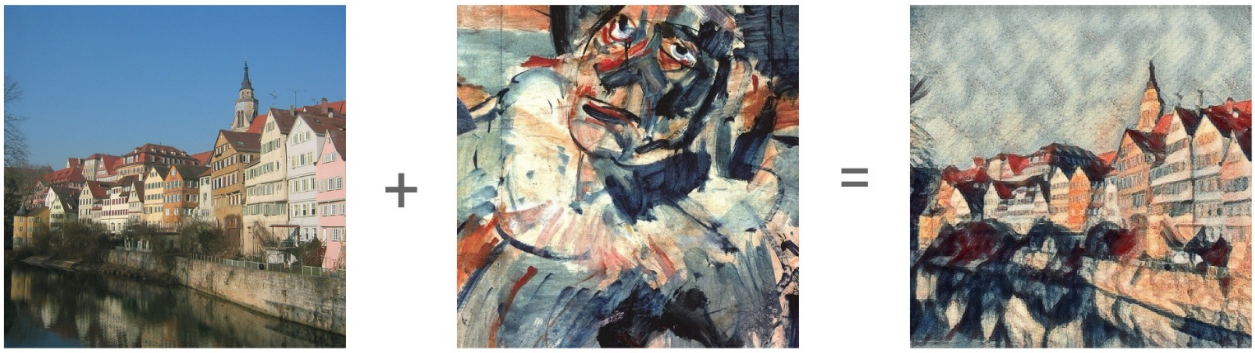
目錄

Introduction	1.1
Getting set up	1.2
Load the Android skeleton app	1.3
A learned representation of artistic style	1.4
Perform inference using TensorFlow	1.5
Congrats! You're done!	1.6

1. 介绍

什么是艺术风格转移？

最近，在深度学习中出现了很多发展方向。其中最令人激动的方向之一就是艺术风格转移，或者说是一种创造新图像的能力。它被称之为“拼图”，基于两张输入图像工作：一张图像表示艺术风格，一张图像表示内容。



使用这种技术，我们可以生成各种风格的美丽的新艺术作品。

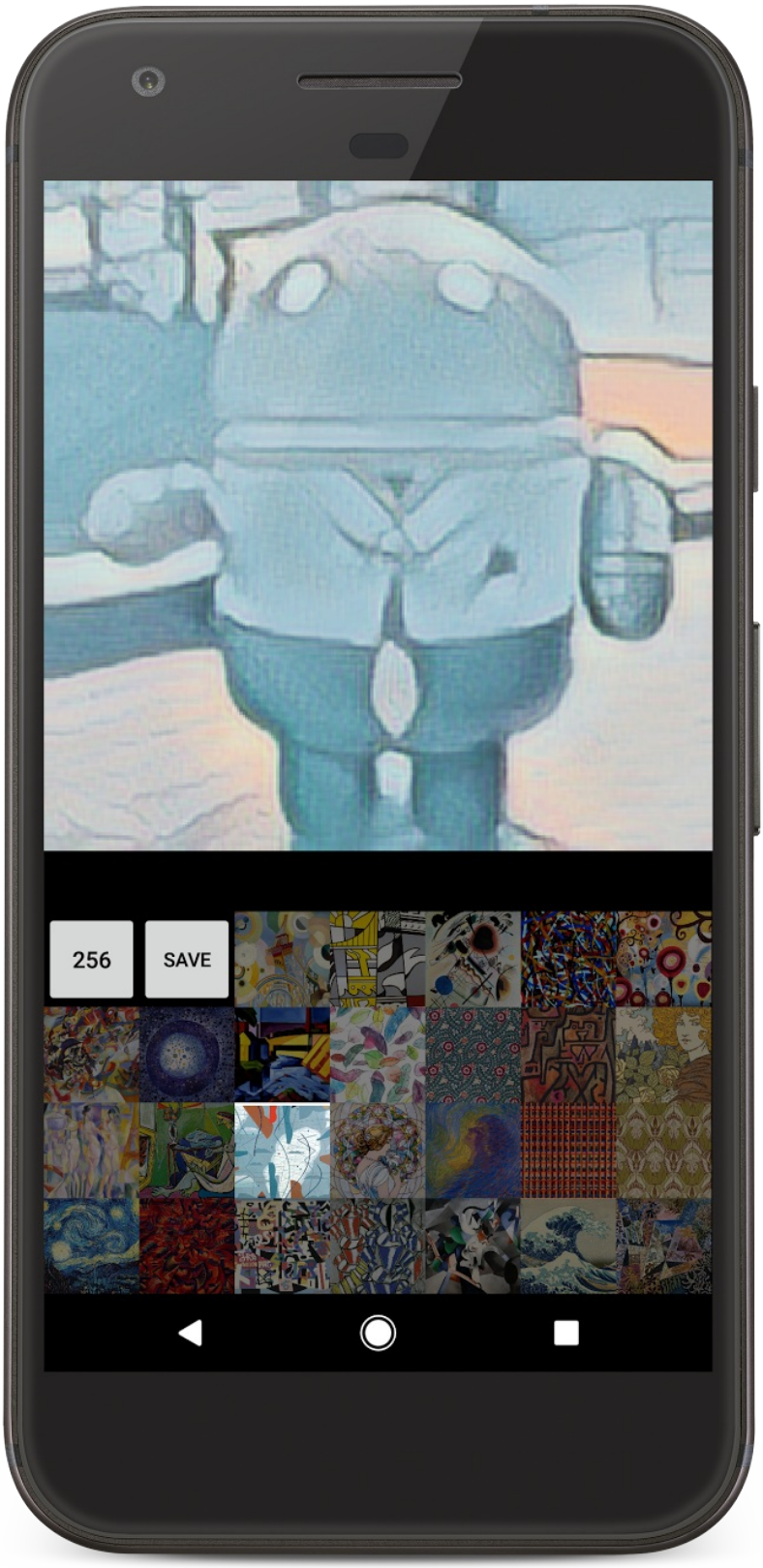
这个 codelab 将引导你完成在 Android 应用程序中使用艺术风格传递神经网络的过程。别担心，只需要 9 行代码。你还可以使用此 codelab 中涉及的技术来实现已经培训完毕的任何 TensorFlow 网络。

> 想看更多相关内容么？查看已经发布的 Google Research 和 Magenta 的博客。

我们将要创造什么？

在这个 codelab 中，你将要使用现有的 Android 应用程序，并在其中添加一个 TensorFlow 模型，最终使用设备的相机生成风格化的图像。你将建立以下技能：

- 在你的应用程序中使用 TensorFlow 的 Android Java 和 NDK 库
- 在 Android 应用程序中导入一个预训练的 TensorFlow 模型
- 在 Android 应用程序中执行逻辑推理
- 在 TensorFlow 图中执行特定张量


256
SAVE

你将要需要什么？

- 一个运行 Lollipop（API 21，v5.0）系统且相机在 Camera2 API 支持下（在API 21中引入）的设备
- v2.2 或者更高版本的 Android Studio
- 包含 v23 或者更高版本的 SDK 编译工具

> 注意：这个实验室专注于在一个 Android 应用程序中使用一个已存在的 TensorFlow 模型。Android 部分的代码将和提供的代码很大程度相同，但我们将说明 TensorFlow 的部分和 Android TensorFlow 专用的部分。

2. 获得资料起步

获得代码

有 2 种方法去抓取这个 codelab 的代码：下载包含代码的 ZIP 文件或者从 Github clone 项目。

下载 ZIP

点击我下载这个 codelab 的全部代码

解压下载的 zip 文件，这将解压出一个根目录（tensorflow-style-transfer-android-codelab-start），目录中包含我们将在这个 codelab 中使用的基本应用程序，包括所有应用程序资源。

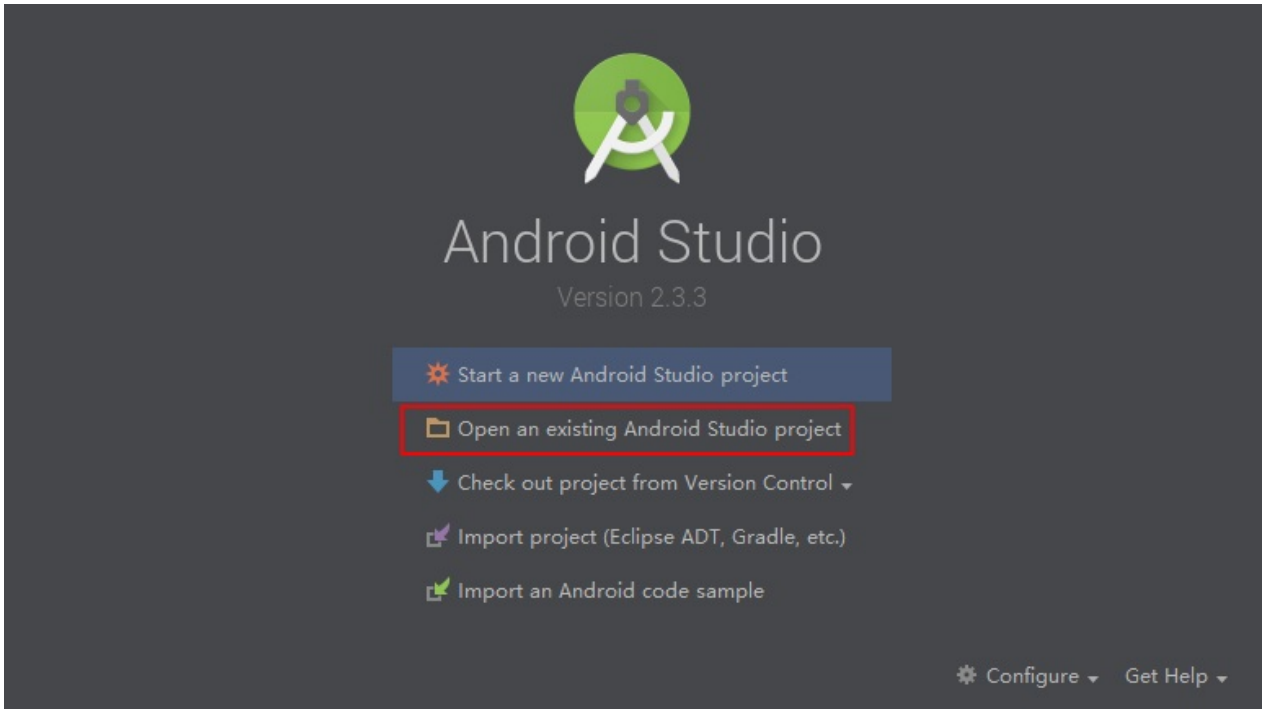
从 Github 检出

从 Github 中检出代码

```
git clone https://github.com/googlecodelabs/tensorflow-style-transfer-android
```

这将会创建一个包含你所需要的一切的目录。如果你想更改它，你可以分别使用 `git checkout codelab-start` 和 `git checkout codelab-finish` 在实验室的起始代码和最终代码中切换。

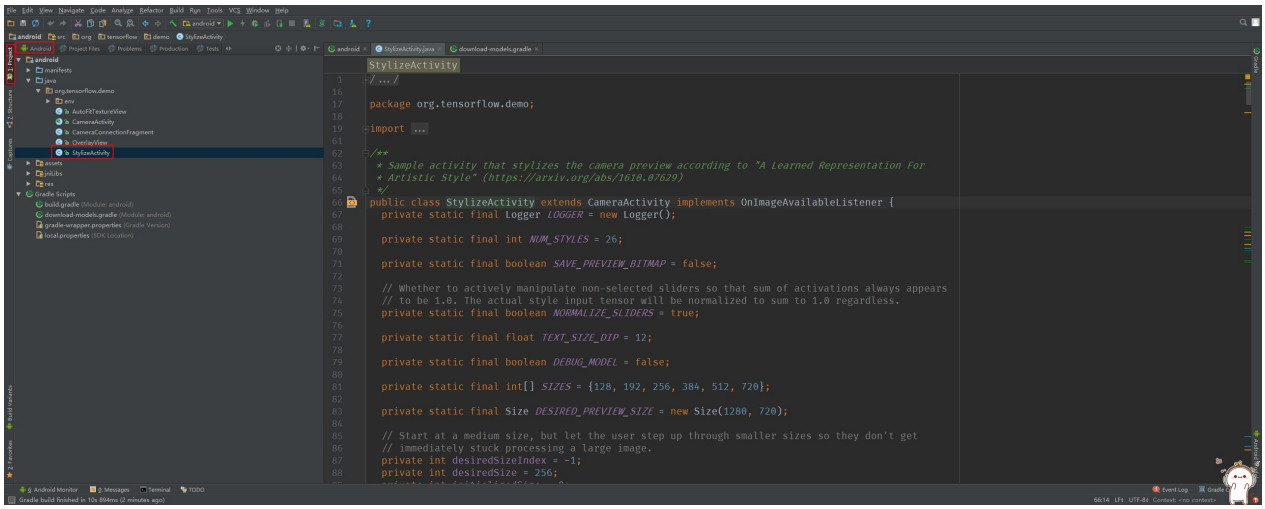
从 Android Studio 中读取代码



打开 Android Studio，选择 Open an existing Android Studio Project。在文件对话框中你将需要导航到你上一步中下载或者检出的 android 目录。举个例子，如果你在 home 目录中检出了代码，你将会打开 $HOME/tensorflow-style-transfer-android/android。

如果出现提示，你应该接受使用 Gradle wrapper，拒绝使用 Instant Run。

> 重要：
>
> - 你需要 open android 目录，而不是 tensorflow-style-transfer-android 目录。



一旦 Android Studio 导入了项目，使用文件浏览器打开 StylizeActivity 类，这就是我们将要工作的地方。如果你成功读取了文件，那么让我们到下一步去。

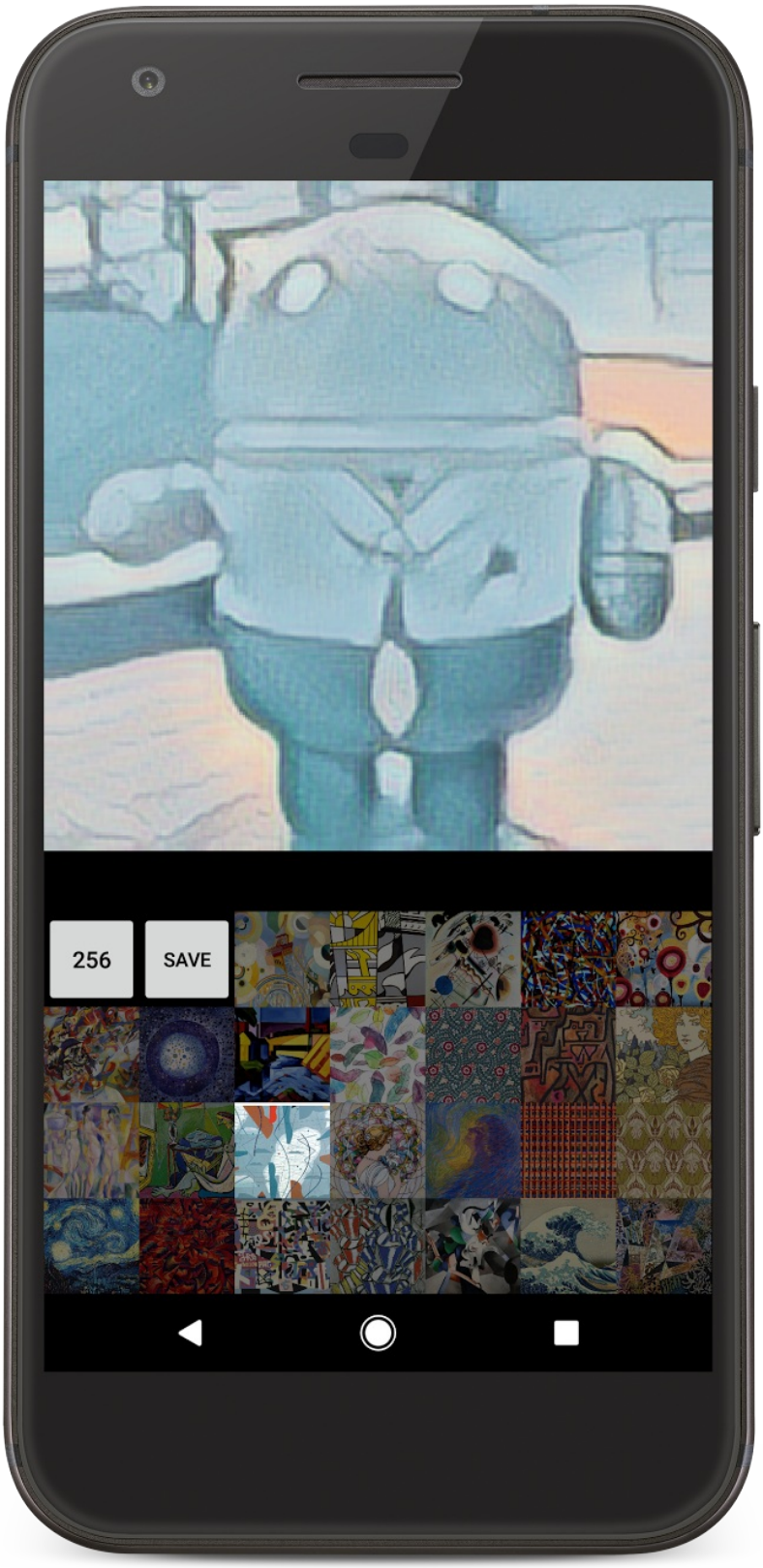
> 注意：这一部分的代码已经从 main TensorFlow 知识库中分叉出来，因此你不需要编译和检出整个知识库。如果你想编译最新的源码，你可以 从 Github 中检出

3. 读取 Android 应用程序骨架

这个应用程序是什么？

这个应用程序骨架包含一个 Android 应用程序，它从设备的相机中获取帧，并主活动的视图中呈现。

UI 控制


256
SAVE

- 第一个按钮，用一个数字标记（默认256）控制显示出图片的大小（最终通过风格转换网络）。更小的数字表示更小的图片，转换的速度更快，但质量低。相反，更大的数字表示更大的图片，但会花费更多的时间转换。
- 第二个按钮，用 `save` 标识，将会保存当前帧到你的设备中，方便你以后使用。
- 缩略图表示可用于转换的样式，每张图像都是一个滑块。你能够混合多个滑块，这将代表你希望应用于相机帧的每种风格的比例，这些比例以及相机帧代表网络的输入。

这些额外的代码是什么？

应用程序代码包括一些需要在本机 TensorFlow 和 Android Java 之间进行连接的帮助器。他们实现的细节并不重要，但你应该了解他们做的是什么。

`StylizeActivity.onPreviewSizeChosen(...)`

这个应用程序骨架使用本地相机帧，一旦授权并且可以使用相机，将调用此方法。

`StylizeActivity.setStyle(...)`

这使得风格滑块的规范化，使得它们的值总和为1.0，符合我们网络的期望。

`StylizeActivity.renderDebug(...)`

当你按设备上的音量向上或向下按钮时显示调试叠加层，包括 TensorFlow 的输出、性能指标、原始和无样式的图像。

`StylizeActivity.stylizeImage(...)`

这是我们要工作的地方，已经提供的代码在整数数组之间进行一些转换（由 Android 的 getPixels() 方法提供）。把表单 [0xRRGGBB] 转化成 [0.0,1.0] 的浮点数组，形式为 [r, g, b, r, g, b ...]。

`ImageUtils.*`

提供一些图像转换的帮助器。相机提供了 YUV 空间(因为它被最广泛的支持)，但网络希望 RGB，因此我们提供帮助器去转换图像。这些方法中大多数都用本地 C++ 实现来提高速度，代码位于 jni 目录下，但对于这个实验室通过预建的方式提供了 libtensorflow_demo.so 二进制文件。文件位于 libs 目录（在 Android Studio 中被定义为 jniLibs）下。如果这些不能使用，代码将会回到 Java 实现。

4. 艺术风格的学术代表

关于这个网络

> 注意：对于使用或者导入网络，理解这个网络的工作原理并不重要，但这里提供了一些背景资料。请随意跳过本节。

我们正在导入的网络是一些重要发展的结果。第一个风格转换的神经网络论文 (Gatys, et al. 2015) 引入了一种利用卷积图像分类网络性质的技术，其中低层级识别简单边缘和形状（风格的组成部分），高层级别识别更复杂的内容，最终生成一个拼图。这种技术适用于任意两个图像，但执行速度缓慢。

此后提出了一些改进措施，其中包括使用一种对每种风格的网络进行预训练的方法实现权衡 (Johnson, et al. 2016) ，从而实时生成图像。

最后，我们在这个实验室中使用的 神经网络 指出不同的神经网络代表不同的风格，可能会复制大量的信息。他们提出了一个用多种风格训练的单一神经网络，并且它有一个有趣的副产品，就是我们在这里所使用的混合风格的能力。

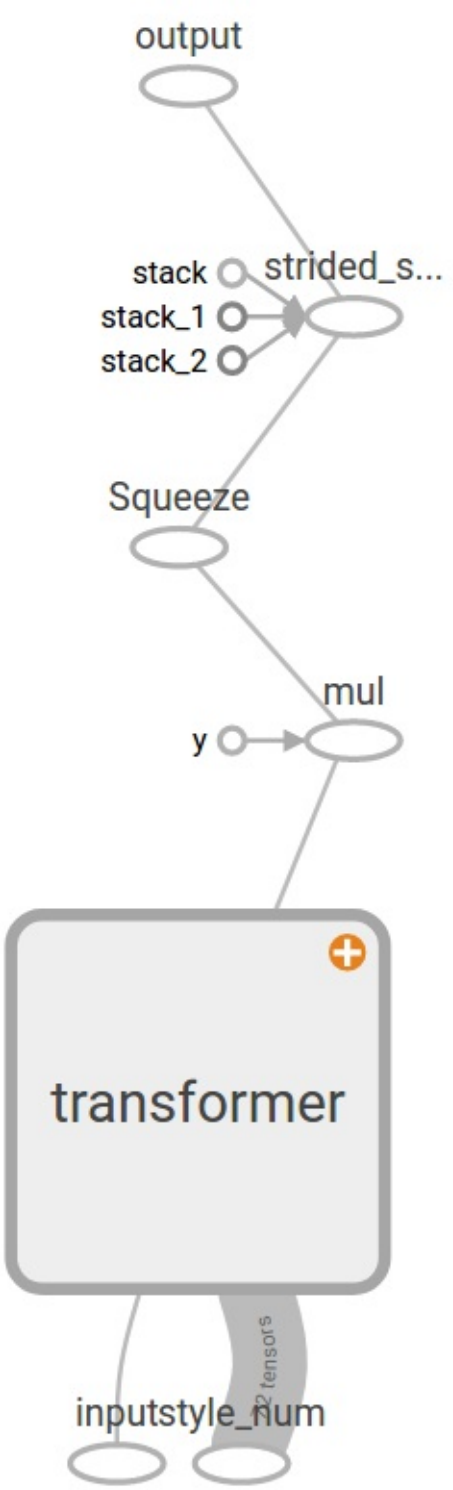
要对这些网络进行更为技术性的比较，以及对其他网络的评估，请查看 Cinjon Resnick 的评论文章。

在神经网络中

产生这个网络的原始 TensorFlow 代码可以在 Magenta 的 GitHub 页面 上获得，特别是 风格化的图像转换模型(README)。

在使用资源有限的环境（如移动应用程序）之前，这个模型已经被转化为更小的数据类型并且删除了冗余的计算。你可以在 Graph Transforms 的文档中获取关于这个过程的更多信息，并且在 TensorFlow for Poets II: Optimize for Mobile 的 Codelab 中尝试。

最后的结果是 stylize_quantized.pb 。这个文件在下方显示，这就是我们在这个应用程序中将会使用的。transformer 节点包含大多数的图，点击到交互版来展开它。


output
stack
stack_1
stack_2
strided_s...
Squeeze
mul
y
transformer
2 tensors
inputstyle_num

交互探索这张图

5. 使用 TensorFlow 进行推理

添加项目依赖

为了在项目中添加接口库和他们的依赖，我们需要添加 TensorFlow Android 接口库和 JAVA API。这些已经可以在 JCenter 使用，或者你可以在从 TensorFlow 源码 编译。

1. 在 Android Studio 中打开 build.gradle。
2. 通过将 API 添加到 android 代码块中的 dependencies 代码块，从而将 API 导入项目。（注意：这不是 buildscript 代码块）。

build.gradle

```
dependencies {
   compile 'org.tensorflow:tensorflow-android:1.2.0-preview'
}
```

1. 点击 Gradle sync 按钮，在 IDE 中使改变生效。

TensorFlow 推理接口

当执行 TensorFlow 代码时，你通常既要管理一个计算图，又要管理一个会话。（在 Getting Started 文档中提及）然而作为 Android 开发者将可能想用预编译的图实现接口，TensorFlow 提供了一个帮你管理图和会话的 JAVA 接口：

TensorFlowInferenceInterface

如果你需要更多的控制，TensorFlow 的 JAVA API 提供了你在 Python API 中熟悉的会话和图对象。

风格转换神经网络。

我们已经把在最后一部分描述的风格转换神经网络放在了项目的 `assets` 目录中，因此它可以供你使用。你也可以 直接下载 或者从 Magenta 的项目 中自行编译。

你可能值得打开交互式图预览以便看到我们将要很快引用的节点。（提示：点击鼠标悬停后出现的 “+” 图标打开 `transformer` 节点）。

交互探索这张图

添加推理代码

1. 在 `StylizeActivity.java` 中，在类的顶部附近添加以下成员字段（举例：在声明 `NUM_STYLES` 之前）

StylizeActivity.java

```
// Copy these lines below
private TensorFlowInferenceInterface inferenceInterface;

private static final String MODEL_FILE = "file:///android_asset/stylize_quantized.pb";

private static final String INPUT_NODE = "input";
private static final String STYLE_NODE = "style_num";
private static final String OUTPUT_NODE = "transformer/expand/conv3/conv/Sigmoid";

// Do not copy this line, you want to find it and paste before it.
private static final int NUM_STYLES = 26;
```

> 注意：这些在图中对应的节点有相同的名称。尝试在上面的交互图工具中找到它们。当你看到 / （斜杠） 的时候你需要展开它来查看子节点。

1. 在相同的类下，找到 `onPreviewSizeChosen` 方法，构造 `TensorFlowInferenceInterface`。我们使用这个方法来初始化，因为一旦向文件系统和相机获取权限，就会调用它。

StylizeActivity.java

```
@Override
public void onPreviewSizeChosen(final Size size, final int rotation) {
 // anywhere in here is fine

 inferenceInterface = new TensorFlowInferenceInterface(getAssets(), MODEL_FILE);

 // anywhere at all...
}
```

> 重要：如果你看到了“无法找到标识 ...” 的警告，你将需要添加 import 语句到此文件中。Android Studio 能帮你做这个事情，把光标移动到红色错误文本上方，按下 Alt-Enter 键，选择 Import ...

1. 现在找到 `stylizeImage` 方法，添加代码将我们的相机位图和选择的风格样式传递给

TensorFlow，并从图中抓取输出。这个目标在两个循环中间。

StylizeActivity.java

```
private void stylizeImage(final Bitmap bitmap) {
    // Find the code marked with: TODO: Process the image in TensorFlow here.
    // Then paste the following code in at that location.

    // Start copying here:

    // Copy the input data into TensorFlow.
    inferenceInterface.feed(INPUT_NODE, floatValues,
    1, bitmap.getWidth(), bitmap.getHeight(), 3);
    inferenceInterface.feed(STYLE_NODE, styleVals, NUM_STYLES);

    // Execute the output node's dependency sub-graph.
    inferenceInterface.run(new String[] {OUTPUT_NODE}, isDebug());

    // Copy the data from TensorFlow back into our array.
    inferenceInterface.fetch(OUTPUT_NODE, floatValues);

    // Don't copy this code, it's already in there.
    for (int i = 0; i < intValues.length; ++i) {
    // ...
}
```

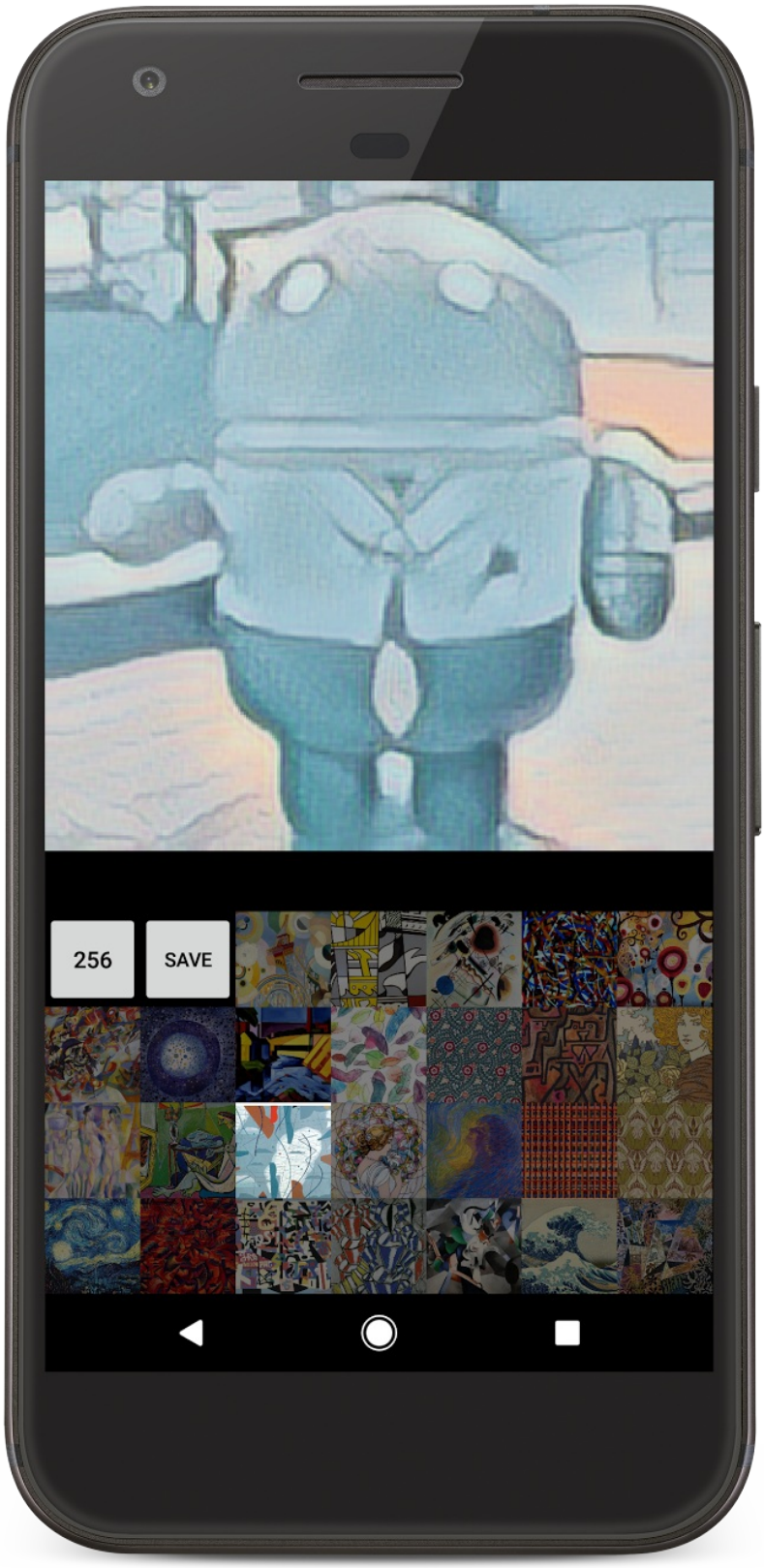
1. 可选：找到 `renderDebug`，在调试覆盖层中添加 TensorFlow 状态文本（当你按下音量键时触发）。

StylizeActivity.java

```
private void renderDebug(final Canvas canvas) {
    // ... provided code that does some drawing ...

    // Look for this line, but don't copy it, it's already there.
    final Vector<String> lines = new Vector<>();

    // Add these three lines right here:
    final String[] statLines = inferenceInterface.getStatString().split("\n");
    Collections.addAll(lines, statLines);
    lines.add("");

    // Don't add this line, it's already there
    lines.add("Frame: " + previewWidth + "x" + previewHeight);
    // ... more provided code for rendering the text ...
}
```

> 重要：如果你看到了“无法找到标识 ...”的警告，你将需要添加 import 语句到此文件中。Android Studio 能帮你做这个事情，把光标移动到红色错误文本上方，按下 Alt-Enter 键，选择 Import ...

1. 在 Android Studio 中按下 `Run` 按钮，等待项目编译
2. 你现在应该看到了在你的设备中发生的风格转换！


256
SAVE

6. 恭喜你！你做到了!

进一步阅读

- The TensorFlow Mobile site
- Cinjon Resnick's review of style transfer approaches
- Pete Warden's TensorFlow for Mobile Poets
- Neural Style Transfer: A Review

有意思的神经网络

- Pix2pix image generation using cGANs
- TensorFlow Models on GitHub
- TensorFlow Magenta project

其他 Code Labs

- TensorFlow for Poets
- TensorFlow for Poets II: Optimize for Mobile
- Nest Cams & TensorFlow
- Artistic Style Transfer in RenderScript