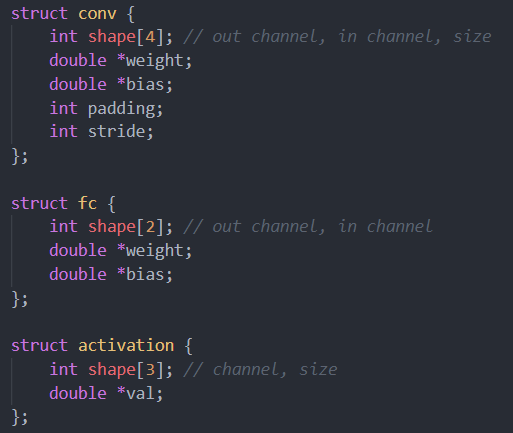
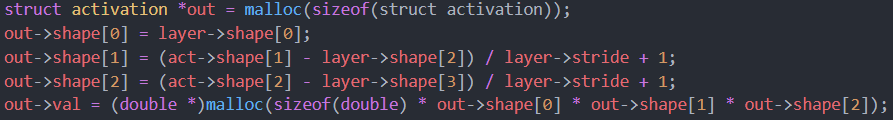
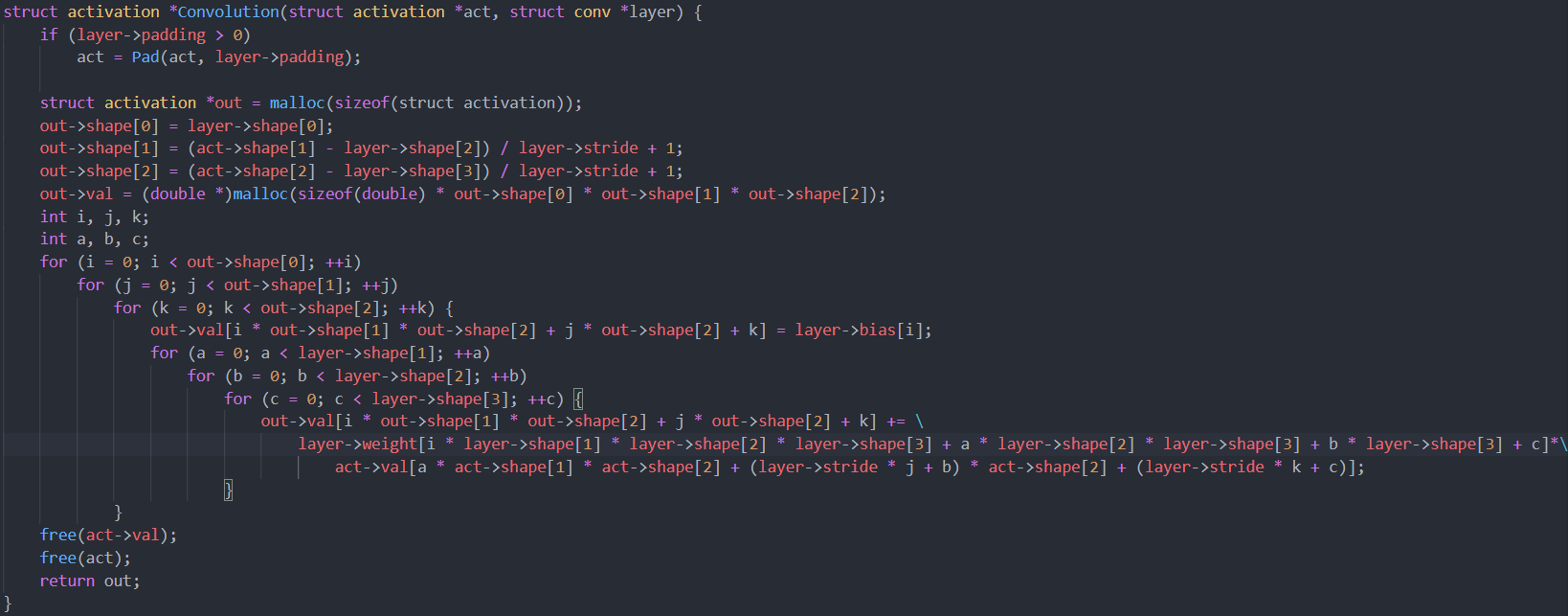
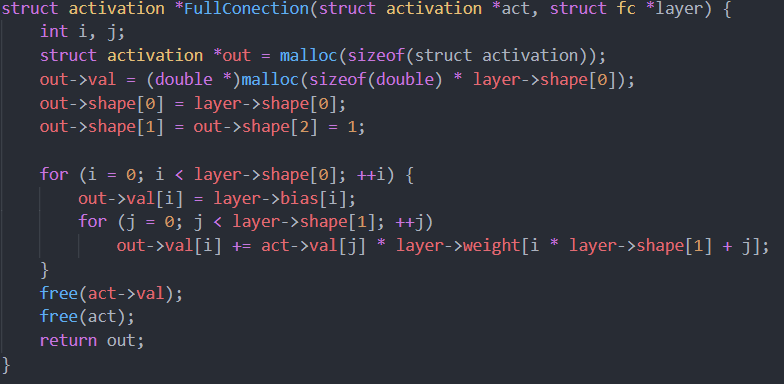
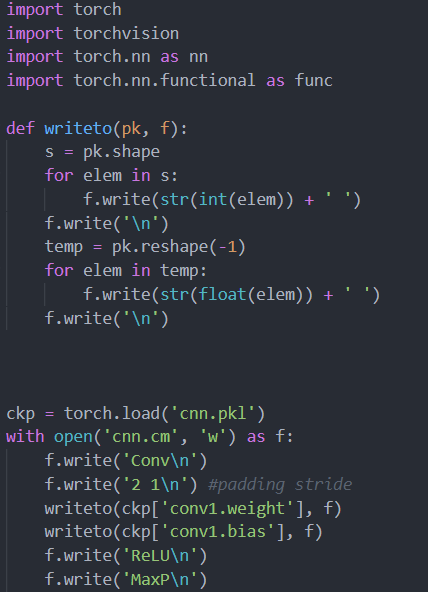
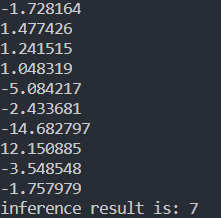
**基于C语言的CNN Inference框架**

朱昱 2017011230

1. 简介

随着深度学习和神经网络的发展，CNN被证明是一种准确且高效的模型来处理图像

问题。Python等语言中已经有了很多很成熟的CNN实现框架，但是这些框架大多是高度封装的，不利于我们学习和理解CNN内部的计算过程。本次实验将使用C语言实现一个简单的CNN Inference框架，用以加深对CNN计算的理解，为今后在硬件上部署做准备。

1. 实现内容
   1. 数据结构定义  
        
      每种数据结构都包含了它的维度参数，指向数据块的指针，以及可能的必要的参数。其中维度参数的顺序与pytorch中的定义相同，比如conv的第一维是output channel，第二维是input channel，之后的两维是卷积核的宽，高。值得一提的是，当前版本的输入图片暂未考虑batch，或者说batch size固定为1，这是因为当前版本只是实现了一个可视化的demo，考虑batch与否并不会对代码结构产生本质变化。  
      在如上数据结构的定义下，我们采用malloc的方式申请内存，并使用数据结构自带的维度参数来计算要访问的数据的相对内存位置。对内存的申请和访问代码示例如下：  
      申请内存:  
        
      访问下标[I, j, k]：  
        
      回收内存：  
      
   2. NN计算模块的实现：  
      卷积层：公式为  
      全连接层：公式为  
        
      值得一提的是，在卷积层和全连接层中，对于输出数据的内存访问都是连续的，这样做有助于减少访存miss和访存时间。此外还实现了Padding，ReLU和MaxPooling，由于原理较为简单，在此不做赘述。
   3. Pytorch接口  
      主要分两个部分，一是从Pytorch中加载模型，二是从Pytorch中加载输入数据。接口部分的主要逻辑是用Python把数据转化成字符形式的文件，方便使用C读入。  
      
   4. 可视化  
      由于使用的是C语言和wsl环境，这里采用字符画来实现可视化。对于灰度在0~0.33的像素采用空格，0.33~0.67的采用o，0.67~1的采用@，效果如下：  
      
2. 结果验证  
   对于同一张图，C代码的分类分数结果：  
     
   pytorch的分类分数结果：  
     
   可以看到数字是一样的，可以验证代码实现的正确性。
3. 实验总结  
   通过本次实验，我实现了基于C语言的CNN Inference框架。在框架中包含了CNN的各种层以及操作，并且用尽可能连续的访存方式实现了这些操作。
4. 文件列表  
   github：<https://github.com/Hazuyuki/CUNIX/tree/master/CNNFramework>  
   operators.h: CNN的操作函数与数据结构定义  
   inference.c: 需求网络模型文件cnn.cm和数据文件cdata.dat，运行一个网络模型  
   trans\_weight.py: 从pkl文件生成模型文件cnn.cm  
   gene\_data.py: 从MNIST数据文件中读取数据生成模型文件cdata.dat  
   train.py: 训练一个CNN网络，参数保存到cnn.pkl