HandwrittenDigitRecogDemo

手写数字识别AI模型在安卓端的调用示例

开发环境

> OS:Windows 10
>
> IDE: Android Studio 3.2
>
> Java(JDK) 1.8
>
> Android SDK 28
>
> Android NDK 19
>
> Cmake 3.6

AI模型部署工具

> NCNN (https://github.com/Tencent/ncnn)

AI模型部署工具编译环境

> Ubuntu 18.04
>
> python 3.6.9
>
> cmake 3.10.2
>
> Android NDK(android-ndk-r19c)
>
> libopencv-dev
>
> protobuf 3.5.1 (https://github.com/protocolbuffers/protobuf)

AI模型编译及部署

模型转换

pytorch转onnx

'''python

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
input = torch.randn(1, 1, 28, 28, device=device)
```

```
model = {Your_Net_Model_Class_Name}().to(device)
net.load_state_dict(torch.load(input_pytorch_model_path))
net.eval()
input_names = ['data']
output_names = ['prob']
torch.onnx._export(model, input, output_onnx_model_path,
                   export_params=True, verbose=True,
                   input_names=input_names, output_names=output_names)
```

'''

精简化onnx

'''

```
sudo pip3 install onnx-simplifier
python3 -m onnxsim ${Your_Onnx_Model_Name}.onnx ${Your_Onnx_Sim_Model_Name}.onnx
```

'''

编译安装protobuf

'''

```
tar -zxvf protobuf3.5.1.tar.gz
cd protobuf-3.5.1/
./autogen.sh
./configure
make -j4
sudo make install
sudo ldconfig
protoc --version
```

'''

编译ncnn相关转换工具

'''

```
cd ${Your_Path}/ncnn/
mkdir -p build
cd build
cmake ..
make -j4
```

onnx转ncnn

'''

```
cd ${ncnn_path}/nuild/tools/onnx/
cp ${your_onnx_file_path} ./
./onnx2ncnn ${your_onnx_file} ${your_ncnn_param_file_name}.param
${your_ncnn_bin_file_name}.bin
```

'''

执行完以上命令，会得到 *.param和*.bin两个文件，可直接用于安卓应用中部署

'''

```
./ncnn2mem ${your_ncnn_param_file_name}.param ${your_ncnn_bin_file_name}.bin
${your_ncnn_file_name}.id.h ${your_ncnn_file_name}.mem.h
```

'''

执行完以上命令，会得到 *.param.bin*、.bin、*.id.h*、.mem.h四个文件，可用于安卓应用中加密部署（该方式无法通过反编译窥探网络模型相关信息）

对于加密方式调用：

拷贝*.param.bin*、.bin两个文件到安卓应用工程中的asset文件夹下

拷贝*.id.h到安卓应用工程中的cpp/include文件夹下

安卓端ncnn调用库编译

编译相关环境配置

> 减少编译库所在内存空间

“ ‘

```
# Edit $ANDROID_NDK/build/cmake/android.toolchain.cmake with your favorite editor
# remove "-g" line
list(APPEND ANDROID_COMPILER_FLAGS
 -g
 -DANDROID
```

’ “

> 配置Vulkan（GPU加速）环境

'''

```
$ wget https://sdk.lunarg.com/sdk/download/1.1.114.0/linux/vulkansdk-linux-x86_64-
1.1.114.0.tar.gz?Human=true -O vulkansdk-linux-x86_64-1.1.114.0.tar.gz
$ tar -xf vulkansdk-linux-x86_64-1.1.114.0.tar.gz
$ export VULKAN_SDK=`pwd`/1.1.114.0/x86_64
```

'''

32位 armV7 cpu

'''

```
cd {ncnn_path}/
mkdir -p build-android-armv7
cd build-android-armv7/
export ANDROID_NDK=${Your_ndk_dir_path}
cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake -
DANDROID_ABI="armeabi-v7a" -DANDROID_ARM_NEON=ON -DANDROID_PLATFORM=android-14
..
make -j4
make install
```

'''

32位 armv7 gpu vulkan

> 修改CMakeLists.txt
>
> option(NCNN_VULKAN "vulkan compute support" ON)

'''

```
mkdir -p build-android-armv7-vk
cd build-android-armv7-vk/
export ANDROID_NDK=${Your_ndk_dir_path}
export VULKAN_SDK=${Your_vulkan_sdk_dir_path}
cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake -
DANDROID_ABI="armeabi-v7a" -DANDROID_ARM_NEON=ON -DANDROID_PLATFORM=android-24
-DNCNN_VULKAN=ON ..
make -j4
make install
```

'''

64位 armv8 cpu

'''

```
mkdir -p build-android-aarch64
cd build-android-aarch64/
export ANDROID_NDK=${Your_ndk_dir_path}
cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake -
DANDROID_ABI="arm64-v8a" -DANDROID_PLATFORM=android-21 ..
make -j4
make install
```

'''

64位 armv8 gpu vulkan

> 修改CMakeLists.txt
>
> option(NCNN_VULKAN "vulkan compute support" ON)

'''

```
mkdir -p build-android-armv8-vk
cd build-android-armv8-vk/
export ANDROID_NDK=${Your_ndk_dir_path}
export VULKAN_SDK=${Your_vulkan_sdk_dir_path}
cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake -
DANDROID_ABI="arm64-v8a" -DANDROID_ARM_NEON=ON -DANDROID_PLATFORM=android-24 -
DNCNN_VULKAN=ON ..
make -j4
make install
```

'''

安卓端ncnn库调用

拷贝上一步编译生成的库文件和头文件到对应安卓工程文件夹中

'''

```
cd ${ncnn_path}/${ncnn_android_build_path}/install
cp -r include/ncnn ${Your_android_project_cpp_dir}/include/
cp lib/libncnn.a ${Your_android_project_jniLibs_dir}/${ANDROID_ABI}/
```

'''

配置安卓端app/build.gradle中ndk编译相关

'''gradle

```
android {
  defaultConfig {
    ndk {
        abiFilters "armeabi-v7a"
        stl "gnustl_static"
    }
    externalNativeBuild {
      cmake {
        arguments "-DANDROID_TOOLCHAIN=clang"
        cFlags "-fopenmp -O2 -fvisibility=hidden -fomit-frame-pointer -fstrict-aliasing -ffunction-
sections -fdata-sections -ffast-math "
        cppFlags "-fopenmp -O2 -fvisibility=hidden -fvisibility-inlines-hidden -fomit-frame-pointer
```

```
    -fstrict-aliasing -ffunction-sections -fdata-sections -ffast-math "
                arguments "-DANDROID_STL=c++_shared", "-DANDROID_CPP_FEATURES=rtti exceptions"
                cppFlags ""
                cppFlags "-std=c++11"
                cppFlags "-frtti"
                cppFlags "-fexceptions"
            }
        }
    }
    externalNativeBuild {
        cmake {
            path "CMakeLists.txt"
        }
    }
}
```

'''

配置CMakeLists.txt

'''txt

```
添加ncnn库
add_library(libncnn STATIC IMPORTED )
set_target_properties(
                libncnn
                PROPERTIES IMPORTED_LOCATION
                ${CMAKE_SOURCE_DIR}/src/main/jniLibs/${ANDROID_ABI}/libncnn.a
                )
#添加工程所依赖的库
find_library(log-lib log android)

target_link_libraries( HwDr
                  libncnn
                  jnigraphics
                  z
                  ${log-lib}
                  android )
```

'''

jni调用代码编写

'''cpp

```
//模型加载不分关键代码
ncnn::Option opt;
opt.blob_allocator = &g_blob_pool_allocator;
opt.workspace_allocator = &g_workspace_pool_allocator;
DigitRecognet.opt = opt;
// init param
```

```
{
    int retp = DigitRecognet.load_param_bin(mgr, "Mnist/models/LeNet_p27_sim.param.bin");
    if (retp != 0)
    {
        LOGE("load_param_bin failed");
    }
}
// init bin
{
    int retm = DigitRecognet.load_model(mgr, "Mnist/models/LeNet_p27_sim.bin");
    if (retm != 0)
    {
        LOGE("load_model_bin failed");
    }
}

//模型推理部分关键代码
ncnn::Extractor ex = DigitRecognet.create_extractor();
ex.set_num_threads(threadnum);
ex.set_light_mode(true);
ex.input(LeNet_p27_sim_param_id::BLOB_data, img_);
ncnn::Mat out;
ex.extract(LeNet_p27_sim_param_id::BLOB_prob, out);
//输出数据概率化
{
    ncnn::Layer* softmax = ncnn::create_layer("Softmax");
    ncnn::ParamDict pd;
    softmax->load_param(pd);
    softmax->forward_inplace(out, DigitRecognet.opt);
    delete softmax;
}
out = out.reshape(out.w * out.h * out.c);
unsigned int out_size = (unsigned int)(out.w);
feature_out.resize(out_size);
//赋值输出数据
for (int j = 0; j < out.w; j++)
{
    feature_out[j] = out[j];
}

//jni数据输入与结果输出部分关键代码
//字节数组预处理
jbyte *digitImgData = env->GetByteArrayElements(digitImgData_, NULL);
unsigned char *digitImgCharData = (unsigned char *) digitImgData;
//转换图片数据格式
ncnn::Mat ncnn_img = ncnn::Mat::from_pixels_resize(digitImgCharData,
ncnn::Mat::PIXEL_RGBA2GRAY, w, h, 28, 28);
//输入数据归一化
const float norm_vals[3] = {1/255.f, 1/255.f, 1/255.f};
ncnn_img.substract_mean_normalize(0, norm_vals);
std::vector<float> feature;
//数字手写识别推理
```

```
mDigitRecog->start(ncnn_img, feature);
//提取并赋值概率数组
float *featureInfo = new float[10];
for(int i = 0;i<10;i++){
  featureInfo[i] = feature[i];
}
```

'''

java调用代码编写

'''java

```
//native方法类部分关键代码
//jni编译so库加载
static {
  System.loadLibrary("HwDr");
}
//模型初始化
public native boolean MnistAssetModelInit(AssetManager amgr);
//模型反初始化
public native boolean MnistModelUnInit();
//模型推理
public native float[] HwDigitRecog(byte[] digitImgData, int w, int h);
//加密方式初始化模型
boolean init = false;
public HwDr(AssetManager assetManager){
  init = MnistAssetModelInit(assetManager);
  if(init) {
    Log.i(TAG, "模型初始化成功");
  }
}

//Bitmap(手写数字黑底白字图像)转Byte[]
int bytes = image.getByteCount();
ByteBuffer buffer = ByteBuffer.allocate(bytes);
image.copyPixelsToBuffer(buffer);
byte[] byteTemp = buffer.array();
```

'''

安卓端ncnn调用相关数据预处理

> 模型初始化
>
> 正常方式

'''

```
// init param
{
  int retp = DigitRecognet.load_param(mgr, "Mnist/models/LeNet_p27_sim.param");
  if (retp != 0)
  {
    LOGE("load_param failed");
  }
}
// init bin
{
  int retm = DigitRecognet.load_model(mgr, "Mnist/models/LeNet_p27_sim.bin");
  if (retm != 0)
  {
    LOGE("load_model_bin failed");
  }
}
```

'''

> 加密方式

'''

```
// init param bin
{
  int retp = DigitRecognet.load_param_bin(mgr, "Mnist/models/LeNet_p27_sim.param.bin");
  if (retp != 0)
  {
    LOGE("load_param_bin failed");
  }
}
// init bin
{
  int retm = DigitRecognet.load_model(mgr, "Mnist/models/LeNet_p27_sim.bin");
  if (retm != 0)
  {
    LOGE("load_model_bin failed");
  }
}
```

'''

> 输入模型图像数据预处理
>
> 字节数组输入

'''

```
//java部分关键代码
//Bitmap转byte[]
int bytes = bitmap.getByteCount();
ByteBuffer buffer = ByteBuffer.allocate(bytes);
image.copyPixelsToBuffer(buffer);
byte[] byteImg = buffer.array();
//字节数组预处理
jbyte *digitImgData = env->GetByteArrayElements(digitImgData_, NULL);
unsigned char *digitImgCharData = (unsigned char *) digitImgData;
//转换图片数据格式
ncnn::Mat ncnn_img = ncnn::Mat::from_pixels_resize(digitImgCharData,
ncnn::Mat::PIXEL_RGBA2GRAY, w, h, 28, 28);
//输入数据归一化
const float norm_vals[1] = {1/255.f};
ncnn_img.substract_mean_normalize(0, norm_vals);
```

'''

位图输入

'''

```
//jni部分关键代码
cv::Mat matBitmap;
bool ret = BitmapToMatrix(env, digitImgBitmap, matBitmap);
if(!ret){
  return NULL;
}
//转换图片数据格式
ncnn::Mat ncnn_img = ncnn::Mat::from_pixels_resize(matBitmap.data,
ncnn::Mat::PIXEL_BGRA2GRAY, w, h, 28, 28);
//输入数据归一化
const float norm_vals[1] = {1/255.f};
ncnn_img.substract_mean_normalize(0, norm_vals);
```

'''

路径输入

'''

```
//jni部分关键代码
const char *digitImgPath = env->GetStringUTFChars(imgPath, 0);
std::string imgPath_ = digitImgPath;
cv::Mat digitImageMat = cv::imread(imgPath_);
//转换图片数据格式
ncnn::Mat ncnn_img = ncnn::Mat::from_pixels_resize(digitImageMat.data,
ncnn::Mat::PIXEL_BGR2GRAY, digitImageMat.cols, digitImageMat.rows, 28, 28);
```

```
//输入数据归一化
const float norm_vals[1] = {1/255.f};
ncnn_img.substract_mean_normalize(0, norm_vals);
```

'''

安卓端性能测试

> 修改${ncnn_path}/benchmark/benchncnn.cpp

'''cpp

```
// benchmark测试部分关键代码
// cpu热身
for (int i = 0; i < g_warmup_loop_count; i++)
{
    ncnn::Extractor ex = net.create_extractor();
    ex.input("data", in);
    ex.extract("output", out);
    // ex.extract("prob", out);
}

double time_min = DBL_MAX;
double time_max = -DBL_MAX;
double time_avg = 0;

// 批量推理模型调用测试
for (int i = 0; i < g_loop_count; i++)
{
    double start = ncnn::get_current_time();

    {
        ncnn::Extractor ex = net.create_extractor();
        ex.input("data", in);
        ex.extract("prob", out);
    }

    double end = ncnn::get_current_time();

    double time = end - start;
}
// 调用代码
benchmark("LeNet_p27_sim", ncnn::Mat(28, 28, 1), opt);
```

'''

> 重新编译ncnn安卓库

'''

```
$ adb push ${ncnn-android-build-path}/benchmark/benchncnn /data/local/tmp/
$ adb push <ncnn-root-dir>/benchmark/*.param /data/local/tmp/
$ adb shell
$ cd /data/local/tmp/
$ ./benchncnn [loop count] [num threads] [powersave] [gpu device] [cooling down]
```

'''

A920 7.1 MSM8909

'''

```
A920:/data/local/tmp # ./benchncnn 10 4 0 -1 1
loop_count = 10
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
     LeNet_p27_sim time =    1.87
     LeNet_p27_sim time =    2.21
     LeNet_p27_sim time =    2.01
     LeNet_p27_sim time =    2.01
     LeNet_p27_sim time =    2.43
     LeNet_p27_sim time =    2.20
     LeNet_p27_sim time =    2.22
     LeNet_p27_sim time =    2.26
     LeNet_p27_sim time =    2.83
     LeNet_p27_sim time =    2.59
     LeNet_p27_sim  min =    1.87  max =    2.83  avg =    2.26
```

'''

A930 7.1

'''

```
A930:/data/local/tmp # ./benchncnn 10 4 0 -1 1
loop_count = 10
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
     LeNet_p27_sim time =    0.45
     LeNet_p27_sim time =    0.92
     LeNet_p27_sim time =    0.80
     LeNet_p27_sim time =    0.80
```

```
    LeNet_p27_sim time =   0.79
    LeNet_p27_sim time =   0.83
    LeNet_p27_sim time =   0.80
    LeNet_p27_sim time =   0.83
    LeNet_p27_sim time =   0.81
    LeNet_p27_sim time =   0.80
    LeNet_p27_sim  min =   0.45  max =   0.92  avg =   0.78
```

'''

A920 5.1 MSM8909

'''

```
root@A920:/data/local/tmp # ./benchncnn 10 4 0 -1 1
loop_count = 10
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
    LeNet_p27_sim time =   2.55
    LeNet_p27_sim time =   2.40
    LeNet_p27_sim time =   1.27
    LeNet_p27_sim time =   1.30
    LeNet_p27_sim time =   1.14
    LeNet_p27_sim time =   1.13
    LeNet_p27_sim time =   1.15
    LeNet_p27_sim time =   1.17
    LeNet_p27_sim time =   1.08
    LeNet_p27_sim time =   1.22
    LeNet_p27_sim  min =   1.08  max =   2.55  avg =   1.44
```

'''