于以下软硬件环境：

1. Ubuntu宿主机：安装了Ubuntu 16.04.3操作系统，并且已配置交叉编环境。
2. 开发板：制C30B891的卡（制卡请参考：<https://ascend.huawei.com/doc/Atlas200DK/1.3.0.0/zh/zh-cn_topic_0195268776.html>）。
3. 安装了C30B891版本的MindStudio和DDK
4. 已获取训练好的黑白上色caffe模型（<https://github.com/richzhang/colorization>）
5. 设计思路

程序的设计围绕着功能和模型展开。该黑白上色模型实现对图片的自动上色，程序的输入输出均为图片。该模型输入图像减均值后的L通道数据，输出预测的ab通道数据。故模型推理前，程序需要对图片进行预处理，这里采用opencv读取图片，读取出来的为BGR格式，需要把BGR格式转为Lab格式，并分离出L通道数据，作为模型推理的输入数据。



后处理部分，获取模型的输出（预测的ab通道数据），resize至原图片大小，并与原图片L通道数据合并，获取完整Lab数据，再转换为BGR保存成图片文件。



因此，本应用程序的设计思路为读取图片、图片预处理、推理、后处理保存图片。在已有社区样例中，通用分类网络应用（sample-classification）的实现为：读取输出图片，调用ezdvpp进行图像resize，对resize后的图像进行推理，获取分类结果并排序打印，大体符合本应用实现流程，故基于通用分类网络应用进行二次开发。

1. 代码实现

3.1 图像预处理

图像预处理部分的代码入口如下：

|  |
| --- |
| HIAI\_IMPL\_ENGINE\_PROCESS**(**"general\_image"**,**  GeneralImage**,** INPUT\_SIZE**)** **{**  HIAI\_StatusT ret **=** HIAI\_OK**;**  // Step1: check arg0  **if** **(**arg0 **==** **nullptr)** **{**  ERROR\_LOG**(**"Failed to deal file=nothing. Reason: arg0 is empty."**);**  **return** HIAI\_ERROR**;**  **}**  // Step2: get all files  shared\_ptr**<**ConsoleParams**>** console\_param **=** static\_pointer\_cast**<**ConsoleParams**>(**  arg0**);**  string input\_path **=** string**(**console\_param**->**input\_path**);**  vector**<**string**>** file\_vec**;**  GetAllFiles**(**input\_path**,** file\_vec**);**  **if** **(**file\_vec**.**empty**())** **{**  ERROR\_LOG**(**"Failed to deal all empty path=%s."**,** input\_path**.**c\_str**());**  **return** HIAI\_ERROR**;**  **}**  // Step3: send every image to inference engine  **for** **(**string path **:** file\_vec**)** **{**  shared\_ptr**<**EngineTrans**>** image\_handle **=** **nullptr;**  MAKE\_SHARED\_NO\_THROW**(**image\_handle**,** EngineTrans**);**  **if** **(**image\_handle **==** **nullptr)** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: new EngineTrans failed."**,**  path**.**c\_str**());**  **continue;**  **}**  // arrange image information, if failed, skip this image  **if** **(!**ArrangeImageInfo**(**image\_handle**,** path**))** **{**  **continue;**  **}**  // send data to inference engine  image\_handle**->**console\_params**.**input\_path **=** console\_param**->**input\_path**;**  image\_handle**->**console\_params**.**model\_height **=** console\_param**->**model\_height**;**  image\_handle**->**console\_params**.**model\_width **=** console\_param**->**model\_width**;**  image\_handle**->**console\_params**.**output\_path **=** console\_param**->**output\_path**;**  **if** **(!**SendToEngine**(**image\_handle**))** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: send data failed."**,**  path**.**c\_str**());**  **continue;**  **}**  // sleep  usleep**(**kSleepInterval**);**  **}**  // Step4: send finished data  shared\_ptr**<**EngineTrans**>** image\_handle **=** **nullptr;**  MAKE\_SHARED\_NO\_THROW**(**image\_handle**,** EngineTrans**);**  **if** **(**image\_handle **==** **nullptr)** **{**  ERROR\_LOG**(**"Failed to send finish data. Reason: new EngineTrans failed."**);**  ERROR\_LOG**(**"Please stop this process manually."**);**  **return** HIAI\_ERROR**;**  **}**  image\_handle**->**is\_finished **=** **true;**  **if** **(**SendToEngine**(**image\_handle**))** **{**  **return** HIAI\_OK**;**  **}**  ERROR\_LOG**(**"Failed to send finish data. Reason: SendData failed."**);**  ERROR\_LOG**(**"Please stop this process manually."**);**  **return** HIAI\_ERROR**;**  **}** |

预处理部分resize、色域转换、提取L通道代码如下：

|  |
| --- |
| bool GeneralImage**::**ArrangeImageInfo**(**shared\_ptr**<**EngineTrans**>** **&**image\_handle**,**  const string **&**image\_path**)** **{**  // read image using OPENCV  cv**::**Mat mat **=** cv**::**imread**(**image\_path**,** CV\_LOAD\_IMAGE\_COLOR**);**  //resize  cv**::**Mat mat\_rs**;**  cv**::**resize**(**mat**,** mat\_rs**,** cv**::**Size**(**224**,**224**));**    // deal image  mat\_rs**.**convertTo**(**mat\_rs**,**CV\_32FC3**);**  mat\_rs **=** 1.0**\***mat\_rs**/**255**;**  cv**::**cvtColor**(**mat\_rs**,** mat\_rs**,** CV\_BGR2Lab**);**  // pull out L channel and subtract 50 for mean-centering  std**::**vector**<**cv**::**Mat**>** channels**;**  cv**::**split**(**mat\_rs**,** channels**);**  cv**::**Mat mat\_rs\_l **=** channels**[**0**]** **-** 50**;**  **if** **(**mat**.**empty**())** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: read image failed."**,**  image\_path**.**c\_str**());**  **return** **false;**  **}**  // set property  image\_handle**->**image\_info**.**path **=** image\_path**;**  image\_handle**->**image\_info**.**width **=** mat\_rs\_l**.**cols**;**  image\_handle**->**image\_info**.**height **=** mat\_rs\_l**.**rows**;**  // set image data  uint32\_t size **=** mat\_rs\_l**.**total**()** **\*** mat\_rs\_l**.**channels**()\***4**;**  u\_int8\_t **\***image\_buf\_ptr **=** **new** **(**nothrow**)** u\_int8\_t**[**size**];**  **if** **(**image\_buf\_ptr **==** **nullptr)** **{**  HIAI\_ENGINE\_LOG**(**"new image buffer failed, size=%d!"**,** size**);**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: new image buffer failed."**,**  image\_path**.**c\_str**());**  **return** **false;**  **}**  error\_t mem\_ret **=** memcpy\_s**(**image\_buf\_ptr**,** size**,** mat\_rs\_l**.**ptr**<**u\_int8\_t**>(),**  mat\_rs\_l**.**total**()** **\*** mat\_rs\_l**.**channels**()\***4**);**  **if** **(**mem\_ret **!=** EOK**)** **{**  **delete[]** image\_buf\_ptr**;**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: memcpy\_s failed."**,**  image\_path**.**c\_str**());**  image\_buf\_ptr **=** **nullptr;**  **return** **false;**  **}**  image\_handle**->**image\_info**.**size **=** size**;**  image\_handle**->**image\_info**.**data**.**reset**(**image\_buf\_ptr**,**  default\_delete**<**u\_int8\_t**[]>());**  **return** **true;**  **}** |

其他部分代码，如遍历路径下所有图片等请参考工程目录下的general\_image.cpp文件

3.2 推理engine

原通用分类网络应用中推理engine中调用ezdvpp进行图像resize，但是ezdvpp输出的图像只有yuv格式，而本模型需要输入Lab格式，因此不用ezdvpp进行resize。图像预处理部分已经用opencv进行过resize，所以这一部分去掉resize的操作。修改后的推理engine入口代码如下：

|  |
| --- |
| HIAI\_IMPL\_ENGINE\_PROCESS**(**"general\_inference"**,**  GeneralInference**,** INPUT\_SIZE**)** **{**  HIAI\_StatusT ret **=** HIAI\_OK**;**  // arg0 is empty  **if** **(**arg0 **==** **nullptr)** **{**  HIAI\_ENGINE\_LOG**(**HIAI\_ENGINE\_RUN\_ARGS\_NOT\_RIGHT**,** "arg0 is empty."**);**  **return** HIAI\_ERROR**;**  **}**  // just send data when finished  shared\_ptr**<**EngineTrans**>** image\_handle **=** static\_pointer\_cast**<**EngineTrans**>(**arg0**);**  **if** **(**image\_handle**->**is\_finished**)** **{**  **if** **(**SendToEngine**(**image\_handle**))** **{**  **return** HIAI\_OK**;**  **}**  SendError**(**"Failed to send finish data. Reason: Inference SendData failed."**,**  image\_handle**);**  **return** HIAI\_ERROR**;**  **}**  // inference  vector**<**shared\_ptr**<**hiai**::**IAITensor**>>** output\_data**;**  **if** **(!**Inference**(**image\_handle**,** output\_data**))** **{**  string err\_msg **=** "Failed to deal file=" **+** image\_handle**->**image\_info**.**path  **+** ". Reason: inference failed."**;**  SendError**(**err\_msg**,** image\_handle**);**  **return** HIAI\_ERROR**;**  **}**  // send result  **if** **(!**SendResult**(**image\_handle**,** output\_data**))** **{**  string err\_msg **=** "Failed to deal file=" **+** image\_handle**->**image\_info**.**path  **+** ". Reason: Inference SendData failed."**;**  SendError**(**err\_msg**,** image\_handle**);**  **return** HIAI\_ERROR**;**  **}**  **return** HIAI\_OK**;**  **}** |

完整代码请参考general\_inference.cpp文件

3.3 后处理engine

后处理部分入口代码如下：

|  |
| --- |
| HIAI\_IMPL\_ENGINE\_PROCESS**(**"general\_post"**,** GeneralPost**,** INPUT\_SIZE**)** **{**  HIAI\_StatusT ret **=** HIAI\_OK**;**  // check arg0  **if** **(**arg0 **==** **nullptr)** **{**  ERROR\_LOG**(**"Failed to deal file=nothing. Reason: arg0 is empty."**);**  **return** HIAI\_ERROR**;**  **}**  // just send to callback function when finished  shared\_ptr**<**EngineTrans**>** result **=** static\_pointer\_cast**<**EngineTrans**>(**arg0**);**  **if** **(**result**->**is\_finished**)** **{**  **if** **(**SendSentinel**())** **{**  **return** HIAI\_OK**;**  **}**  ERROR\_LOG**(**"Failed to send finish data. Reason: SendData failed."**);**  ERROR\_LOG**(**"Please stop this process manually."**);**  **return** HIAI\_ERROR**;**  **}**  // inference failed  **if** **(**result**->**err\_msg**.**error**)** **{**  ERROR\_LOG**(**"%s"**,** result**->**err\_msg**.**err\_msg**.**c\_str**());**  **return** HIAI\_ERROR**;**  **}**  // arrange result  **return** ColorizationPostProcess**(**result**);**  **}** |

后处理部分获取预测的ab通道数据、resize回原图像尺寸、与L通道合并代码如下：

|  |
| --- |
| HIAI\_StatusT GeneralPost**::**ColorizationPostProcess**(**  const std**::**shared\_ptr**<**EngineTrans**>** **&**result**)** **{**  string file\_path **=** result**->**image\_info**.**path**;**  cout **<<** "file\_path = " **<<**file\_path **<<** endl**;**  // check vector  **if** **(**result**->**inference\_res**.**empty**())** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: inference result empty."**,**  file\_path**.**c\_str**());**  **return** HIAI\_ERROR**;**  **}**  // only need to get first one  Output out **=** result**->**inference\_res**[**0**];**  int32\_t size **=** out**.**size **/** **sizeof(**float**);**  // get a channel and b channel result data  cv**::**Mat mat\_a**(**56**,** 56**,** CV\_32FC1**,** **const\_cast<**float **\*>((**float **\*)**out**.**data**.**get**()));**  cv**::**Mat mat\_b**(**56**,** 56**,** CV\_32FC1**,** **const\_cast<**float **\*>((**float **\*)**out**.**data**.**get**()+**size**/**2**));**  // pull out L channel in original image  cv**::**Mat mat **=** cv**::**imread**(**file\_path**,** CV\_LOAD\_IMAGE\_COLOR**);**  mat**.**convertTo**(**mat**,**CV\_32FC3**);**  mat **=** 1.0**\***mat**/**255**;**  cv**::**cvtColor**(**mat**,** mat**,** CV\_BGR2Lab**);**  std**::**vector**<**cv**::**Mat**>** channels**;**  cv**::**split**(**mat**,** channels**);**  // resize to match size of original image L  int r **=** mat**.**rows**;**  int c **=** mat**.**cols**;**  cv**::**Mat mat\_a\_up**(**r**,**c**,**CV\_32FC1**);**  cv**::**Mat mat\_b\_up**(**r**,**c**,**CV\_32FC1**);**  cv**::**resize**(**mat\_a**,** mat\_a\_up**,** cv**::**Size**(**c**,**r**));**  cv**::**resize**(**mat\_b**,** mat\_b\_up**,** cv**::**Size**(**c**,**r**));**  // result Lab image  cv**::**Mat newChannels**[**3**]** **=** **{**channels**[**0**],** mat\_a\_up**,** mat\_b\_up**};**  cv**::**Mat resultImage**;**  cv**::**merge**(**newChannels**,** 3**,** resultImage**);**  //convert back to rgb  cv**::**cvtColor**(**resultImage**,** resultImage**,** CV\_Lab2BGR**);**  // generate colorized image  int pos **=** file\_path**.**find\_last\_of**(**kFileSperator**);**  string file\_name**(**file\_path**.**substr**(**pos **+** 1**));**  cout **<<** "file\_name = " **<<** file\_name **<<** endl**;**  stringstream sstream**;**  sstream**.**str**(**""**);**  sstream **<<** result**->**console\_params**.**output\_path **<<** kFileSperator  **<<** kOutputFilePrefix **<<** file\_name**;**  string outputPath **=** sstream**.**str**();**  resultImage **=** resultImage **\*** 255**;**  cv**::**imwrite**(**outputPath**,** resultImage**);**  **if** **(**size **<=** 0**)** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: inference result size=%d error."**,**  file\_path**.**c\_str**(),** size**);**  **return** HIAI\_ERROR**;**  **}**  // transform results  float **\***res **=** **new** **(**nothrow**)** float**[**size**];**  **if** **(**res **==** **nullptr)** **{**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: new float array failed."**,**  file\_path**.**c\_str**());**  **return** HIAI\_ERROR**;**  **}**  errno\_t mem\_ret **=** memcpy\_s**(**res**,** **sizeof(**float**)** **\*** size**,** out**.**data**.**get**(),**  out**.**size**);**  **if** **(**mem\_ret **!=** EOK**)** **{**  **delete[]** res**;**  ERROR\_LOG**(**"Failed to deal file=%s. Reason: call memcpy\_s failed."**,**  file\_path**.**c\_str**());**  **return** HIAI\_ERROR**;**  **}**  **delete[]** res**;**  **return** HIAI\_OK**;**  **}** |

完整代码请参考general\_post.cpp文件