基于华为自研AI芯片的计算机视觉深度学习算法研究与移植

设计文档

浙江大学

## 1 项目简介和开发目的

深度学习利用多层神经网络结构,从大量数据中学习现实世界中各类事物，越来越被广泛应用于计算机视觉领域，且可以达到不错的效果。用计算机视觉深度算法进行动作检测是目前视频理解方向的研究热点，因为该任务贴近生活，且在监控安防中有潜在的巨大价值，有造福社会、便利生活的巨大潜力。

为了探究基于深度学习的计算机视觉算法在Atlas 200 Developer Kit（版本20.0.0）上运行的可行性，本项目尝试将OpenPose、ST-GCN两种网络模型相结合，实现一个实时动作检测系统，并在Atlas 200 DK上进行移植与部署，观察其运行性能与效果。

## 2相关深度学习算法介绍

本章节将对该项目中所实现的OpenPose、ST-GCN两种算法进行介绍，OpenPose进行人体骨架提取，ST-GCN利用提取出的骨架关键点进行动作识别。

### 2.1 OpenPose

人体关键点通常对应人体上有一定自由度的关节，比如颈、肩、肘、腕、腰、

膝、踝等，通过对人体关键点在三维空间相对位置的计算，来估计人体当前的姿态。

进一步，增加时间序列，看一段时间范围内人体关键点的位置变化，可以更加准确的

检测姿态，估计目标未来时刻姿态，以及做更抽象的人体行为分析，比如判断一个人

是否在打电话等等。

OpenPose 是基于卷积神经网络和监督学习的开源库,可以实现人的面部表情、躯 干和四肢甚至手指的跟踪，不仅适用于单人也适用于多人，同时具有较好的鲁棒性。

OpenPose 处理图像的主要流程是:输入一幅图像，经过卷积网络提取特征，得 到一组特征图，然后分成两个岔路，分别使用 CNN 网络提取 Part Confidence Maps 和 Part Affinity Fields (PAF)。得到这两个信息后，使用图论中的 Bipartite Matching(偶匹配)求出 Part Association，将同一个人的关节点连接起来，由于 PAF 自身的矢量性，使得生成的偶匹配很正确，最终合并为一个人的整体骨架。 最后基于 PAFs 求解 Multi-PersonParsing，通过把 Multi-PersonParsing 问题转 换成图论问题并利用匈牙利算法求解[1]。

### 2.2 ST-GCN

ST-GCN 的网络处理图片流主要分为三部分:归一化、ST-GCN 单元交替处理、 特征分类。

归一化在时间和空间维度下进行，主要是将一个关节在不同帧下的位置特征进行 归一化，以利于算法收敛(关节在不同帧下的关节位置变化很大);同时，在不同 batch 不同帧下的关节位置基本上服从随机分布，不会造成不同 batch 归一化结果 相差太大，而导致准确率波动。

ST-GCN 单元交替使用 ATT(图注意力模型)、GCN(图卷积网络)和 TCN(时间卷 积网络)对图片的关节的特征数(空间维度)和关键帧数(时间维度)进行变换并提 取特征。

最后使用平均池化和全连接层对特征进行分类，输出行为分析结果[2]。

## 3模型训练

本项目使用开源代码进行了模型的训练。其中，SRCNN与FSRCNN采用了原作者提供的训练代码，ESPCN采用了github上的开源代码。另外，在ESPCN的项目中，项目作者也提供了预训练的模型供用户下载。这三种模型都采用Caffe框架进行训练，并在训练前后分别使用了MATLAB进行数据集的制作与模型的推演测试。以下几节将对模型训练部分进行简要介绍，详细的训练步骤可参考完整的代码，三种模型的代码页面链接分别为：[SRCNN](http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html)、[FSRCNN](http://mmlab.ie.cuhk.edu.hk/projects/FSRCNN.html)与[ESPCN](https://github.com/wangxuewen99/Super-Resolution/tree/master/ESPCN)。

### 3.1 网络结构定义

在启动训练前，需要编写prototxt文件用以描述网络的结构。本项目所使用的三个算法模型的放大倍数都为3，其结构在第2章中已经给出。需要注意的是，由于Caffe框架中不存在与bicubic和亚像素卷积层中像素重排操作（以下简称subpixel）对应的算子，因此这两个操作需要在Caffe框架之外执行。

在实际应用时，对于SRCNN网络，需要先将低分辨率图像进行bicubic插值放大，再作为网络的输入进行计算；而对于ESPCN网络，输入为低分辨率图像，但是需要对网络的输出再执行一个subpixel像素重排的操作才能得到放大的图像；而FSRCNN网络则无需改动，网络的输入为低分辨率图像，输出即为放大后的图像。图3-1用圆角矩形框表示了各算法中具体的Caffe网络部分。

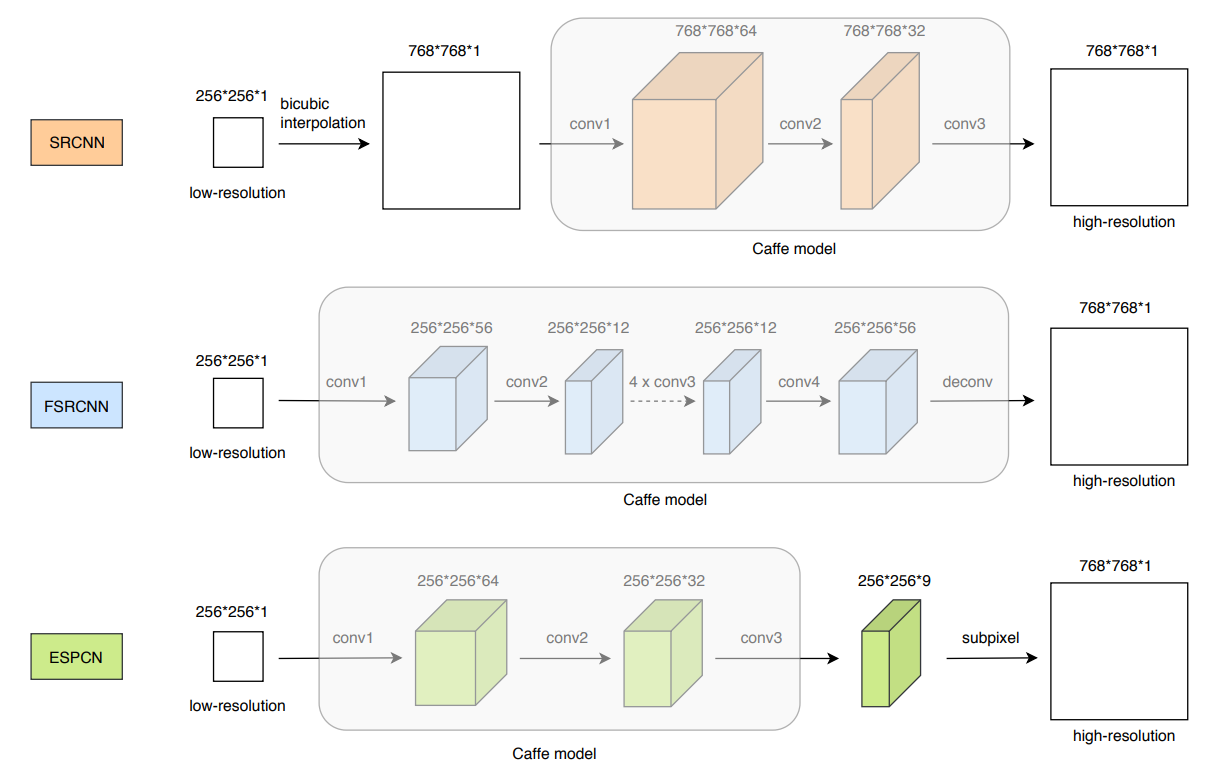


图3-1 各算法中Caffe网络部分

当然，图像在送进网络前，需要进行预处理操作，将0~255的像素值归一化成0~1间的浮点数；对于放大后的结果，还需要进行后处理操作，将输出的数据值（大部分在0~1之间）再转化为0~255的8位像素值，这样才能得到最后的高分辨率图像。

训练网络所需的net.prototxt与solver.prototxt文件，作者在源代码中已有提供；而对于实际部署时所使用的deploy.prototxt网络结构文件，为了保证卷积后特征图尺寸不改变，我们为其增加了padding参数，从而保证最后的高分辨率图像与低分辨率图像的尺寸成倍数关系，三个模型的deploy.prototxt文件内容参见附录一。

### 3.2 数据集制作

以上三个模型都采用了91-image数据集作为训练集，91-image数据集包含了91张图片，是图像超分辨率领域中常用的训练集。另外，FSRCNN还采用General-100数据集作为补充训练集，并对训练集做了数据增强，提高了模型的泛化能力。91-image数据集与General-100数据集在FSRCNN的代码连接中可进行下载，这两个数据集的部分图像分别如图3-2、3-3所示。

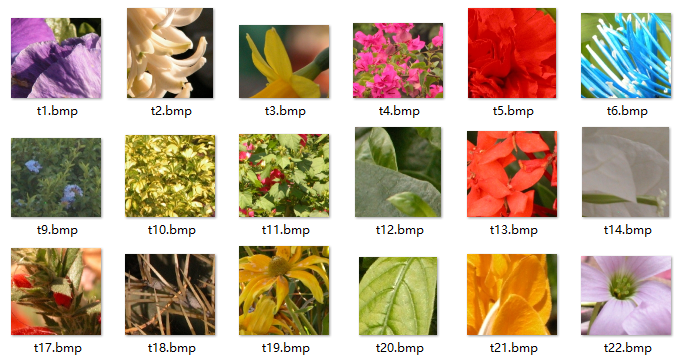


图3-2 91-image数据集部分图像

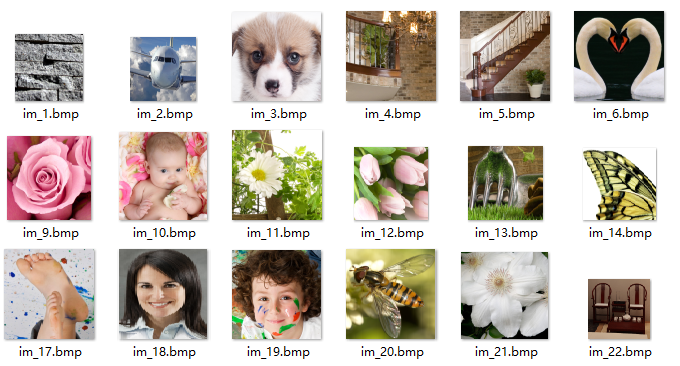


图3-3 General-100数据集部分图像

准备好数据集后，接下来使用MATLAB进行训练数据集的处理和制作。

首先，由于三种模型都是以灰度图片作为模型的输入，因此需要对原数据集进行灰度化处理，使用rgb2ycbcr函数进行格式转换，并提取出Y通道数据作为所需数据。

接着，使用imresize函数进行图像下采样，以得到低分辨的图像数据。这里需要注意的是，MATLAB的下采样函数自带“抗混叠滤波”效果，这会使得下采样得到的图像更加平滑，更接近真实数据。

接下来，需要根据不同的模型，制作各自的输入（低分辨率）与输出（高分辨率）图像数据对。如上一节所述，在Caffe框架下，SRCNN不包含bicubic操作，ESPCN不包含subpixel操作，因此，对于SRCNN网络，需要将低分辨率图像先进行bicubic插值，再作为输入数据进行训练；对于ESPCN网络，需要将高分辨率图像进行一次反向的subpixel操作，作为模型的输出进行训练。

最后，对上一步得到的输入与输出图像进行分块操作，进一步提取出尺寸更小的图像数据对写入hdf5文件，作为最终的训练数据。

### 3.3 模型训练与测试

准备好数据集并定义好网络结构与训练参数文件solver.prototxt后即可启动训练。最后我们得到所需的caffemodel权重文件，并结合附录中给出的推演时的网络结构deploy.prototxt文件，可作为后续Atlas 200 DK模型转换的输入。

另外，作者提供的源代码中还给出了使用MATLAB在Sep5数据集上进行推演测试的代码，这里不再进行过多介绍。下一节我们将会介绍如何在Atlas 200 DK上部署实现这三个网络模型，并在第5节中给出Sep5数据集在Atlas 200 DK上进行推演的结果。

## 4 算法模型在Atlas 200 DK上的实现

从本节开始，我们将会介绍如何将之前训练得到的模型部署至Atlas 200 DK开发板上，最终实现一个较为完整的实时动作检测系统。本节的内容包括：硬件平台简介、离线模型转换的步骤及要点、系统各引擎模块的搭建，以及最后介绍一下系统的使用流程。

### 4.1 硬件平台

本项目所使用的硬件平台为Atlas 200 DK开发板，Atlas 200 DK开发板硬件外形如图4-1所示。

### 4.2 离线模型转换

在完成超分辨模型的训练，得到全精度的算法模型之后，首先需要进行离线模型转换这一步骤，将模型转换为Ascend 310芯片支持的模型（Davinci架构模型），才可进一步将其部署在Atlas 200 DK开发板上。



图4-1 Atlas 200 DK开发板

在Mind Studio界面中，通过图形化的离线模型转换工具，可调用Matrix提供的模型管家接口将Caffe模型转换为Davinci架构模型，完成离线模型转换。图4-2展示了将训练完成的超分辨模型（以FSRCNN为例）转换为Davinci模型的参数配置。

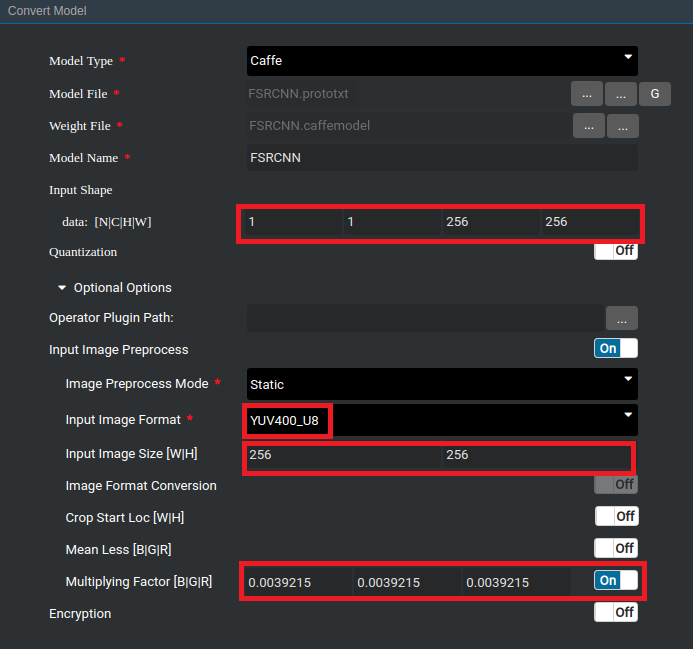


图4-2 FSRCNN离线模型转换

在模型转换的过程中需要注意以下几个参数的配置，在图4-2中也以红色框进行了标注。

1. Input Shape：以FSRCNN放大256×256的图像为例，NCHW需要设置为[1, 1, 256, 256]，其中C通道为1是由于以上训练得到的超分辨率模型都是针对灰度图像进行处理，若是需要展示彩色图像结果，可参考4.4节中的处理方式。另外，如果使用SRCNN模型，此处的H与W需要设置为768，即低分辨率图像经过bicubic放大后的尺寸。
2. Input Image Format：由于模型的输入是灰度图像，并且对于超分辨项目，不需要使用DVPP模块进行预处理，因此此处需要指定为格式为YUV400\_U8。
3. Input Image Size：与Input Shape中的HW相同。
4. Multiplying Factor：由于在模型训练时，输入数据的范围为[0, 1]，因此此处需将[0, 255]的数据乘上1/255进行归一化。

对于本项目中使用到的三种模型（SRCNN、FSRCNN以及ESPCN），离线模型转换的参数配置基本相同，唯一区别在于，转换SRCNN模型时，HW需要设置为输入图像尺寸的3倍，即低分辨率图像经过bicubic放大后的尺寸。

### 4.3 系统模块搭建

本项目在Atlas 200 DK硬件平台上搭建了一个基于计算机视觉深度学习算法的实时动作检测系统。该系统将摄像头捕获的实时图像数据作为输入，处理后送入模型OpenPose和ST-GCN模型进行推理，最后将带有结果的图像通过Presenter Server在浏览器上显示出来。系统模块（引擎）包括三个部分：图像数据模块、推演模块以及后处理模块，其连接图如图4-3所示。

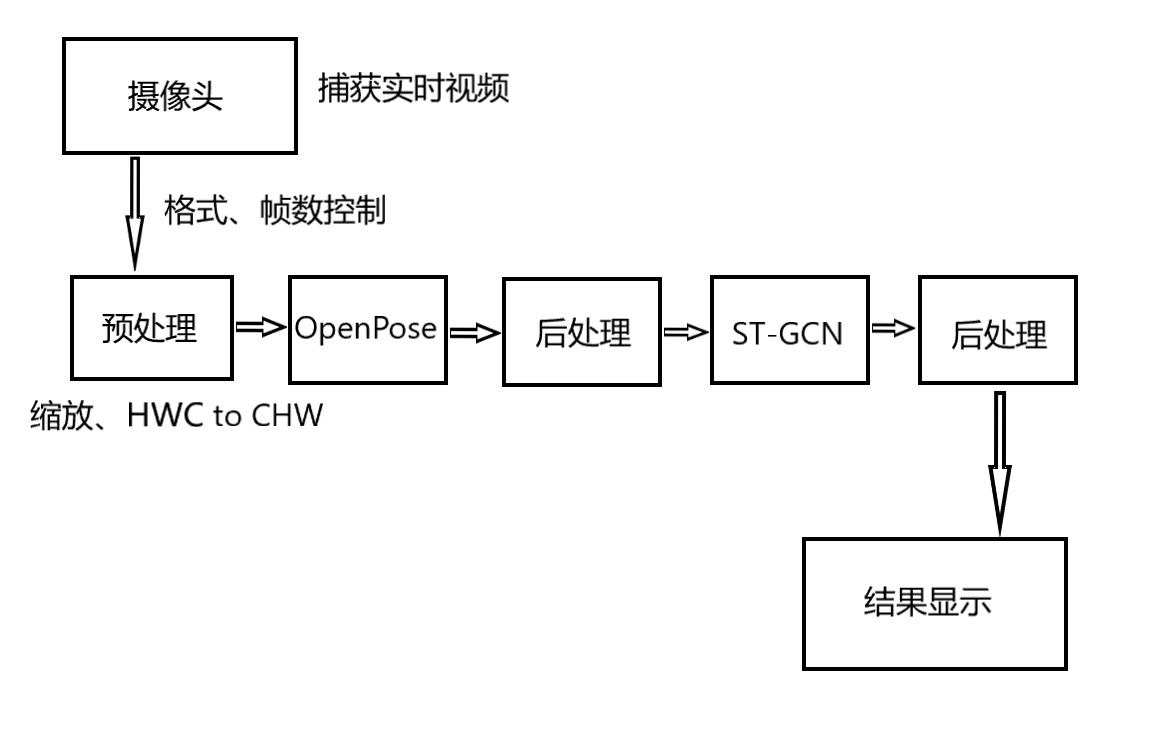


图4-3实时动作检测系统

图像数据模块的功能主要是图像文件的读取。首先，使用OpenCV导入所需要处理的低分辨率图像数据，由于本项目所使用模型的输入图像通道都为1，因此此处需要以灰度形式进行图像导入。对于SRCNN网络，由于模型的输入为bicubic放大后的图像，所以此处导入图像数据后需要继续使用OpenCV的resize函数进行bicubic插值放大（FSRCNN和ESPCN网络则无需这一步操作）。接下来即可将数据发送至下一个模块。

推演模块顾名思义，主要处理模型推理的相关部分，包括模型管家的初始化，离线模型的加载与使用、前后模块的数据交互等。首先，根据用户选择的模型类型以及给定的模型路径，离线模型管家会加载已完成转换的图像超分辨率模型SRCNN、FSRCNN和ESPCN及其权值。接下来即可调用模型管家的计算接口进行推演，由于在离线模型转换时，我们配置了Multiplying Factor等参数，因此在进行实际推演之前，还会启动AIPP在AI Core上完成图像的预处理。得到网络的输出数据后，即可送入后处理模块进行进一步处理。

后处理模块的工作包括数据范围转化、ESPCN网络的subpixel操作、彩色图像的生成以及图像的存储等。首先，接收到推演网络的输出数据之后，需要进行后处理操作，将数据范围（大部分数据为[0, 1]之间的浮点数）还原为[0, 255]，并转换为八位像素数据值。如果当前处理的网络为ESPCN，则需要对输出的数据进行subpixel像素重排的操作。例如，若此时网络输出的数据为[1, 256, 256, 9]，经过subpixel操作（实际上是对一维数组进行数据重新排列的过程），可得到[1, 768, 768, 1]形状的图像数据，即为最后的高分辨率图像。另外，该模块中还进行了使用单通道模型生成彩色图像（详见4.4节），以及使用bicubic图像插值算法生成对比图片的相关操作。最后，使用OpenCV将生成的图像进行保存。

### 4.4 彩色图像处理

通常，在图像超分辨率模型（包括本项目使用的三种模型）当中，输入图像与输出图像的通道数往往都为1，因此仅适用于单通道图像的放大。为了将这类单通道模型用于处理彩色图像，可采用图4-4的处理流程。

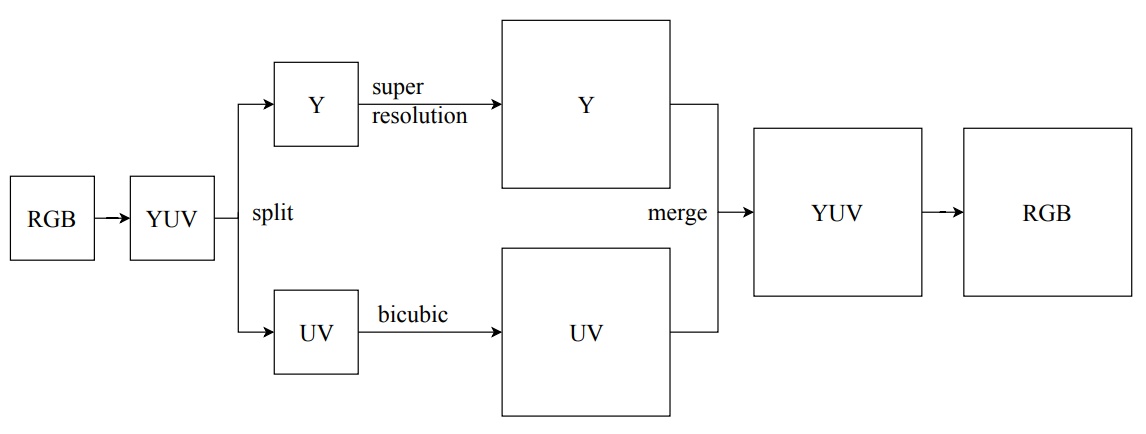


图4-4 超分辨网络彩色图像处理流程

首先，将RGB格式的图像转换为YUV格式，然后将通道进行分离。对于亮度通道Y，使用图像超分辨率算法进行放大，得到一张高分辨率灰度图像，而对于色度通道UV只需使用bicubic插值算法进行处理，这是由于人眼对亮度信号的空间分辨率大于对色度信号的空间分辨率，因此对于相对不敏感的色度通道，使用插值算法即可满足视觉要求。最后，将三个通道的高分辨率数据进行合并，再转换回RGB格式，即可完成可使用单通道模型进行彩色图像的超分辨率应用过程。

### 4.5 系统使用流程

在完成以上内容的设计之后，本节将对所搭建的超分辨率图像转换系统的实际使用流程进行简要介绍，详细的应用部署步骤、执行命令等相关内容请参考代码介绍及后续文档。

在运行之前，需要先进行环境配置以及代码编译等步骤，完成应用在Atlas 200 DK开发板上的部署。部署完成后，即可进入实际的使用，系统的使用流程包括以下四个步骤：模型转换、图像与模型上传、应用执行以及查看结果。

1. 模型转换

首先，根据需要进行处理的低分辨率的图像的宽高尺寸，准备相应大小的Davinci模型。例如，若低分辨率图像的宽高都为256，则需要准备输入形状为[1, 1, 768, 768]的SRCNN模型，或者输入形状为[1, 1, 256, 256]的FSRCNN或ESPCN模型，离线模型转换的具体过程和参数配置可参考4.2节。

1. 图像与模型上传

将需要使用的已经转换好的Davinci离线模型文件与低分辨率图像上传至Atlas 200 DK开发板。

1. 应用执行

在Mind Studio所在Ubuntu服务器中，以HwHiAiUser用户SSH登录到开发板侧，进入本项目对应的可执行文件所在路径，接着运行python脚本启动应用程序。在运行脚本时，还需要选择所使用的模型类型，以及是否需要输出彩色结果。

1. 查看结果

在浏览器中输入presenter server的IP地址192.168.1.134：7008，点击要查看的链接即可。

## 5 实现结果及性能展示

为了查看本项目搭建的超分辨率图像系统的实现效果以及在硬件平台上的性能，本章使用Set5数据集在Atlas 200 DK上进行验证。Set5数据集包含五张风格不同的图片，被广泛用在超分辨率图像算法的性能验证，图5-1展示了Set5数据集的全部图片。

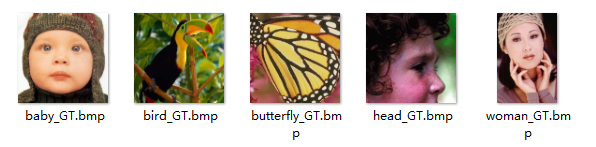
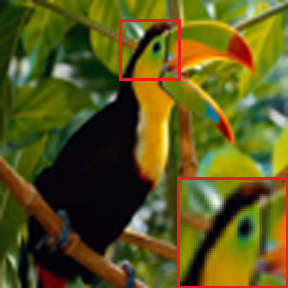
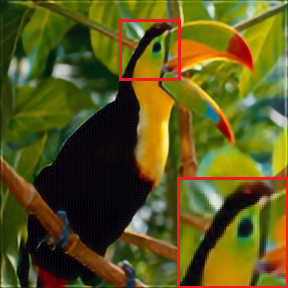


图5-1 Set5数据集

### 5.1 实现结果

按照4.5节所述的使用流程，我们使用Set5数据集在部署完成的Atlas 200 DK开发板上进行了验证。对于本项目的三种模型，Set5数据集都能得到正确的超分辨结果，并且相比于传统的bicubic插值算法，性能都有明显的提升。

图5-2展示了FSRCNN超分辨网络在Atlas 200 DK上运行得到的结果，其中LR表示放大前的低分辨率图像，bicubic和FSRCNN分别表示使用插值算法和FSRCNN超分辨网络得到的结果，可以看出FSRCNN网络相比于bicubic插值算法能实现更好的效果。

LR bicubic FSRCNN

LR bicubic FSRCNN

图5-2 FSRCNN超分辨网络在Atlas200DK上运行得到的结果

### 5.2 运行速度

本节以宽高为256×256的图像作为输入，记录了三种超分辨率网络在不同平台上的推演时间，如表5-1所示。其中，网络在CPU/GPU上的推演时间使用Caffe框架提供的计时方法time获得，而在Atlas 200 DK上，则是通过Profiling工具采集性能数据，以Model Statistic一栏中的Inference End Time - Inference Start Time作为推演时间。

表5-1 超分辨率算法模型的推演时间 (ms)

|  |  |  |  |
| --- | --- | --- | --- |
|  | CPU (Intel i7-7700) | GPU (GTX 1080 Ti) | Atlas 200 DK |
| SRCNN | 1336.18 | 8.37 | 20.69 |
| FSRCNN | 208.34 | 2.70 | 4.96 |
| ESPCN | 198.23 | 0.87 | 2.89 |

注：

输入为256×256图像，输出为768×768图像。

Atlas 200 DK使用的版本为1.3.T34.B891。

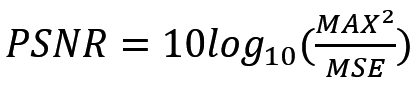
以上时间仅为网络推演耗时。对于SRCNN，推演时间不包括前处理bicubic的耗时；对于ESPCN，不包括后处理subpixel的耗时。

另测得subpixel操作在Atlas200DK后处理模块中耗时约40ms。

从表5-1的数据结果可以看出，本项目使用的三种超分辨率网络在Atlas 200 DK上的运行速度已远超其在Intel i7-7700 CPU上的速度，但相比于GTX 1080 Ti GPU还是存在一些差距。

### 5.3 PSNR

PSNR (Peak Signal to Noise Ratio) 峰值信噪比可用于衡量图像失真程度，两张图像之间的PSNR值越大，则表示它们越相似，PSNR的计算公式如(5-1)所示，其中，*MAX*表示图像颜色的最大数值，对于8bit图像取值为255，*MSE*表示两张图像间的均方差。本节将使用Set5数据集在Atlas 200 DK上进行测试，考察不同网络模型的PSNR指标。

 (5-1)

超分辨率图像算法的PSNR测试通常在灰度通道上进行，因此，我们首先在MATLAB中对Set5数据集进行处理，包括图像灰度化以及下采样，得到高、低分辨率灰度图像数据对。接下来，将5张低分辨率图像分别送入对应大小的网络模型中进行推演，得到超分辨网络的输出结果。最后，根据公式(5-1)进行计算，得出各算法模型在Atlas 200 DK上的PSNR。另外，计算PSNR时需要先对图像每个边界进行6个像素的裁剪，忽略图像边界位置处可能存在的数据误差。例如图5-3所示，SRCNN生成的图像在边界处可能存在一圈“黑边”，因此计算PSNR时对其进行了裁剪。

图5-3 图像边界处可能存在的数据误差

最后，使用Set5数据集在Atlas 200 DK上进行测试得到的PSNR结果如表5-2所示。

表5-2 超分辨率算法模型的实现PSNR结果（dB）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bicubic | SRCNN | FSRCNN | ESPCN |
| Atlas 200 DK | 30.62 | 31.96 | 33.10 | 32.52 |
| 论文效果 | 30.39 | 32.39 | 33.16 | 33.13 |

## 6 优化尝试及后续展望

从第5节的结果来看，本项目所使用的图像超分辨模型算法在Atlas 200 DK上的实现效果良好，但是在运行速度上或许仍存在可优化的空间。为了进一步提升算法模型的实现速度，减小硬件资源占用，本节尝试使用剪枝的方法对模型进行压缩，观察性能是否有所变化。在本节最后，我们也提出了一些本项目目前的不完备之处，以便后续进一步完善与优化。

### 6.1 模型剪枝

本节使用基于滤波器权重大小的剪枝方法[7]对本项目使用的三个网络进行了压缩。剪枝前后SRCNN、FSRCNN以及ESPCN的网络结构分别如表6-1、表6-2和表6-3所示。

表6-1 剪枝前后SRCNN网络结构

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Kernel (Co, Ci, Hk, Wk) | Bias | Activation |
| Conv1 | (64, 1, 9, 9) → (**48**, 1, 9, 9) | (64) → (**48**) | ReLU |
| Conv2 | (32, 64, 1, 1) → (**24**, **48**, 1, 1) | (32) → (**24**) | ReLU |
| Conv3 | (1, 32, 5, 5) → (1, **24**, 5, 5) | (1) |  |

表6-2 剪枝前后FSRCNN网络结构

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Kernel (Co, Ci, Hk, Wk) | Bias | Activation |
| Conv1 | (56, 1, 5, 5) -> (**48**, 1, 5, 5) | (56) -> (**48**) | PReLU |
| Conv21 | (12, 56, 1, 1) -> (**8**, **48**, 1, 1) | (12) -> (**8**) | PReLU |
| Conv22 | (12, 12, 3, 3) -> (**8**, **8**, 3, 3) | (12) -> (**8**) | PReLU |
| Conv23 | (12, 12, 3, 3) -> (**8**, **8**, 3, 3) | (12) -> (**8**) | PReLU |
| Conv24 | (12, 12, 3, 3) -> (**8**, **8**, 3, 3) | (12) -> (**8**) | PReLU |
| Conv25 | (12, 12, 3, 3) -> (**8**, **8**, 3, 3) | (12) -> (**8**) | PReLU |
| Conv26 | (56, 12, 1, 1) -> (**48**, **8**, 1, 1) | (56) -> (**48**) | PReLU |
| Deconv | (1, 56, 9, 9) -> (1, **48**, 9, 9) | (1) |  |

表6-3 剪枝前后ESPCN网络结构

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Kernel (Co, Ci, Hk, Wk) | Bias | Activation |
| Conv1 | (64, 1, 5, 5) -> (**48**, 1, 5, 5) | (64) -> (**48**) | TanH |
| Conv2 | (32, 64, 3, 3) -> (**24**, **48**, 3, 3) | (32) -> (**24**) | TanH |
| Conv3 | (9, 32, 3, 3) -> (9, **24**, 3, 3) | (9) |  |

经计算，SRCNN网络剪枝前需要存储的权重（包括偏置）参数量为8129，剪枝后为5713，减少了29.7%，FSRCNN和ESPCN网络参数量也分别减少了34.3%和40.2%。

以宽高为256×256的图像作为输入，剪枝前后三个网络在各平台上的推演时间如表6-4所示。

表6-4 剪枝前后SRCNN网络推演时间 (ms)

|  |  |  |  |
| --- | --- | --- | --- |
|  | CPU (Intel i7-7700) | GPU (GTX1080 Ti) | Atlas 200 DK |
| SRCNN | 1336.18 | 8.37 | 20.69 |
| SRCNN-pruned | **1034.97** | **6.73** | 23.32 |
| FSRCNN | 208.34 | 2.70 | 4.96 |
| FSRCNN-pruned | **172.69** | **2.30** | **4.60** |
| ESPCN | 198.23 | 0.87 | 2.89 |
| ESPCN-pruned | **163.79** | **0.69** | 2.91 |

从表6-4可以看出，在Atlas 200 DK开发板上，相比剪枝前，FSRCNN的推演时间从4.96ms缩短为4.60ms，推演速度有7%的提升；ESPCN剪枝前后推演时间几乎无变化；而SRCNN的推演时间反而变长了一些。

另外，使用剪枝后的网络在Set5数据集上进行PSNR测试，剪枝后的FSRCNN会有0.18dB性能损失，而SRCNN与ESPCN分别有0.09dB与0.06dB性能提升，这说明SRCNN与ESPCN原模型存在部分冗余，可通过调整网络结构得到进一步优化。

### 6.2 后续展望

本项目在Atlas 200 DK开发板上实现了一个简易的超分辨率图像转换系统，但目前仍存在一些可优化的空间，例如，可通过自定义算子开发，将bicubic以及subpixel操作转化为Davinci模型的一部分，从而利用AI Core或AI CPU进行运算加速，还可尝试使用模型量化功能进行进一步提速等。

## 7 参考文献

[1]  Cao Z, Hidalgo G, Simon T, et al. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields[J]. arXiv preprint arXiv:1812.08008, 2018.

[2]  Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition[C]//Thirty-second AAAI conference on artificial intelligence. 2018.