LaTeX 中文书籍模板

作者: Nelson Cheung

zhangjunyu@nelson-cheung.cn

如果你有任何问题或评论,
可以通过邮件 zhangjunyu@nelson-cheung.cn 联系我
Happy LaTeX-ing!

作者自序

这是一个书籍模板。模板希望能够排版平时技术报告的主要内容，如正文、公式、代码等。文案选自《成都理工大学本科生实验报告 LaTeX 模板》

Nelson Cheung
2021 年 12 月 17 日

目录

第一章　本书内容的基本环境配置

本书是目标是介绍一个 AI 换脸系统的开发流程，主要是讲解整个系统的整体结构、流程，核心算法及加速算法。并附带完整代码的 Git 地址。读者学完后可以开发这个系统作为基础，有想法的读者可以进行扩充，就可以开发一个商业软件。我使用的开发环境是操作系统是 wsl，即在 windows10 系统上安装了 Ubuntu 22.04 子系统，具体安装方法，可以自行查阅相关资料。首先安装的开发工具 CUDA，我选的版本是 11.8，不算特别新，也不算特别古老。适合于当前流行的 Nvida 显卡。同时安装的 CUDNN，这是神经网络的开发库。具体安装方法可以参考`https://gist.github.com/MihailCosmin/affa6b1b71b43787e9228c25fe15aeba?permalink_comment_id=4850089`。演示了 CUDA 和 CUDNN 的安装及验证代码。TensorRT 是 NVIDIA 推出的用于深度学习推理加速的高性能推理引擎。它可以将深度学习模型优化并部署到 NVIDIA GPU 上，实现低延迟、高吞吐量的推理过程。TensorRT 主要用于加速实时推理任务，如物体检测、图像分类、自然语言处理等。所需的库是：

- **CUDA**
- **TensorRT**
- **OpenCV**：

下面就逐一讲解每个库的安装方法，

1. 安装 CUDA 我们选择安装的 CUDA 版本是 11.8。它是一个比较通用的版本。安装过程如下：

```
mkdir Downloads && cd Downloads #为了模仿Ubuntu系统，我们自己创建了Downloads文件
sudo apt update && sudo apt upgrade -y #更新一些库
sudo apt install g++ freeglut3-dev build-essential libx11-dev libxmu-dev libxi-
sudo wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x
```

APT 软件包优先级配置文件，确保系统从正确的仓库（如 NVIDIA 官方仓库）安装 CUDA 工具包

```
sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600 #移动
sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cud
sudo add-apt-repository "deb https://developer.download.nvidia.com/compute/cuda
sudo apt install cuda-11-8 -y
echo 'export PATH=/usr/local/cuda-11.8/bin:$PATH' >> ~/.bashrc
```

```
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64:$LD_LIBRARY_PATH'
source ~/.bashrc
sudo ldconfig
nvcc -V #查看安装结果，如果显示 "Cuda compilation tools, release 11.8, V1
```

接着还要安装一下 CUDNN 库。下面指令仍然是在 /Downloads 路径下面执行。

```
CUDNN_TAR_FILE="cudnn-linux-x86_64-8.7.0.84_cuda11-archive.tar.xz"
sudo wget https://developer.download.nvidia.com/compute/redist/cudnn/v8.7.0
sudo tar -xvf ${CUDNN_TAR_FILE}
sudo tar -xvf ${CUDNN_TAR_FILE}  #解压
sudo mv cudnn-linux-x86_64-8.7.0.84_cuda11-archive cuda
sudo cp -P cuda/include/cudnn*.h /usr/local/cuda-11.8/include
sudo cp -P cuda/lib/libcudnn* /usr/local/cuda-11.8/lib64/
sudo chmod a+r /usr/local/cuda-11.8/lib64/libcudnn*
```

查看 CUDNN 的版本，执行如下命令

```
cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

我主要参考这个网页https://gist.github.com/MihailCosmin/affa6b1b71b43787e9228c25fe15aeba?permalink_comment_id=4850089。

2. 安装 TensorRT 安装 TensorRT 比较简单，就是下载库文件，解压，设置环境变量即可。

```
#下载到 ~/Downloads
cd ~/Downloads
wget https://developer.nvidia.com/downloads/compute/machine-learning/tensorrt/
#解压到  /usr/local/
sudo tar -xzvf TensorRT-8.6.1.6.Linux.x86_64-gnu.cuda-11.8.tar.gz -C /usr/local
```

设置环境变量

```
sudo vim ~/.bashrc
export LD_LIBRARY_PATH=/path/to/tensorrt/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/TensorRT-8.6.1.6/lib:$LD_LIBRARY_PATH
source ~/.bashrc
```

3. 安装 OpenCV(CUDA 版)

OpenCV 是出来图片的基础库，我们使用 CUDA 版本，有利于加速。步骤如下：

其他的几个辅助库，它们有：

sudo apt update sudo apt upgrade sudo apt install build-essential cmake pkg-config unzip yasm git checkinstall sudo apt install libjpeg-dev libpng-dev libtiff-dev sudo apt install libavcodec-dev libavformat-dev libswscale-dev sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev sudo apt install libxvidcore-dev libx264-dev libmp3lame-dev libopus-dev sudo apt install libmp3lame-dev libvorbis-dev sudo apt install ffmpeg sudo apt install libva-dev sudo apt install libdc1394-25 libdc1394-dev libxine2-dev libv4l-dev v4l-utils sudo ln -s /usr/include/libv4l1-videodev.h /usr/include/linux/videodev.h sudo apt-get install libgtk-3-dev sudo apt-get install libtbb-dev sudo apt-get install libatlas-base-dev gfortran sudo apt-get install libprotobuf-dev protobuf-compiler sudo apt-get install libgoogle-glog-dev libgflags-dev sudo apt-get install libgphoto2-dev libeigen3-dev libhdf5-dev doxygen 进入 /Downloads 目录下载 Opencv 源码并进行编译

cd /Downloads mkdir opencv cd opencv wget -O opencv.zip https://github.com/opencv/opencv/archive/refs/tags/4.10.0.zip wget -O opencv$_{c}ontrib.ziphttps$: $//github.com/opencv/opencv_{c}ontrib/archive/refs/tags/4.10.0.zipunzipopencv.zipunzipopencv_{c}ontrib.zipcdopencv-4.10.0mkdirbuildcdbuild$

cmake -D CMAKE$_{B}UILD_{T}YPE = RELEASE - DCMAKE_{I}NSTALL_{P}REFIX = /usr/local - DWITH_{T}BB = ON - DENABLE_{F}AST_{M}ATH = 1 - DCUDA_{F}AST_{M}ATH = 1 - DWITH_{C}UBLAS = 1 - DWITH_{C}UDA = ON - DBUILD_{o}pencv_{c}udacodec = OFF - DWITH_{C}UDNN = ON - DOPENCV_{D}NN_{C}UDA = ON - DCUDA_{A}RCH_{B}IN = 7.5 - DWITH_{V}4L = ON - DWITH_{Q}T = OFF - DWITH_{O}PENGL = ON - DWITH_{G}STREAMER = ON - DOPENCV_{G}ENERATE_{P}KGCONFIG = ON - DOPENCV_{P}C_{F}ILE_{N}AME = opencv.pc - DOPENCV_{E}NABLE_{N}ONFREE = ON-DOPENCV_{E}XTRA_{M}ODULES_{P}ATH = /home/an/Downloads/opencv/opencv_{c}ontrib-4.10.0/modules - DINSTALL_{P}YTHON_{E}XAMPLES = OFF - DINSTALL_{C}_{E}XAMPLES = OFF - DBUILD_{E}XAMPLES = OFF.. nprocmake - j8sudomakeinstall$

c git clone https://github.com/fmtlib/fmt.git cd fmt mkdir build cd build cmake .. make -j8 sudo make install

c find$_package(FMTREQUIRED)target_link_libraries(xxxfmt :: fmt)$

git clone https://github.com/gabime/spdlog.git cd spdlog mkdir build cd build cmake .. make -j16

fmt 库是 C++ 中格式化输出的现代解决方案，提供了类型安全、易用性和高性能。fmt 库比传统 sprintf 和 iostream 更快：安装方法：

然后再 cmakelists 补上 fmt 的 include 和 lib

spdlog spdlog 是一个快速的 C++ 日志库，具有高性能、易用性和丰富的功能特性。spdlog 是 header-only 的，可以直接下载并包含头文件：include "spdlog/spdlog.h" 下载、编译源码:

一个实例。在安装了这些库之后，我们可以举例子演示一下使用方法。参数项目的 github 网址如下。参考项目网址 `https://github.com/cyrusbehr/tensorrt-cpp-api` 主要演示的功能如下：

- **如何生成 TensorRT engine 文件**
- **如何声明一个简单的优化简表**
- **推理精度**
- **如何读写 GPU 显存**
- **如何处理单个或者多个输入/输出**

首先看一下简单流程：

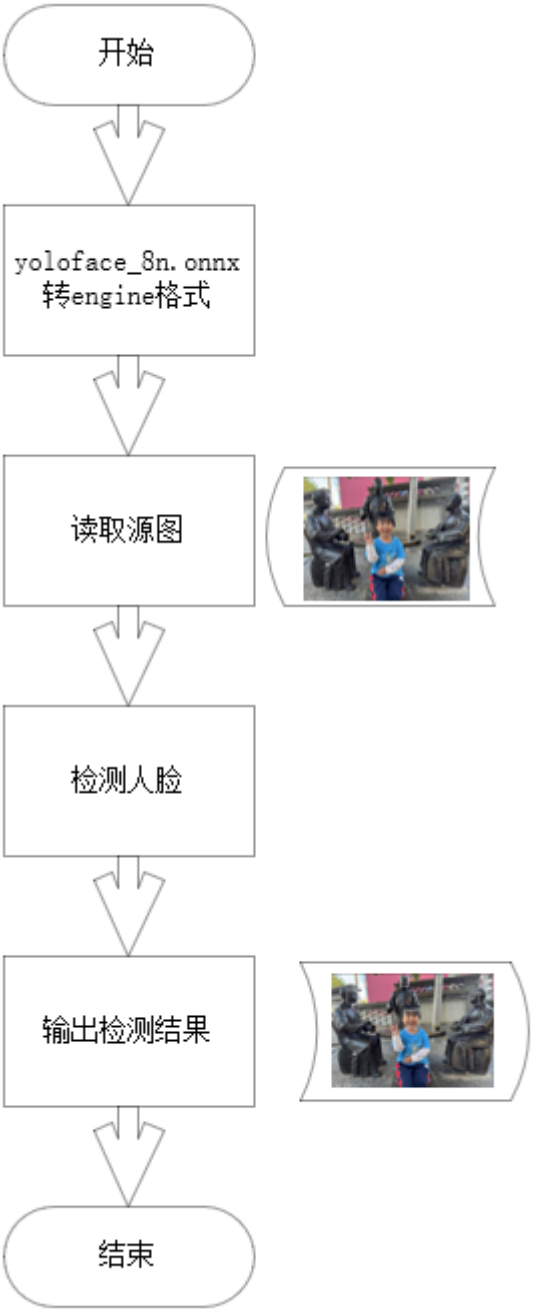


图 1.1: Title of picture

– **套装发行版：**俗称 LaTeX 的核心，常见的有 CTeX 套装、TeXlive 套装（作者使用）、MacTeX 套装等等，本文使用 TeXlive2018 版本，下载地址为`http://www.latexstudio.`

net/texsoftware。

- **编译引擎**：也是编译时点的按钮，本文采用魔法注释设置了编译引擎为 xelatex，也是中文文档排版常用的编译引擎，其他的编译引擎有 pdflatex、latex、bibtex 等等。
- **编辑器**：顾名思义，编辑器就是编辑 tex 文档所用的软件，编写本文档使用的是 TeXstudio，也是作者推荐的一款编辑器，网上可以免费下载。下载 tl2018 时也会自带编辑器，名为 TeXwork，也是一款很好的编辑器，编辑器的种类很多，甚至于 word 都可以拿来写 tex 代码，只不过 tex 编辑器拥有自带的语法高亮、自动补全等快捷方式。

阅读或使用本模板时建议与 tex 文档对比学习。

1.1 文件结构

解压压缩包后，内含的文件内容与结构如下

- **样本.pdf**：本文件，内包括写作方法与注意事项；
- **CDUT_ Lab_ report.tex**：实验报告主文件，包括报告整体结构与一些全局定义；
- **attachment 文件夹**：内含文件 attachment.pdf 与 attachment.docx，为实验报告中的心得体会，具体书写方式详见文末心得体会处；
- **body 文件夹**：内含各章节 tex 文档（如 section_1.tex），方便起见，每一章（每一实验）单独使用一个文档书写；
- **figure 文件夹**：文档中各类图片依照命名规范至于此文件夹中，方便统一管理，其中 CDUT.png 为文档必须图片，请勿删除；
- **preface 文件夹**：内含文档 cover.tex 为首页封面与目录设置；
- **setup 文件夹**：内含文档 format.tex 与 package.tex，分别为全局设置文档与宏包文档，模板使用了模板必须与一些常用宏包，如有特殊需要可以自行添加在文档内，方便统一管理；
- **texclear.bat**：用于清理辅助文件。

1.2 由 onnx 到 engine

换脸的基础动作就是检测到人脸。对源图和目标图首先检测到人脸才能交换其特征，然后才能进行相关处理。第一个网络就是使用 Yolo 网络进行人脸检测。yoloface$_8n.onnx$ $YOLO(YouOnlyLoo$

可使用 Netron 工具可视化.onnx 文件，直接查看输入/输出层名称和维度。下图是 Netron 截图的输入输出部分：

图 1.2: Title of picture

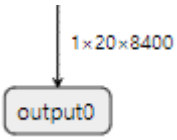


图 1.3: Title of picture

我们设计了一个类，用来处理 onnx 转成 engine 格式，并且处理图片，检测人脸。类图如下：

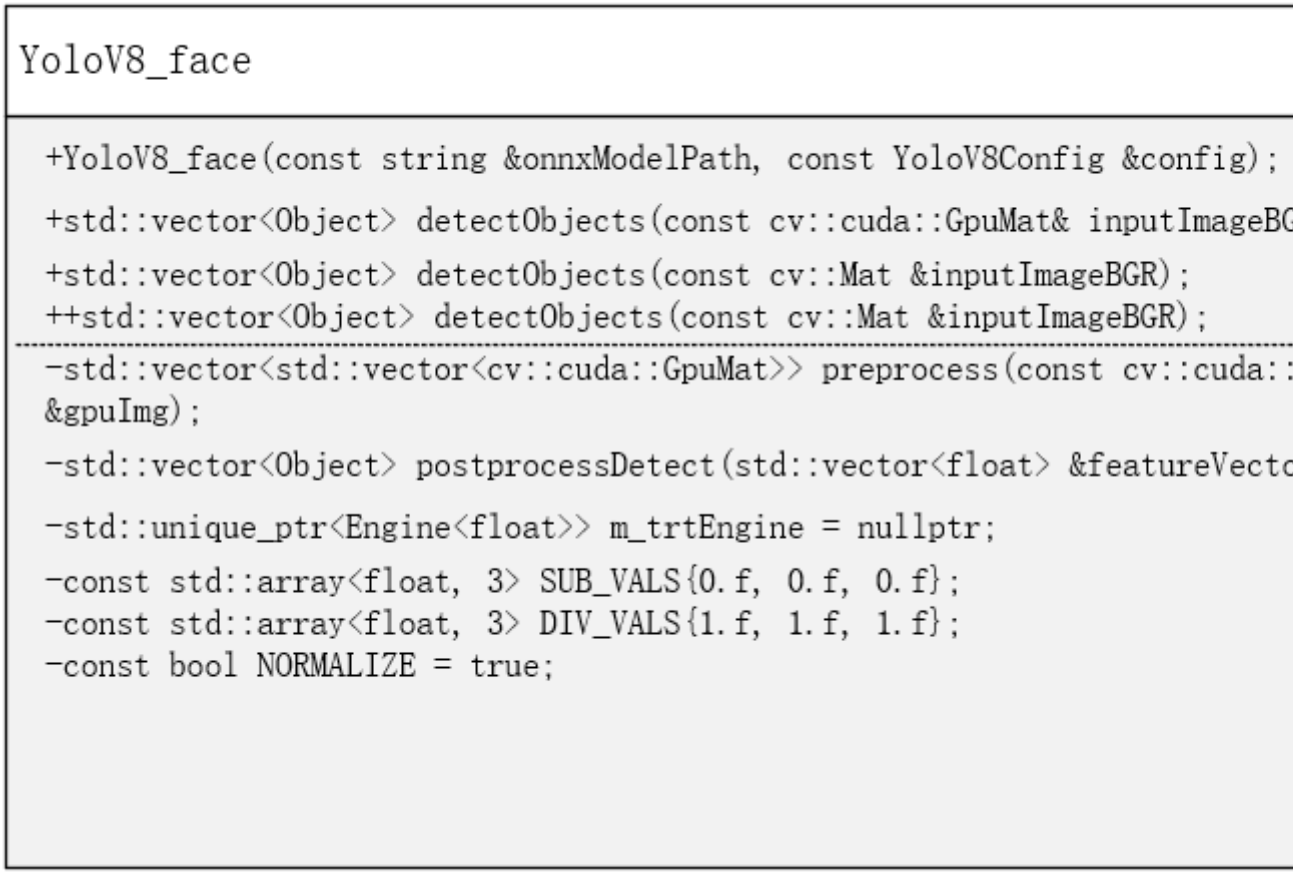


图 1.4: Title of picture

类说明：类名：YoloV8$_face$ $.onnx$ $.engine$

成员函数:+YoloV8$_face(const string onnxModelPath, const YoloV8Config config)$ $onnx$ $.engine$ $onnxM$

第二章　人脸检测算法

换脸算法的输入是两张图片，一张称之为”源图“（source），一般为客户的照片，另一张为”目标图“（target），一般可选明星的写真图。通过换脸算法生成换脸图。流程如下：

我们需要考虑换算算法。中学的时候，我们认识了函数。1. 函数，表示输入与输出之间的映射关系。给定一个输入 x 函数 f f 会生成一个输出 y=f(x)。我们一般常见的函数，也就三四个参数，多至 8，9 个参数。对于我们认识在一张图片上找出人脸的检测任务，简单想一下，不止大小，还有位置，这些函数肯定不能完成。这时候我们就需要更多更复杂的函数了。我们就考虑使用神经网络。神经网络可以看作是一个复杂的函数，能够逼近任何连续函数（通用逼近定理）。通过组合多个简单函数（神经元），神经网络能够学习复杂的非线性映射。人脸检测 yolov8face 的参数量，可能达到几百万个。我们需要大量数据进行训练，保存参数的系数，Pytorch 保存格式为.pth 文件。ONNX(开放神经网络交换) 格式，是一个用于表示深度学习模型的标准，可使模型在不同框架之间进行转移。对于人脸检测网络 yolofac$e_8$$n.onnx$　　$Netron$　$onnx$

图 2.1: Title of picture

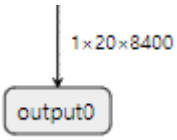


图 2.2: Title of picture

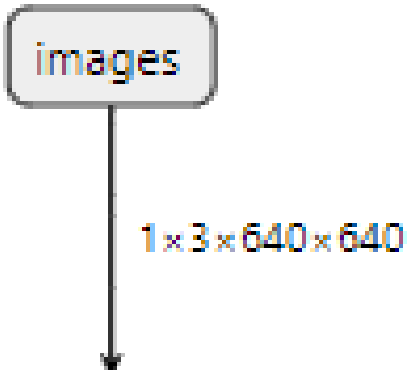
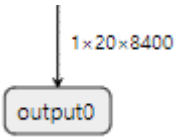


图 2.3: ”输入”

输出:



在换脸的应用中，我会使用多个深度学习的网络，像是人脸关键点检测网络，换脸网络及人脸美化网络。他们有着类似的网络结果，所以，提取抽象类，封装为一个接口。它通过定义纯虚函数强制派生类实现特定功能。

```
template <typename T>
class IEngine {
public:
    virtual ~IEngine() = default;
        const std::array<float, 3> &divVals = {1.f, 1.f, 1.f},
        bool normalize = true, int method = 0) = 0;
    virtual bool loadNetwork(
        std::string trtModelPath,
        const std::array<float, 3> &subVals = {0.f, 0.f, 0.f},
        const std::array<float, 3> &divVals = {1.f, 1.f, 1.f},
        bool normalize = true) = 0;
    virtual bool runInference(
        const std::vector<std::vector<cv::cuda::GpuMat>> &inputs,
        std::vector<std::vector<std::vector<T>>> &featureVectors) = 0;
    virtual const std::vector<nvinfer1::Dims3> &getInputDims() const = 0;
    virtual const std::vector<nvinfer1::Dims> &getOutputDims() const = 0;
};
```

定义 Engine 类实例化这个接口，实现接口定义的功能。

```
template <typename T>
class Engine : public IEngine<T> {
public:
    Engine(const Options &options);
    ~Engine();

    // Build the onnx model into a TensorRT engine file, cache the model to disk
    // (to avoid rebuilding in future), and then load the model into memory The
    // default implementation will normalize values between [0.f, 1.f] Setting the
    // normalize flag to false will leave values between [0.f, 255.f] (some
    // converted models may require this). If the model requires values to be
    // normalized between [-1.f, 1.f], use the following params:
    //     subVals = {0.5f, 0.5f, 0.5f};
    //     divVals = {0.5f, 0.5f, 0.5f};
    //     normalize = true;
    bool buildLoadNetwork(std::string onnxModelPath, const std::array<float, 3> &sub
                          const std::array<float, 3> &divVals = {1.f, 1.f, 1.f}, boo

    // Load a TensorRT engine file from disk into memory
    // The default implementation will normalize values between [0.f, 1.f]
    // Setting the normalize flag to false will leave values between [0.f, 255.f]
    // (some converted models may require this). If the model requires values to
    // be normalized between [-1.f, 1.f], use the following params:
    //     subVals = {0.5f, 0.5f, 0.5f};
    //     divVals = {0.5f, 0.5f, 0.5f};
    //     normalize = true;
    bool loadNetwork(std::string trtModelPath, const std::array<float, 3> &subVals =
                     const std::array<float, 3> &divVals = {1.f, 1.f, 1.f}, bool nor

    // Run inference.
    // Input format [input][batch][cv::cuda::GpuMat]
    // Output format [batch][output][feature_vector]
    bool runInference(const std::vector<std::vector<cv::cuda::GpuMat>> &inputs, std:

    // Utility method for resizing an image while maintaining the aspect ratio by
    // adding padding to smaller dimension after scaling While letterbox padding
    // normally adds padding to top & bottom, or left & right sides, this
```

```
        // implementation only adds padding to the right or bottom side This is do
        // so that it's easier to convert detected coordinates (ex. YOLO model) ba
        // to the original reference frame.
        static cv::cuda::GpuMat resizeKeepAspectRatioPadRightBottom(const cv::cuda
                                                                    const cv::Scal

        [[nodiscard]] const std::vector<nvinfer1::Dims3> &getInputDims() const ove
        [[nodiscard]] const std::vector<nvinfer1::Dims> &getOutputDims() const ove

        // Utility method for transforming triple nested output array into 2D arra
        // Should be used when the output batch size is 1, but there are multiple
        // output feature vectors
        static void transformOutput(std::vector<std::vector<std::vector<T>>> &input

        // Utility method for transforming triple nested output array into single
        // array Should be used when the output batch size is 1, and there is only
        // single output feature vector
        static void transformOutput(std::vector<std::vector<std::vector<T>>> &input
        // Convert NHWC to NCHW and apply scaling and mean subtraction
        static cv::cuda::GpuMat blobFromGpuMats(const std::vector<cv::cuda::GpuMat>
                                                const std::array<float, 3> &divVals

    //private:
        // Build the network
        bool build(std::string onnxModelPath, const std::array<float, 3> &subVals,
        bool build1(std::string onnxModelPath, const std::array<float, 3> &subVals

        bool constructNetwork(std::string onnxModelPath, std::unique_ptr<nvinfer1::
            std::unique_ptr<nvinfer1::INetworkDefinition>& network, std::unique_ptr
            std::unique_ptr<nvonnxparser::IParser>& parser);


        // Converts the engine options into a string
        std::string serializeEngineOptions(const Options &options, const std::strin

        void getDeviceNames(std::vector<std::string> &deviceNames);

        void clearGpuBuffers();
```

```
    // Normalization, scaling, and mean subtraction of inputs
    std::array<float, 3> m_subVals{};
    std::array<float, 3> m_divVals{};
    bool m_normalize;

    // Holds pointers to the input and output GPU buffers
    std::vector<void *> m_buffers;
    std::vector<uint32_t> m_outputLengths{};
    std::vector<nvinfer1::Dims3> m_inputDims;
    std::vector<nvinfer1::Dims> m_outputDims;
    std::vector<std::string> m_IOTensorNames;
    int32_t m_inputBatchSize;

    // Must keep IRuntime around for inference, see:
    // https://forums.developer.nvidia.com/t/is-it-safe-to-deallocate-nvinfer1-irunti
    //std::unique_ptr<nvinfer1::IRuntime> m_runtime = nullptr;
    std::shared_ptr<nvinfer1::IRuntime> m_runtime = nullptr;

    std::unique_ptr<Int8EntropyCalibrator2> m_calibrator = nullptr;
    //std::unique_ptr<nvinfer1::ICudaEngine> m_engine = nullptr;
    std::shared_ptr<nvinfer1::ICudaEngine> m_engine = nullptr;
    std::unique_ptr<nvinfer1::IExecutionContext> m_context = nullptr;
    const Options m_options;
    Logger m_logger;
};

template <typename T> Engine<T>::Engine(const Options &options) : m_options(options)

template <typename T> Engine<T>::~Engine() { clearGpuBuffers(); }
```

LaTeX 的源文件本质上是文本文档，利用 Windows 自带文本编辑器、note++、word、vim 等文本编辑器均可编写出 tex 文档，至于 texwork、texstudio、winedt 等则为转述的 tex 编辑器，提供了语法高亮、匹配查找、自动补全命令等等用途。

除此之外 LaTeX 还可以排版数学公式、图片、表格等等，内容将在后续章节件数。

2.1 LaTeX 基本的命令与代码结构

LaTeX 命令均由反斜线 \ 开头，并为下列两种形式填空后续：

- 由反斜线 \ 与一连串字母组成，如\LaTeX。注意在命令后需加空格或其他非字母作分隔符；
- 由反斜线 \ 由后面的非字母符号组成，不需要分隔符，如\%（百分号在 LaTeX 中为注释），为转义意。

注意 LaTeX 命令对**大小写是十分敏感的**,比如输入\LaTeX可以得到错落有致的 LaTeX 而输入\LaTex或者\latex则会报错，不会得到任何内容。

在 LaTeX 中的参数大多在 $\{\cdots\}$ 或是在 $[\cdots]$ 内，如之前所述\documentclass[CJK, GBK, UTF-8, oneside, a4paper, 12pt]ctexart。一些命令会在后面附带 * 号，带 * 号与不带 * 号结果不同。

为使一些状态、效果在局部生效，LaTeX 引入了**环境**的用法，需要局部生效的内容被输入在环境内，由\begin{environment name}{arguments}开始，由\end{environment}结束。其中 *environment* 为环境名称，\begin{environment}与\end{environment}内的环境名应该一致，*arguments* 为可选参数，环境之间允许嵌套使用。

2.2 LaTeX 排版中文

排版中文文章时，与 word 不同，无需关注缩进、标题等等，在 LaTeX 中可以方便快捷的设置。一级标题设置代码为\section{title}大括号内为一级标题的名称，对应的可以书写二级、三级标题，LaTeX 命令分别为\subsection{title}与\subsubsection{title}。书写时，LaTeX 会自动忽略文字中间的空格，在换行时需要多空一行。另外的，LaTeX 中的注释为“%”号。下面给出一个简短的例子。

```
\section{一级标题名称}
这里是第一章的内容
% 空一行代表分段，百分号在 LaTeX 中代表注释
这里是第一章的内容
\subsection{二级标题名称}
这里是 1.1 的内容
```

用户可以将代码放置在本模板中进行尝试，需要注意的是，在 body 文件夹内新建文档并书写完后，需要在主文档中依照给定格式导入新书写的文档。

在 LaTeX 中书写中文，无需注意文章标题的编号，在\section{title}类命令中，自带有计数器，可以为标题自动编号，这使得用户无需关注排版格式，更多的关注在文档内容上。

2.3 常用环境

2.3.1 居中

在 LaTeX 中有两种居中方式：

- `\centering`，在环境内使用，该环境内所有内容居中
- `center` 环境，在环境内的所有内容居中

当使用了`center`环境时候，环境内的所有内容都会被居中显示，且不会首行缩进。如果有特殊需要还可以使用 flushleft 和 flushright 环境，用于居左或者居右。

2.3.2 带有编号的显示方式-列表（悬挂缩进）

在书写实验报告时，经常会遇到需要分条叙述的方式，在一般书写排版中需要整体悬挂缩进。

在 LaTeX 中常用的两种环境，分别是 itemize（无序）环境与 enumerate 环境（有序）两种环境可以互相嵌套使用，使用方法如下：

```
% itemize 环境
\begin{itemize}
\item 第一条内容
\item 第二条内容
\end{itemize}
% enumerate 环境
\begin{enumerate}[aa.]
\item 第一条内容
\item 第二条内容
\end{enumerate}
```

itemize 环境会给每一条内容前加 •，而 enumerate 环境可以自定义，如 (1. 2. 3. 或者是 A. B. C.)，对应的设置方法需要在环境后的参数中写`1.`、`A.`，需要注意的是在给 enumerate 环境添加参数时候需要导入 enumerate 包，否则只会有`1. 2. 3.`的序号，并且无法设置样式。

- 第一条内容
- 第二条内容

aa. 第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容第一条内容

bb. 第二条内容

2.3.3 代码

在实验报告中，多半部分需要加入对应的程序，编写代码的常用环境有 verbatim、minted（需要 python 环境支持）、listings，下面分别叙述。

verbatim 环境使用方法简单，但是不够美观，不能实现语法高亮。

```
#include<stdio.h>
int main(){
 printf("hello world");
 return 0;
}
```

lstlisting 环境需要 listings 包，可以自定义设置，本模板自带 listings 包并且给出了一种代码环境格式供用户参考。

```
#include<stdio.h>
int main(){
        printf("hello world");
        return 0;
}
```

minted 环境需要 minted 包，并且电脑中安装有 python 环境。设置 xelatex 编译为 `xelatex.exe -synctex=1 -shell-escape -interaction=nonstopmode %.tex`，添加了 `-shell-escape`参数，可以胜任大部分计算机语言 (Lingo 除外)，并且自动设置语法高亮。需要注意的是 minted 会将`Tab`替换为下文中的`^^I`，将所有的`Tab`替换为空格即可，更多内容详见 minted 宏包文档，Win+R 键后输入`texdoc minted`即可查看 minted 宏包文档，在 LaTeX 中所有宏包的宏包文档均以上述方式打开。

```
#include<stdio.h>
int main(){
^^Iprintf("hello world");
    return 0; /* 将 Tab 替换为空格 */
}
```

要排版简短的代码或关键字，可使用`\verb` 命令：`\verb<delim>\LaTeX<delim>`。在该命令中所有的`\`都不起作用。

第三章　需要用到的神经网络

在 LaTeX 中排版数学公式需要 *amsmath* 宏包（已经包含在本模板中），对多行公式的排版提供了有力的支持。在实验报告的书写中，主要以 *amsmath* 宏包的内容为主，其余内容不做阐述。

3.1　公式排版基础

在 LaTeX 中书写数学公式时必须带有数学环境，数学环境内可以识别特殊的命令并且字体改变为数学字体，一般数学环境有两种，书写方式如下：

- **行内公式**：行内公式是出现在文字陈述中间的数学公式，需要用双 $ 符号括起来，例如我们知道对于矩阵而言，乘法交换率是不成立的，也就是说
  $\forall A \neq B, \exists A \times B \neq B \times A$(`$A\times B\neq B\times A$`)，书写时除去命令后必须的空格外，其他的空格会被一概忽略，至于如何添加空白会在后面叙述。
- **行间公式**：行间公式是出现在文字陈述段落中间的数学公式，一般需要编号。当行间公式需要编号时可以使用 equation 环境，不需要编号时可以使用简单的方式编写行间公式：`\[myEquation\]`

在数学环境内，不允许有多于的空格与空行，需要强制空格可以使用命令`\,`、`\quad`、`\qquad`等，他们的产生的空白距离有所不同。其次数学环境中的所有字母都会被当做变量处理，采用数学字体显示，当需要在数学环境中输入公式时，可以使用命令`\text{}`。

3.2　排版数学公式

在以往的实验报告中，数学公式都会使用 word 中的 mathtype 书写，在较高的版本中可以复制 mathtype 为 LaTeX 代码，但这种投机取巧的方法写出来的符号会非常的丑，并且速度不会比直接使用 LaTeX 书写快多少。下面，作者会首先叙述一部分必要的知识，其次的内容会以实例的方式展现，需与 tex 文档（源文档）配合学习。

1. 上标的表示方式为`a^{2}`，显示结果为 a^2，当上标内容单一时可以省略大括号，如`a^2`也可以显示为 a^2，当需要输入符号^ 时输入`\^`即可。

2. 下表的表示方式为`a_{2}`，显示结果为 a_2，当小标内容单一时也可以省略大括号，当需要输入符号 _ 时输入`\_`即可。
3. 同时需要上下标时书写没有先后顺序，`a^{x+y}_{x_1}`与`a_{x_1}^{x+y}`结果都是 $a_{x_1}^{x+y}$。
4. 对于巨运算符,如果直接书写`\sum^{n}_{i=1}n!`会容易显示为 $\sum_{i=1}^{n} n!$,而添加命令`\limits`后，`\sum\limits^{n}_{i=1}n!`则会显示为 $\sum\limits_{i=1}^{n} n!$。
5. 书写分数的命令为`\frac{text}{den}`，其中 text 为分子部分，den 部分为分母部分，如`\frac{1}{2}`会显示为 $\frac{1}{2}$,如果觉得分数略小可以适当的使用命令`\dfrac{text}{den}`显示为 $\dfrac{1}{2}$。
6. 导数直接使用单引号即可`f' f'' f'''`显示为 $f'f''f'''$，常见的运算符号与巨运算符如`+ - \times \div = \sum \prod \int`，分别为 $+ - \times \div = \sum \prod \int$，更多的基础符号见 amsmath 宏包或 lshort，下面也给出了支持所有的符号大全与手写符号识别的网址。

   LaTeX 支持的符号大全：`http://mirrors.ctan.org/info/symbols/comprehensive/symbols-a4.pdf`

   手写符号识别：`http://detexify.kirelabs.org/classify.html`
7. 需要输入大括号时需输入`\{\}`。

下面会用一些函数、习题、定理、证明过程或是计算过程作为实例：

Stolz 定理：设 $\{y_n\}$ 是严格单调增加的正无穷大量，且

$$\lim_{n\to\infty}\frac{x_n-x_{n-1}}{y_n-y_{n-1}}=a \quad (a\text{可以为有限量，}+\infty\text{与}-\infty),$$

则

$$\lim_{n\to\infty}\frac{x_n}{y_n}=a。$$

求极限

$$\lim_{n\to\infty}\frac{1^k+2^k+\cdots+n^k}{n^{k+1}}(k\text{为正整数})。$$

由已知，可得

$$\lim_{n\to\infty}\frac{a^n}{n!}=0。 \tag{3.1}$$

设函数

$$f(x)=(\frac{x+\exp^{\frac{1}{x}}}{1+\exp^{\frac{4}{x}}}+\frac{\sin x}{|x|}),$$

问当 $x\to 0$ 时，$f(x)$ 的极限是否存在？

设 $f(x)$ 在 $[a,b]$ 上连续，且 $f(x)>0$，证明

$$\frac{1}{b-a}\int_a^b \ln f(x)\mathrm{d}x \leqslant \ln\left(\frac{1}{b-a}\int_a^b f(x)\mathrm{d}x\right)。 \tag{3.2}$$

常用的希腊字符如下：

$$\alpha\beta\gamma\delta\varepsilon\zeta\theta\eta\mu\xi\pi\sigma\omega\phi$$

3.3 多行公式的排版

在书写报告时，时长会遇到多行排版，如矩阵、分段函数等等，在下文将介绍部分常用的环境，用于排版多行公式。

多行排版的环境使用方式大致类似，需要对齐的位置利用 & 分割，行末需要使用\\分割。

3.3.1 align 环境

align 环境会给环境内的每一行公式编号，去掉编号可以使用\notag，使用方式如下：

```
\begin{align}
a & = b + c \\
a & = b + c \\
x + y & = d + e \notag
\end{align}
```

显示为

$$a = b + c \tag{3.3}$$

$$a = b + c \tag{3.4}$$

$$x + y = d + e$$

align 环境会在 & 符号处对齐，多个 & 会分段对齐。

3.3.2 aligned 环境

与 align 环境不同，aligned 环境会给公式整体一个编号，而不是每一行都有编号，同时需要 equation 环境套在外面。

```
\begin{equation}
^^I\begin{aligned}
^^I^^Ia & = b + c \\
^^I^^Ix + y & = d + e
^^I\end{aligned}
\end{equation}
```

显示为

$$\begin{aligned} a &= b + c \\ x + y &= d + e \end{aligned} \tag{3.5}$$

split 环境和 aligned 环境用法类似，也用于和 equation 环境套用，区别是 split 只能将每行的一个公式分两栏，aligned 允许每行多个公式多栏。

3.3.3 array 环境

array 环境用于排版数学数组、矩阵等，数组可作为一个公式块，在外套用\left、\right等定界符。跟在环境名后的{cccc}意为矩阵 4 列均居中（c 代表居中、l 代表左对齐、r 代表右对齐，更仔细的将在表格排版处讲述），具体实例如下：

```
\[
\mathbf{X} = \left(
\begin{array}{cccc}
x_{11} & x_{12} & \ldots & x_{1n}\\
x_{21} & x_{22} & \ldots & x_{2n}\\
\vdots & \vdots & \ddots & \vdots\\
x_{n1} & x_{n2} & \ldots & x_{nn}\\
\end{array} \right)
\]
```

显示为

$$\mathbf{X} = \left(\begin{array}{cccc} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{array} \right)$$

其中，\left(、\right)就是矩阵的定界符，小括号可以替换为[{等。

3.3.4 case 环境

借用之前所述的定界符，使用单侧定界符可以用来书写分段函数，如 **Riemann** 函数可以书写为：

```
\[
\mathbf{R}(x)=\left\{\begin{array}{ll}
\frac{1}{p},&x=\frac{q}{p}(p\in \mathbf{N}^{+},q\in \mathbf{Z}-\{0\},p,q\text{互质}),\\
1,&x=0,\\
0,&x\text{为无理数}
\end{array}\right.
\]
```

显示为

$$\mathbf{R}(x) = \begin{cases} \frac{1}{p}, & x = \frac{q}{p}(p \in \mathbf{N}^{+}, q \in \mathbf{Z} - \{0\}, p, q\text{互质}), \\ 1, & x = 0, \\ 0, & x\text{为无理数} \end{cases}$$

对于这类分段函数，可以使用更简单的 cases 环境来完成

```
\[
\mathbf{R}(x)=\begin{cases}
\frac{1}{p},&x=\frac{q}{p}(p\in \mathbf{N}^{+},q\in \mathbf{Z}-\{0\},p,q\text{互质}),\\
1,&x=0,\\
0,&x\text{为无理数}
\end{cases}
\]
```

显示为

$$\mathbf{R}(x)=\begin{cases}\frac{1}{p}, & x=\frac{q}{p}(p\in \mathbf{N}^{+},q\in \mathbf{Z}-\{0\},p,q\text{互质}),\\ 1, & x=0,\\ 0, & x\text{为无理数}\end{cases}$$

第四章　利用 LaTeX 排版图片与表格

在介绍如何排版表格与图片时，先介绍浮动体的概念

4.1　浮动体

在排版中文文档或者实验报告时，尤其是在今后的论文、书籍撰写中，表格与图片均称为**浮动体**。顾名思义，浮动体在文中的位置不是固定的，美观起见需要自动放在合适的位置，在需要的时候做引用。在排版时，作者需要优先排版文字内容，最后再关注图片位置，不应固定死图片的位置，除了造成大片空白也会使得整体不够美观。

4.1.1　浮动体的用法

一般来说浮动体环境有两种，figure 环境与 table 环境，分别用于浮动图片与表格，用法如下：

```
\begin{figure}[<placement>]
content...
\end{figure}
```

表格与图片用法相同，跟在环境名后面的 *placement* 提供了浮动体在页面中允许排版的位置，默认为 tbp，意为允许在顶部、底部、单独成页排版。

h	代码所处的当前位置
t	页面顶端
b	页码底部
p	单独成页
!	在决定位置时忽略限制

4.1.2 浮动体的标题

在浮动体中，利用\caption{...}添加标题，用法与\chapter{...}类似，添加的标题会自动编号，figure 会在内容前显示如“图 1”的样式，表格类似。

紧跟着\caption{...}后面可以添加\label{key}命令交叉引用，具体在后面章节叙述。图片的排版 LaTeX 本身不支持插图功能，需要由 graphicx 宏包（本模板已添加）辅助支持。在本模板下，可以添加.jpg.pdf.eps.png.bmp 格式的图片，

在调用了 graphicx 包后,可以使用命令\includegraphics[options]{ filename }插入图片,*filename* 是图片的位置,本模板中需要将图片放在 figure 文件中,并使用相对路径调用,如figure/filename.png。*options* 是需要的参数，如设置图片宽为 0.7*cm* 需要在该位置书写[width=0.7cm]，具体的参数见表下表。

参数	含义
width=h	将图片缩放到宽度为 h
hight=h	将图片缩放到高度为 h
scale=h	将图片按照原尺寸缩放 h 倍
angle=h	令图片逆时针旋转 h 度

```
\begin{figure}[h]
\centering
\includegraphics[scale = 0.7]{example-image-A}
\caption{导入的图片}
\label{fig1}
\end{figure}
```

需要排版并排子图推荐使用 subfig 宏包，具体使用请看宏包文档表格的排版 在实验报告或者论文中，表格是比不可少的部分。下面给出一个简单的表格排版实例：

```
\begin{table}
\centering
\caption{表格排版实例}
\label{tab1}
\begin{tabular}{|c|l|r|}
\hline
AAA&B&CCC\\
\hline
A&BBB&C\\
\hline
\end{tabular}
```

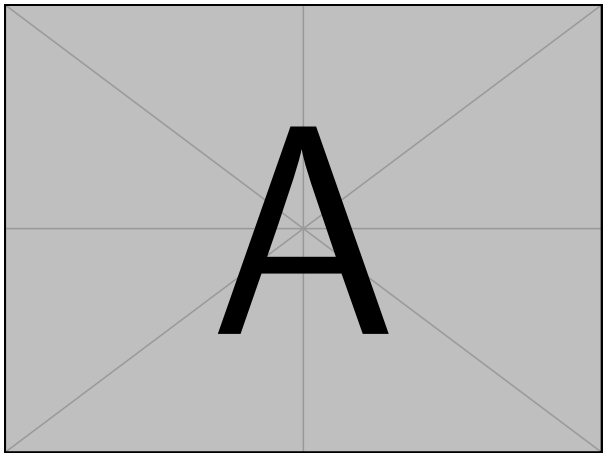


图 4.1: 导入的图片

```
\end{table}
```

显示为表 [4.1]。与多行公式类似，表格排版中的列由 tabular 环境后的参数决定，c、l、r 分别代表

表 4.1: 表格排版实例

AAA	B	CCC
A	BBB	C

居中、居左、居右对齐，必须与列数相同，参数之间的竖线代表是否在表格中绘制竖线。行之间需要添加横线需要命令`\hline`，如果需要合并单元格或者其他操作，具体见 lshort 表格排版章节，这里只讲述简单的部分。三线表的绘制只需要将参数中的竖线去除即可。

对于初次使用者而言，表格排版是一个很大的难题，在 excel 中有一个类似的插件可以快捷的生成大致的表格，在打开 excel 加载项后，下载插件`excel2latex`即可。在生成大致表格后进行细微的调整，可以快速的绘制出想要的表格。

第五章　交叉引用与其他

5.1　交叉引用

在上一章我们强调了表格与图片需要在页面中自动寻找合适的位置，而不是固定在某个位置，那么我们需要引用表格内容时就不能用“如下表”类似的话语，而需要用到交叉引用，也就是“如图 1”类似的文字。

细心的读者可以发现在浮动体环境内作者均加有`\label{key}`的语句，这是为了后文方便引用图片内容而写，每个图片、表格的 *key* 值必须唯一。在做引用的时候需要用到命令`\ref{label}`，*label* 处填写唯一的 *key* 值，这样就可以做到交叉引用，更方便的是在 pdf 阅读时，可以通过单击索引小标来定位到该图片处，如这里引用之前的图片就可以写如图 [4.1]。

除了浮动体可以做交叉引用，公式也是可以做交叉引用的，这里引用文章出现的第一个公式可以写如公式 [3.1]。

需要注意的是，目录与交叉引用需要至少编译两次，也就是点两次编译按钮。

由于本科实验报告中不要求添加参考文献，这里不做多于的叙述，详细的在 lshort 中也有叙述。

5.2　其他

这里将会叙述一些其他的知识，有的是在作者平时写作中遇到的，有的是模板写作中的问题，本章节会即时更新。

按照学校模板要求，在封面字体需为仿宋 GB2312 加粗，事实上，字体的加粗并不想 word 中强制加粗（会出现很多很多很多错误），而是用其他粗体的字体来代替，因此本文由于版权限制，使用黑体代替格式中要求的字体。

在使用 matlab 绘图时，可以在图例、坐标轴等地方利用 LaTeX 公式写出分式等符号，但是 matlab 中只允许一部分 LaTeX 代码，并不是全部的数学公式代码都可以用。

LaTeX 有相当多的宏包用于不同的环境，神经网络、化学、生物等等学科的图都可以在宏包中找到。

第六章　致谢

感谢成都理工大学的 LaTeX 模板给予了笔者 LaTeX 排版的启蒙。

学生实习心得	由于 LaTeX 不适用绘制工作用表，所以附件处的体验心得通过特殊方式得到，在所给的 word 文档中书写完毕后另存为 pdf 版本，或是由 pdf 打印机得到 pdf 版本，放置在本文件夹内即可。切记请勿超过一页，同时需要保证页边距为 0，以本文档为准。 学生（签名）： 年 月 日
诚信承诺	本人郑重声明所呈交的实习报告是本人在指导教师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注的地方外，报告中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同学对本文研究所做的贡献均已在报告中作了明确的说明并表示谢意。 学生（签名）：
指导教师评语	 成绩评定： 指导教师（签名）： 年 月 日