Working with TFLite on Android with C++

Joe Bowser | Senior Computer Scientist

Resources (Slides, example code, etc)

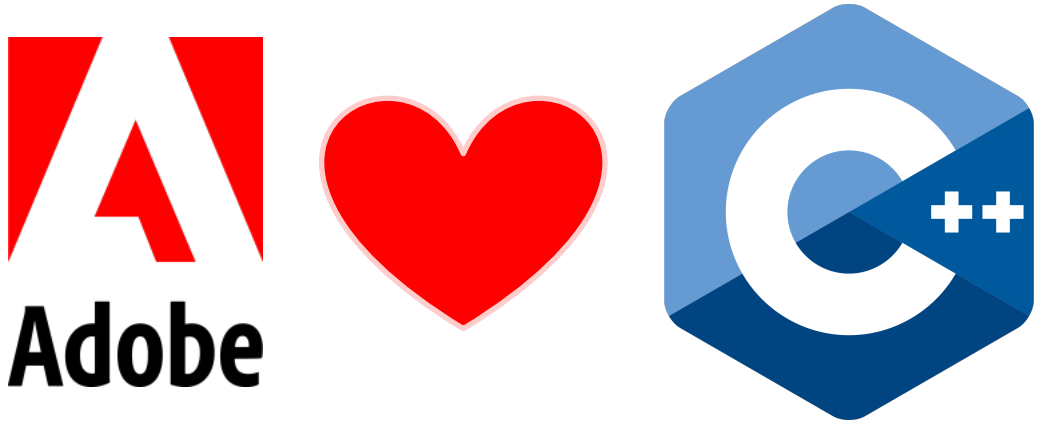
https://github.com/infil00p/tf_world

Adobe Sensei

- AI and Machine Learning for Customer Experiences
- New Features in various products are driven by Machine Learning
- Sensei-On-Device is an initiative to bring ML features into Adobe Desktop and Mobile products

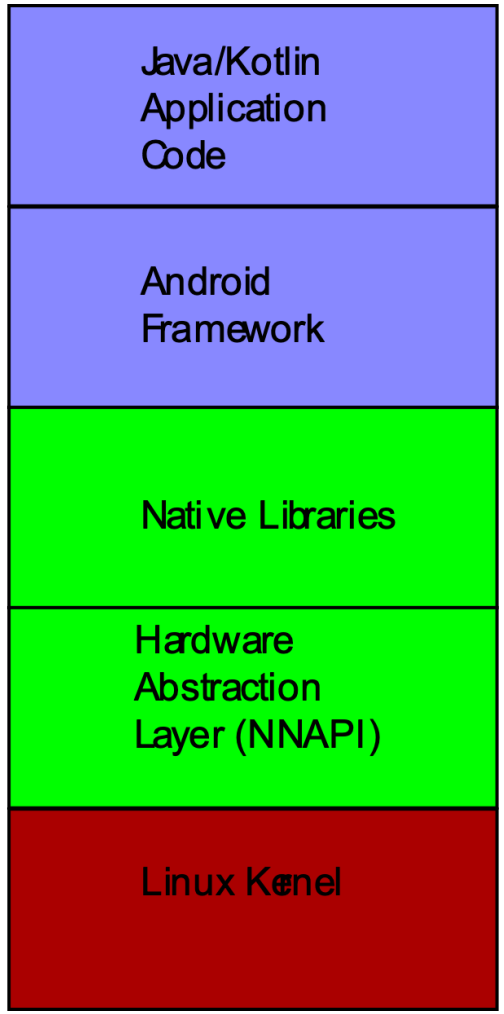
Why C++

- Adobe loves C++
- Adobe has a LOT of Business Logic in C++ across multiple platforms
- TFLite runs as a C++ library
- Many Image Processing Libraries that Adobe uses or contributes to are written in C++ or generate C++ (OpenCV, Halide, etc)



Let’s talk about Android Development

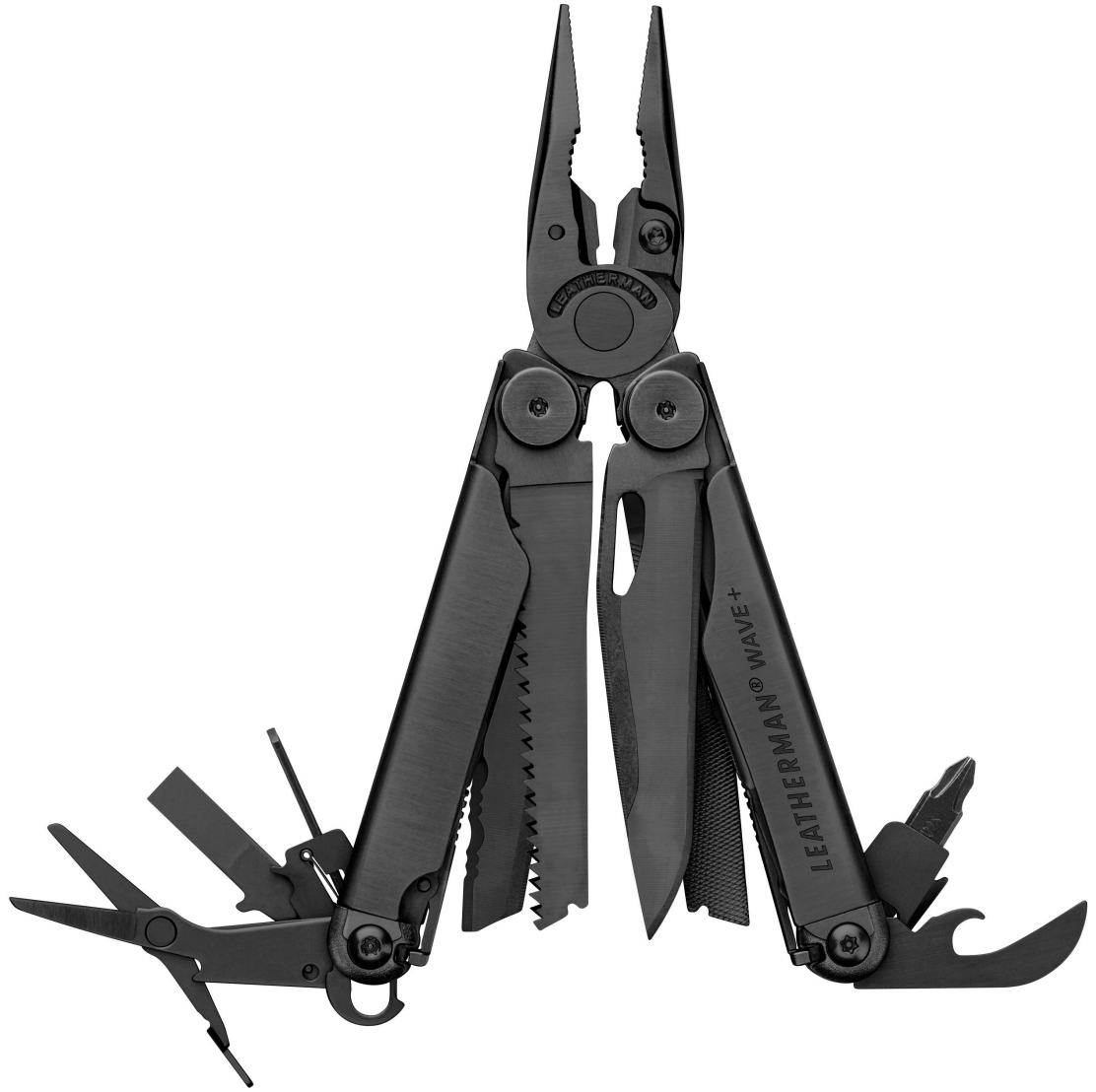
Android Internals



- Android consists of a higher bytecode application layer and a lower level Linux-based stack
- The majority of Android tools are for the application layer
- Libraries that require either lower level hardware visibility or higher performance exist on the lower-level stack

Advantages

- Access to features that are not visible to Java
- Common Code across multiple platforms
- Ability to share common C++ pre-processing code
- Tensorflow and TFLite are Open Source
- Easier to customize and extend classes in Tensorflow – (i.e. FlatBuffer validation)



Disadvantages

- Building is hard
- TFLite build lags behind the NDK
- Writing JNI code is hard
- Android Studio C++ support isn’t as good as Java/Kotlin

Let’s build an app – National Park Style Transfer

Tensorflow Lite, Android and C++

National Park Style Transfer

- Let’s make new posters based on old styles
- Posters were made in the Great Depression era
- Many artists have made derivative works from these posters
- We only have three from the Library of Congress that we can work with today

Artistic Style Transfer

- Example written in Python by Google
- https://www.tensorflow.org/lite/models/style_transfer/overview
- Entirely Tensorflow based (no external libraries)
- We need to figure out how to make this run on Android
- Let’s open the Jupyter Notebook

Adobe

How to get TFLite C++ for Android

TFLite Framework:

```
bazel build --config android_arm64 tensorflow/lite:libtensorflowlite.so
```

TFLite Open GL Delegate:

```
bazel build -c opt --config android_arm64 --copt -Os --copt
-DTFLITE_GPU_BINARY_RELEASE --copt -fvisibility=hidden --linkopt –s
--strip always :libtensorflowlite_gpu_gl.so
```

Make the models available

- Load the model into memory
- Use Asset Manager to access the assets
- https://developer.android.com/reference/android/content/res/AssetManager
- Kotlin code on the left (not from the example)
- In the case of Style Transfer, we do a style prediction and a style transfer

```
var model = "mobilenet_v1_1_ondevice.tflite"
var dataDirectory = this.filesDir
var inputFile = this.assets.open(model)
var outFile = File(dataDirectory, model)
var outStream = FileOutputStream(outFile)
inputFile.use { input ->
    outStream.use { fileOut ->
        input.copyTo(fileOut)
    }
}
```

Step 2: Load the models

- We load the two models at the same time in this example
- The Style Predict model could be built and run before the Style Transfer and results could be cached as an optimization

```
// Spin up the interpreter
style_predict_model_ = ::tflite::FlatBufferModel::BuildFromFile(style_predict_model.c_str());
transfer_model_ = ::tflite::FlatBufferModel::BuildFromFile(style_transfer_model.c_str());
::tflite::ops::builtin::BuiltinOpResolver resolver;
::tflite::InterpreterBuilder style_builder(*style_predict_model_, resolver);
::tflite::InterpreterBuilder transform_builder(*transfer_model_, resolver);
```

Step 3: Run the models

```
style_interpreter_->AllocateTensors();

auto tensorBuffer = style_interpreter_->typed_tensor<float>(inputIndex);
unsigned int tensorSize = styleMat.total() * styleMat.elemSize();
memcpy((void *) tensorBuffer, (void *)styleMat.data, tensorSize);

if(style_interpreter_->Invoke() != kTfLiteOk) {
    // Return the empty vector
    std::vector<float> emptyVec;
    return emptyVec;
}
```

Step 3a: Find a library to help with pre-processing

```
cv::Mat styleMat =  cv::imread(styleImage, CV_LOAD_IMAGE_COLOR);
cv::cvtColor(styleMat, styleMat, cv::COLOR_BGR2RGB);

styleMat.convertTo(styleMat, CV_32F, 1.f/255);
```

- Let’s use OpenCV in this example
- We just need to normalize between 0 and 1
- We then need to denormalize at the ends

3b. Test the preprocessing and inference

- File Permissions
  - We can write to our own local store without requesting permissions
  - You can copy output files using adb

```
cv::Mat StyleTransfer::preProcessImage(cv::Mat input) {

    std::string firstImageStr = APP_PATH + "/raw_input.jpg";
    cv::imwrite(firstImageStr, input);

    cv::Mat resizedImage;
    cv::Size imageSize(384, 384);
    cv::resize(input, resizedImage, imageSize);
    // I don't want to lose the alpha channel of the image coming in
    cv::cvtColor(resizedImage, resizedImage, cv::COLOR_BGR2RGB);

    std::string outputString = APP_PATH + "/test_input.jpg";
    cv::imwrite(outputString, resizedImage);

    resizedImage.convertTo(resizedImage, CV_32F, 1.f/255);

    return resizedImage;

}
```

Get the image back to Android

Two different strategies:

- ByteBuffers
  - No need to write to a file
  - Not as straight forward
- File System

```
} else if (reqCode == REQUEST_PICK_STYLE && data != null) {
    lastPickedStyle = data.getIntExtra(MainActivity.StyleCode, 0);
    String uri = doStyleTransform(lastPickedStyle);
    ImageView view = findViewById(R.id.imageView);
    Bitmap output = BitmapFactory.decodeFile(uri);
    view.setImageBitmap(output);
}
```

LIVE DEMO TIME

If you’re reading this after the fact, just imagine that you’re there!!!!

Things to remember

- RUN ON AN ACTUAL DEVICE
  - The TF Lite Interpreter is NOT the same as actual TFLite
- Make sure that you get your channels right on the input and output
  - i.e. OpenCV is BGR, Tensorflow models are RGB
  - Android Bitmaps are ARGB8888
- Write tests when making an actual application
  - Test Pre/Post Processing
  - Tiny Unit Testing Frameworks work well, like Catch2
- Test on actual versions of TFLite, not just on latest
  - Easier to reproduce issues
  - If you can’t use a release, create your own tags and keep track of them

Thank you

Questions?

#AdobeRemix
Hiroyuki-Mitsume Takahashi


Adobe