

零基础入门旷视天元MegEngine

MEGVII 旷视

欢迎加入“天元开发者交流群”



Android移动端模型推理部署

讲师：赵凯

天元开发者交流群

群号：1029741705



扫一扫二维码，加入群聊。

课程大纲:

以ShuffleNet V2为例, 介绍如何将预训练权重的模型部署到Android移动终端并做实时推理

- 预训练权重模型获取与转换
- 网络输入和输出
- MegEngine交叉编译及部署
- Android Camera预览实时推理实现
- Demo&QA



- 1 预训练权重模型的获取与转换
- 2 网络的输入和输出
- 3 MegEngine的交叉编译及部署
- 4 Android Camera 预览实时推理实现
- 5 Demo&QA



在移动终端上实现高效的推理，需要用到Native MegEngine Runtime

- MegEngine是需要加载经过序列化的静态图模型
- 使用jit.trace可以无缝将训练和测试使用的动态图转换成静态图
- 使用trace的dump接口完成静态图模型的序列化



<https://megengine.org.cn/model-hub>



使用MegEngine自带的ModelHub获取模型文件

1. 安装Python megengine框架
2. 通过megengine.hub.load接口获取预训练的动态图
3. 将动态图和静态网络转换为序列化模型

安装框架

```
pip3 install megengine -f https://megengine.org.cn/whl/mge.html
```

下载模型

```
megengine.hub.load('megengine/models', "shufflenet_v2_x1_0", pretrained=True)
```

序列化模型

```
jit.trace.dump("shufflenet_deploy.mge", arg_names=["data"])
```



模型转换参考代码

```
if __name__ == '__main__':  
  
    import megengine.hub  
    import megengine.functional as F  
    from megengine.jit import trace  
  
    net = megengine.hub.load("megengine/models", "shufflenet_v2_x1_0", pretrained=True)  
  
    @trace(symbolic=True)  
    def fun(data, *, net):  
        net.eval()  
        pred = net(data)  
        pred_normalized = F.softmax(pred)  
        return pred_normalized  
  
    data = np.random.random([1, 3, 224,  
                             224]).astype(np.float32)  
  
    fun.trace(data, net=net)  
    fun.dump(args.output, arg_names=["data"])
```

获取动态图

计算图全流程及优化

序列化模型

<https://megengine.org.cn/doc/latest/advanced/deployment.html>

天元官网: <https://megengine.org.cn/>

GitHub: <https://github.com/MegEngine>

天元开发者交流群
群号: 1029741705



扫一扫二维码, 加入群聊。

- 1 预训练权重模型的获取与转换
- 2 网络的输入和输出
- 3 MegEngine的交叉编译及部署
- 4 Android Camera 预览实时推理实现
- 5 Demo

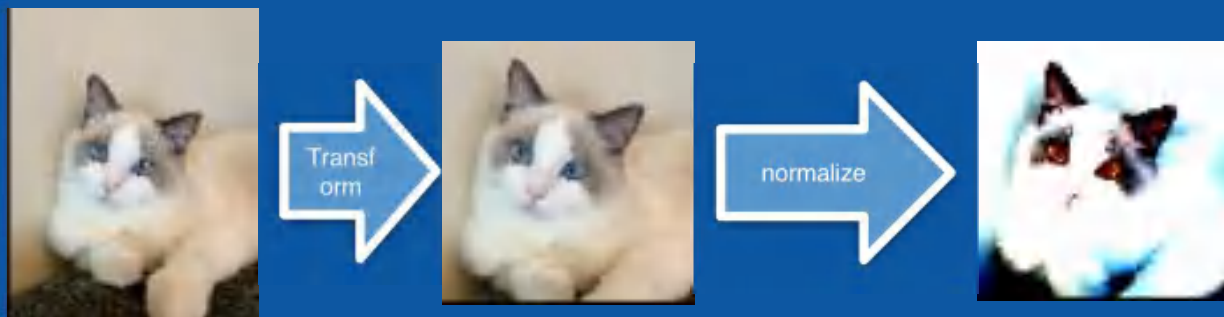


模型的输入需要做正确的预处理

预处理

1. 将图像短边缩放到256，并中心裁切到 224*224 的大小，在适配模型输入要求，尽可能保留ROI区域
2. 对裁切后的图像做归一化处理，确保输入图像是在同一个维度进行计算
3. 将图像转成模型需要的 'NCHW' 数据格式

```
transform = T.Compose(  
    [  
        T.Resize(256),  
        T.CenterCrop(224),  
        T.Normalize(  
            mean=[103.530, 116.280, 123.675], std=[57.375, 57.120, 58.395]  
        ), # BGR  
        T.ToMode("CHW"),  
    ]  
)
```



C++图像格式转换

```
1  #define RESIZE_WIDTH 256
2  #define RESIZE_HEIGHT 256
3  #define CROP_SIZE 224
4  void image_transform(const cv::Mat& src, cv::Mat& dst){
5
6      cv::Mat tmp;
7      cv::Mat tmp2;
8      // resize
9      cv::resize(src, tmp, cv::Size(RESIZE_WIDTH, RESIZE_HEIGHT), (0, 0), (0, 0), cv::INTER_LINEAR);
10
11     //center crop
12     const int offsetW = (tmp.cols - CROP_SIZE) / 2;
13     const int offsetH = (tmp.rows - CROP_SIZE) / 2;
14     const cv::Rect roi(offsetW, offsetH, CROP_SIZE, CROP_SIZE);
15     tmp = tmp(roi).clone();
16
17     // basic normalized, should convert to float32 first, otherwise got 1 or 0 but not 1-0
18     tmp.convertTo(tmp2, CV_32FC1);
19     cv::normalize(tmp2, dst, 0, 1, cv::NORM_MINMAX, CV_32F);
20 }
```

高宽resize到256x256

中心裁切为224x224

归一化处理



将图像数据按照 ‘NCHW’ 的格式传给网络输入

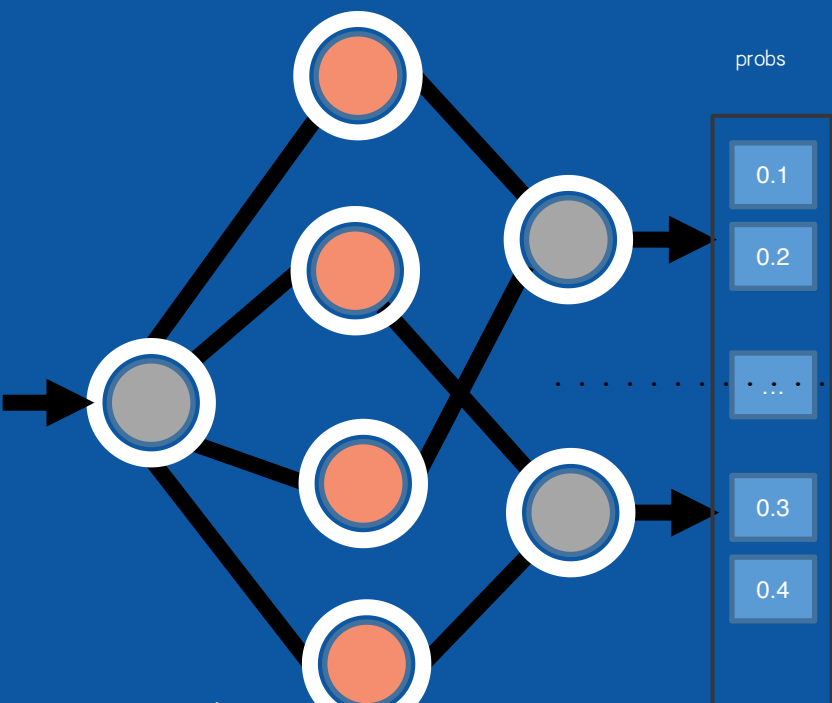


```
1 auto data = network.tensor_map.at("data");  
2 data->resize({1,3,224,224});  
3  
4 // 给输入 Tensor 赋值  
5 auto iptr = data->ptr<float>();  
6 auto iptr2 = iptr + 224*224;  
7 auto iptr3 = iptr2 + 224*224;  
8 auto imgptr = dst.ptr<float>();  
9 for (size_t j =0; j< 224*224; j++){  
10     iptr[j] = imgptr[3*j];  
11     iptr2[j] = imgptr[3*j +1];  
12     iptr3[j] = imgptr[3*j +2];  
13 }
```



分类网络的输出就是待分辨物体分类的可能值

1. 调用MegEngine推理接口执行推理。
2. 通过注册的回调函数,将结果保存下来,待后续访问



```
// 读取通过运行参数指定的模型文件,inp_file 需要输入的shufflenet_v2.mge文件
std::unique_ptr<serialization::InputFile> inp_file = serialization::InputFile::make_fs(argv[1]);

// 使用 GraphLoader 将模型文件转成 LoadResult, 包括了计算图和输入等信息
auto loader = serialization::GraphLoader::make(std::move(inp_file));
serialization::GraphLoadConfig config;
serialization::GraphLoader::LoadResult network =
    loader->load(config, false);

// 参考上一节代码, 将图像数据输入input layer

// 将网络编译为异步执行函数
// 输出output_var为一个字典的列表, second拿到键值对中的值, 并存在 predict 中
HostTensorND predict;
std::unique_ptr<cg::AsyncExecutable> func =
    network.graph->compile({make_callback_copy(
        network.output_var_map.begin()->second, predict)});
func->execute();
func->wait();
```

推理接口调用



获取网络输出的具体代码

- 回调函数 `make_callback_copy` 将结果保存在 `predict` 中
- 根据类别数量，依次打印出每个类别的 `confidence`

```
cg::ComputingGraph::OutputSpecItem make_callback_copy(SymbolVar dev, HostTensorND& host) {  
    auto cb = [&host](DeviceTensorND& d) { host.copy_from(d); };  
    return {dev, cb};  
}
```

回调函数

```
for (int i = 0; i < num_classes; i++){  
    sum += predict_ptr[i];  
    if (predict_ptr[i] > THRESHOLD)  
        std::cout << " Predicted: " << predict_ptr[i] << " i: " << i << std::endl;  
}
```

打印predict结果

```
item 284 -> Label: Siamese_cat, Predicted: 0.55
```



- 1 预训练权重模型的获取与转换
- 2 网络的输入和输出
- 3 MegEngine的交叉编译及部署**
- 4 Android Camera 预览实时推理实现
- 5 Demo



MegEngine C++ Runtime

- 目前支持Linux平台下的android/ios/arm linux/pc linux的交叉编译

```
scripts/  
├── cmake-build  
│   ├── cross_build_android_arm_inference.sh  
│   ├── cross_build_ios_arm_inference.sh  
│   ├── cross_build_linux_arm_inference.sh  
│   └── host_build.sh
```

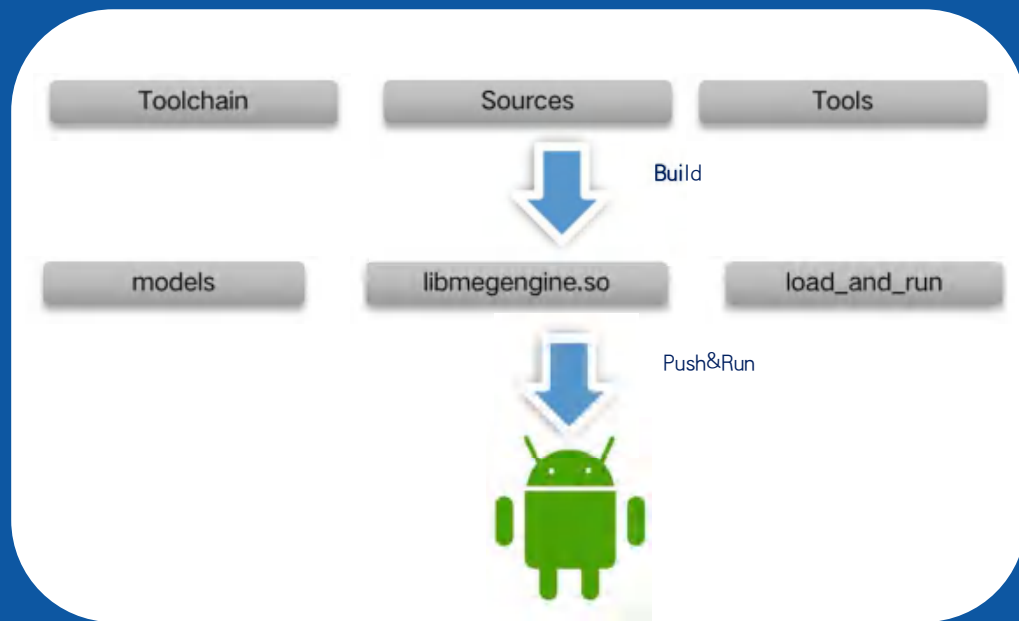
- 选择脚本完成对应平台的编译，获得libmegengine.so

默认编译位CPU版本，如编译CUDA GPU版本，添加-c

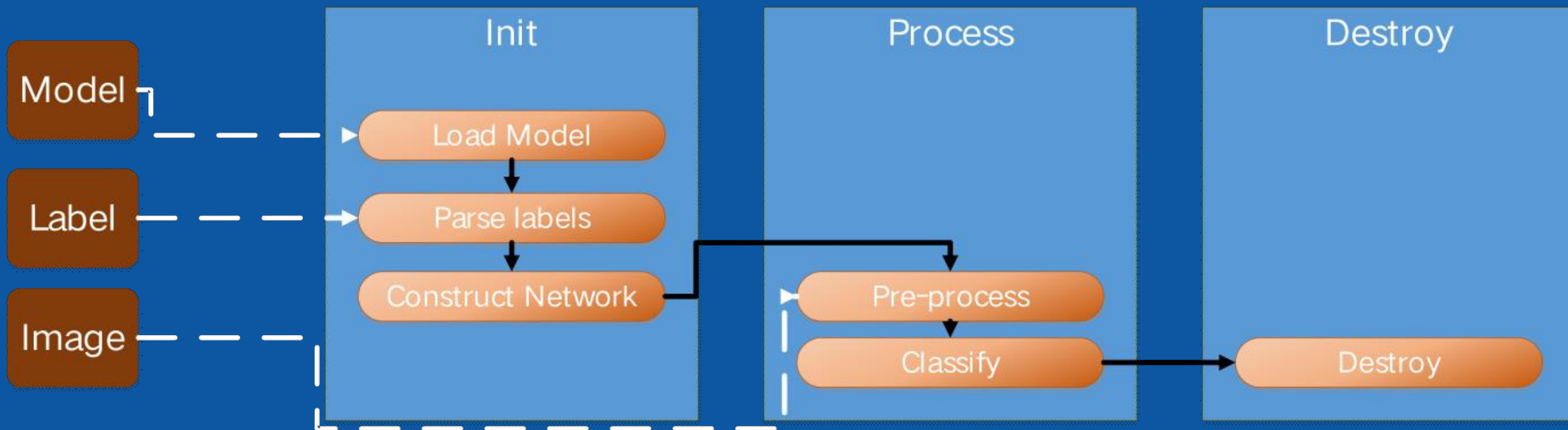
```
$ bash ./scripts/cmake-build/cross_build_android_arm_inference.sh
```

- 如需要测试，只需要构建可执行程序加载动态库并执行

Ex. Android NDK + CMake实现load and run



Load and run data flow



Label 文件

https://github.com/MegEngine/Models/blob/master/official/assets/imagenet_class_info.json

天元官网: <https://megengine.org.cn/>

GitHub: <https://github.com/MegEngine>

天元开发者交流群
群号: 1029741705



```
typedef enum {  
    ROTATION_0 = 0,  
    ROTATION_90 = 90,  
    ROTATION_180 = 180,  
    ROTATION_270 = 270,  
} FRAME_ROTATION;
```

Native Interface Reference

zhaokai, 6 hours ago | 1 author (zhaokai)

```
typedef struct {  
    void *data;  
    size_t size;  
    int width;  
    int height;  
    FRAME_ROTATION rotation;  
} FrameData;
```

zhaokai, a day ago | 1 author (zhaokai)

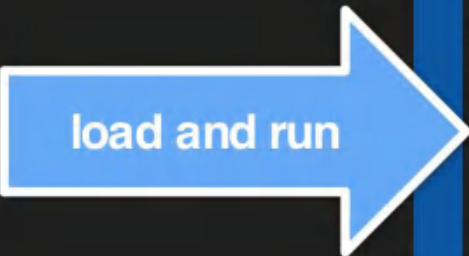
```
typedef struct {  
    char label[128];  
    float accuracy;  
} FrameResult;
```

zhaokai, a day ago * update ui

zhaokai, a day ago | 1 author (zhaokai)

```
typedef struct {  
    void *model_data;  
    size_t model_size;  
    const char *json;  
    int label_count;  
    int limit_count;  
    float threshold;  
} ModelInit;
```

```
ShuffleNetContext_PTR PUBLIC_API shufflenet_init(const ModelInit &init);  
void PUBLIC_API shufflenet_recognize(ShuffleNetContext_PTR sc, const FrameData &frame, int number,  
                                     FrameResult *results, int *output_size);  
void PUBLIC_API shufflenet_close(ShuffleNetContext_PTR sc);
```



```
cv::Mat jpeg_ = cv::imread(argv[2], cv::IMREAD_COLOR);  
  
fprintf(stdout, "pic %dx%d c%d\n", jpeg_.cols, jpeg_.rows, jpeg_.elemSize());  
vector<uint8_t> models;  
readBufFromFile(models, argv[1]);  
fprintf(stdout, "=====  
model size %ld\n", models.size());  
int num_size = 5;  
int output_size = 0;  
FrameResult f_results[5];  
ShuffleNetContext_PTR ptr = shufflenet_init({.model_data = models.data(), .model_size = models.size(), .json =  
IMAGENET_CLASS_INFOS, .limit_count = 1});  
if (ptr == nullptr)  
{  
    fprintf(stderr, "fail to init model\n");  
    return false;  
}  
  
shufflenet_recognize(ptr, FrameData{.data = jpeg_.data, .size = static_cast<size_t>(jpeg_.rows * jpeg_.cols * jpeg_.elemSize()),  
.width = jpeg_.cols, .height = jpeg_.rows, .rotation = ROTATION_0}, num_size, f_results, &output_size);  
for (int ii = 0; ii < output_size; ii++)  
{  
    LOGD("output result[%d] Label:%s, Predict:%.2f", ii, (f_results + ii)->label,  
        (f_results + ii)->accuracy);  
}  
shufflenet_close(ptr);
```

```
Shell$ LD_LIBRARY_PATH=./ ./shufflenet_loadrun ./shufflenet_deploy.mge ./cat.jpg
```

```
item 284 -> Label: Siamese cat, Predicted: 0.55
```

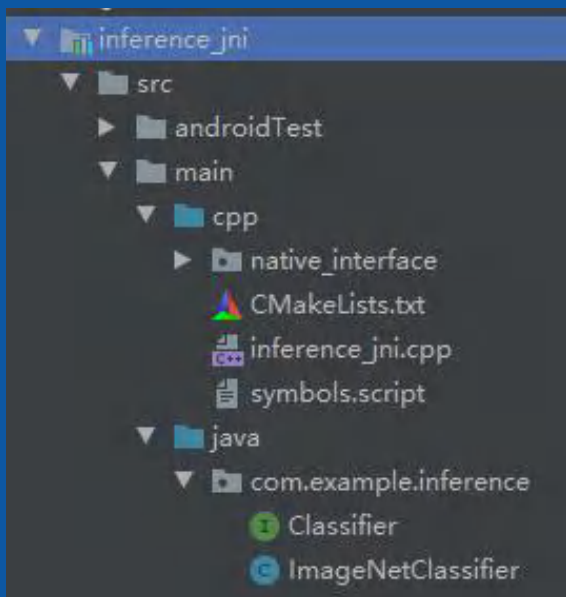
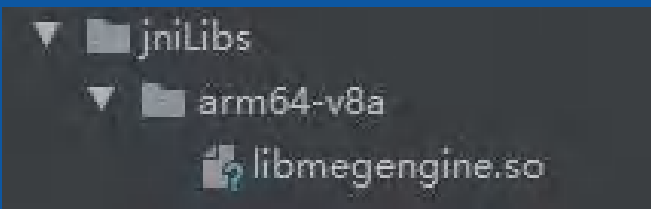
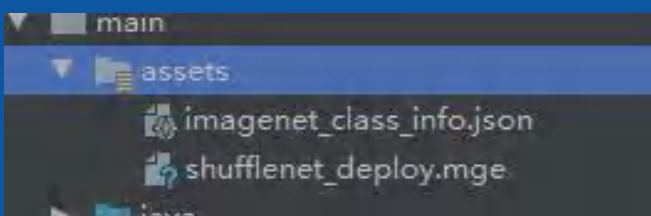


- 1 预训练权重模型的获取与转换
- 2 网络的输入和输出
- 3 MegEngine的交叉编译及部署
- 4 **Android Camera 预览实时推理实现**
- 5 Demo

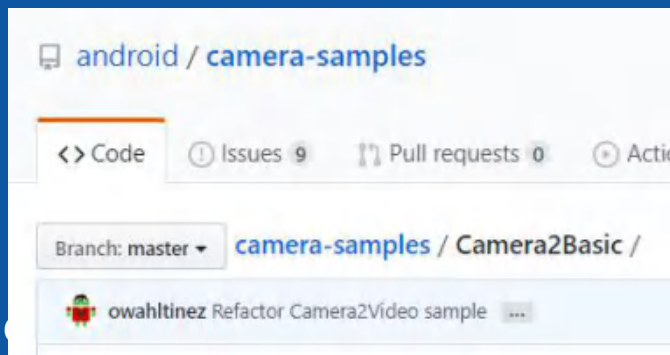


完成Android Camera API2 Demo需要实现的内容

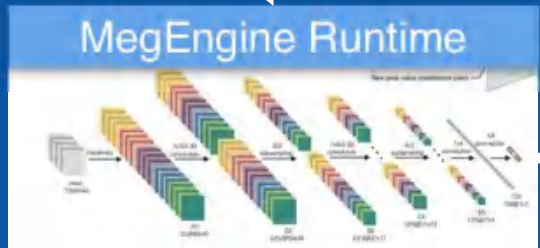
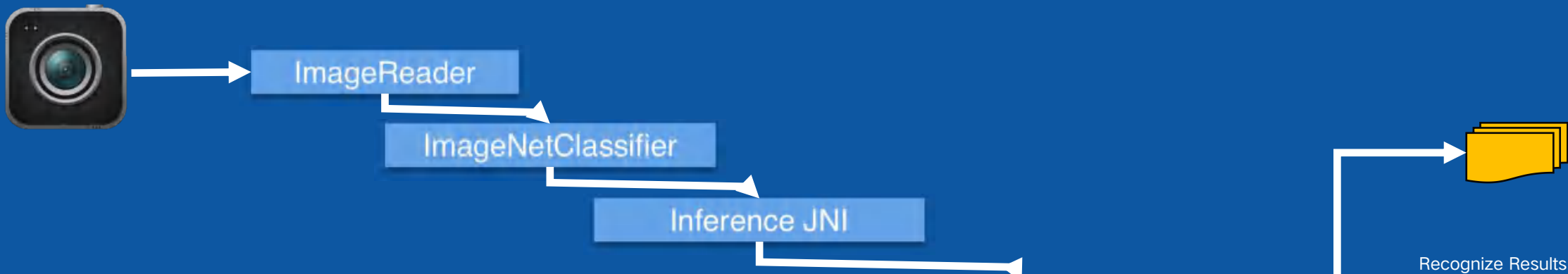
1. 将labels信息及Model文件以assets方式打包
2. 将libmegengine.so 作为动态库打包
3. 基于load and run封装的interface实现JNI interface及Java Interface
4. 获取Android Camera Preview数据, 送至Java Interface, 最终调用到MegEngine Runtime完成推理



<https://github.com/android/camera-samples/tree/master/Camera2Basic>



Camera Data Flow



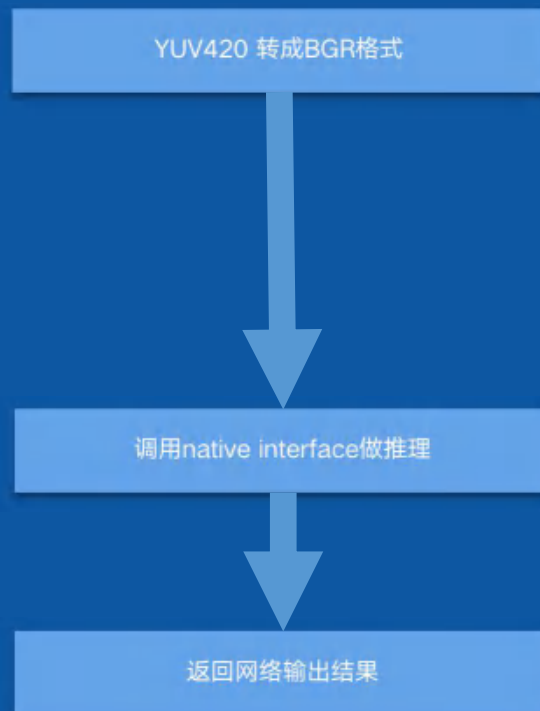
```
mPreviewSize = chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
    rotatedPreviewWidth, rotatedPreviewHeight, maxPreviewWidth,
    maxPreviewHeight, largest);
Log.d(TAG, msg: "preview size " + mPreviewSize);
// We fit the aspect ratio of TextureView to the size of preview we picked.
int orientation = getResources().getConfiguration().orientation;
if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
    mTextureView.setAspectRatio(
        mPreviewSize.getWidth(), mPreviewSize.getHeight());
} else {
    mTextureView.setAspectRatio(
        mPreviewSize.getHeight(), mPreviewSize.getWidth());
}

mPreviewImageReader = ImageReader.newInstance(mPreviewSize.getWidth(), mPreviewSize.getHeight(),
    ImageFormat.YUV_420_888, /*maxImages*/ maxImages: 5);
mPreviewImageReader.setOnImageAvailableListener(
    mOnPreviewAvailableListener, mBackgroundHandler);
```

```
imageRecognize = new Runnable() {
    @Override
    public void run() {
        recog_result_tpl = classifier.recognizeYUV420Tp1(yuvBytes[0], yuvBytes[1], yuvBytes[2],
            mPreviewSize.getWidth(), mPreviewSize.getHeight(), yRowStride, uvRowStride, uvPixelStride, finalRotation);
    }
};
```



```
std::vector<uint8_t> bgr;  
bgr.reserve(width * height * 3);  
ConvertYUV420ToBGR888(  
    reinterpret_cast<uint8_t *>(y_buff), reinterpret_cast<uint8_t *>(u_buff),  
    reinterpret_cast<uint8_t *>(v_buff), reinterpret_cast<uint8_t *>( bgr.data()),  
    width, height, y_row_stride, uv_row_stride, uv_pixel_stride);  
  
FrameResult fr = {0};  
int output_size = 0;  
int num_size = 1;  
FrameData frameData{.data = bgr.data(), .size=static_cast<size_t>(width * height * 3),  
                    .width=width, .height=height, .rotation=static_cast<FRAME_ROTATION>(rotation)};  
shufflenet_recognize(handle_ptr, frameData,  
                    num_size, &fr, &output_size);  
  
char ret_str[128] = {0};  
if (output_size > 0) {  
    snprintf(ret_str, 128, "Label: %s, Confidence: %.2f", fr.label, fr.accuracy);  
} else {  
    snprintf(ret_str, 128, "Label: ...., Confidence: 0.00");  
}
```



YUV420888 转 BGR888 格式

```
void ConvertYUV420ToBGR888(const uint8_t *const yData,
                            const uint8_t *const uData,
                            const uint8_t *const vData, uint8_t *const output,
                            const int width, const int height,
                            const int y_row_stride, const int uv_row_stride,
                            const int uv_pixel_stride) {
    uint8_t *out = output;

    for (int y = 0; y < height; y++) {
        const uint8_t *pY = yData + y_row_stride * y;

        const int uv_row_start = uv_row_stride * (y >> 1);
        const uint8_t *pU = uData + uv_row_start;
        const uint8_t *pV = vData + uv_row_start;

        for (int x = 0; x < width; x++) {
            const int uv_offset = (x >> 1) * uv_pixel_stride;
            YUV2BGR(pY[x], pU[uv_offset], pV[uv_offset], out, out + 1, out + 2);
            out += 3;
        }
    }
}
```

```
static inline void YUV2BGR(int nY, int nU, int nV, uint8_t *b,
                            nY -= 16;
                            nU -= 128;
                            nV -= 128;
                            if (nY < 0) nY = 0;

                            nR = (int)(1.164 * nY + 2.018 * nU);
                            nG = (int)(1.164 * nY - 0.813 * nV - 0.391 * nU);
                            nB = (int)(1.164 * nY + 1.596 * nV);
                            *r = MAX(0, nR);
                            *g = MAX(0, nG);
                            *b = MAX(0, nB);
    }
```



- 1 预训练权重模型的获取与转换
- 2 网络的输入和输出
- 3 MegEngine的交叉编译及部署
- 4 Android Camera 预览实时推理实现
- 5 Demo



演示



MEGVII 旷视

天元开发者交流群
群号: 1029741705



扫一扫二维码，加入群聊。

内容：

使用ResNet预训练模型完成基于Android Camera的实时推理

1. 了解ModelHub 使用
2. 了解模型预处理方法
3. 了解Android 移动端快速部署方法

要求：

完成课程内容后， 需要提供测试效果原图， standalone测试原图的运行结果图和Android Camera实时预览推理效果图



欢迎加入“天元开发者交流群”



行正则致远

AI向善，行胜于言。

MEGVII 旷视