Read the Docs Template Documentation

Versión 1.0

Read the Docs

15 de diciembre de 2017

Indice:

TensorFlow es una biblioteca de código abierto para el cálculo numérico utilizando gráficos de flujo de datos. Originalmente fue desarrollado por el Equipo de Google Brain en la organización de investigación de Machine Learning de Google para el aprendizaje automático y la investigación de redes neuronales, pero el sistema es apto como para ser aplicable en una amplia variedad de otros dominios también. Llegó a la versión 1.0 en febrero de 2017 y ha continuado su rápido desarrollo.

CAPÍTULO 1

Descripción

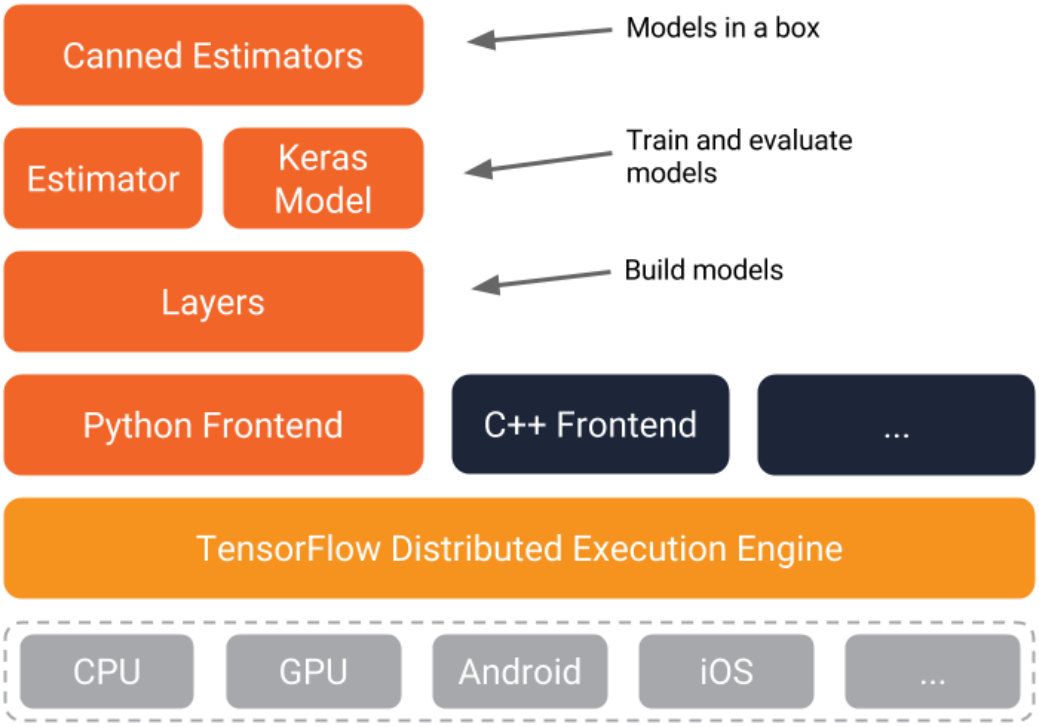
¿Qué es Tensorflow?.

Una biblioteca open-source para Machine Intelligence.

TensorFlow tiene APIs disponibles en varios lenguajes de programación para utilizarlas en nuestras aplicaciones. La API de Python está en el lenguaje más completo y fácil de usar, pero en otros idiomas las API pueden ser más fáciles de integrar en los proyectos y pueden ofrecer algunos ventajas de rendimiento en la ejecución de gráficos, por ejemplo.

Python, C++, Java, Go.

La API Layers proporciona una interfaz más simple para capas utilizadas comúnmente en modelos de deep learning. Además de eso, se encuentran API de nivel superior, que incluyen Keras y la API Estimator que facilita el entrenamiento y la evaluación de modelos distribuidos.



TensorFlow y la comunidad de software de código abierto:

TensorFlow fue de origen abierto en gran parte para permitir a la comunidad mejorarlo con contribuciones. Cualquier duda o consulta podemos ir a los siguientes enlaces Stack Overflow y mail.

CAPÍTULO 2

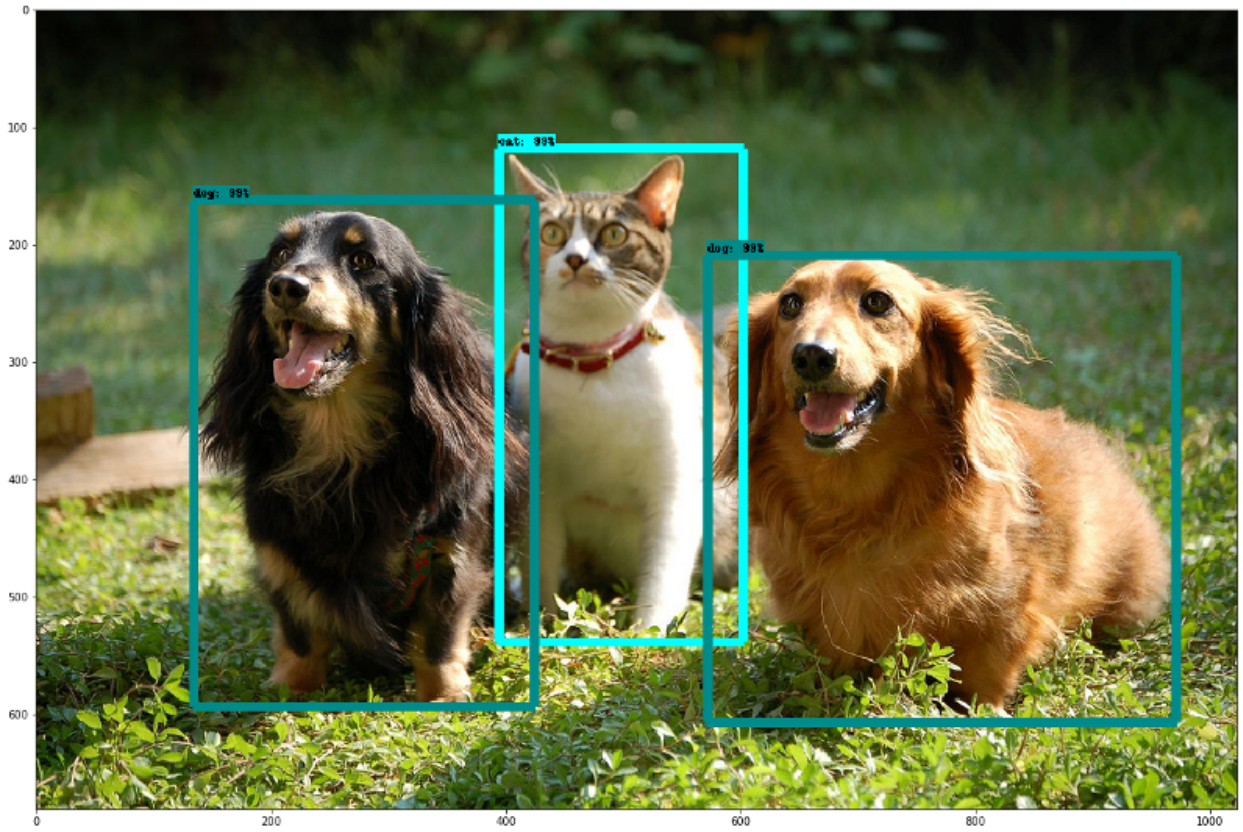
Modelos de código abierto de TensorFlow

Modelos de código abierto de TensorFlow

El equipo de TensorFlow ha abierto una gran cantidad de modelos. Puede encontrarlos en el repositorio tensorflow / models. Para muchos de estos, el código publicado incluye no solo el gráfico del modelo, sino también los pesos del modelo entrenado. Esto significa que puede probar estos modelos al instante, y puede utilizar muchos más con un proceso llamado aprendizaje de transferencia.

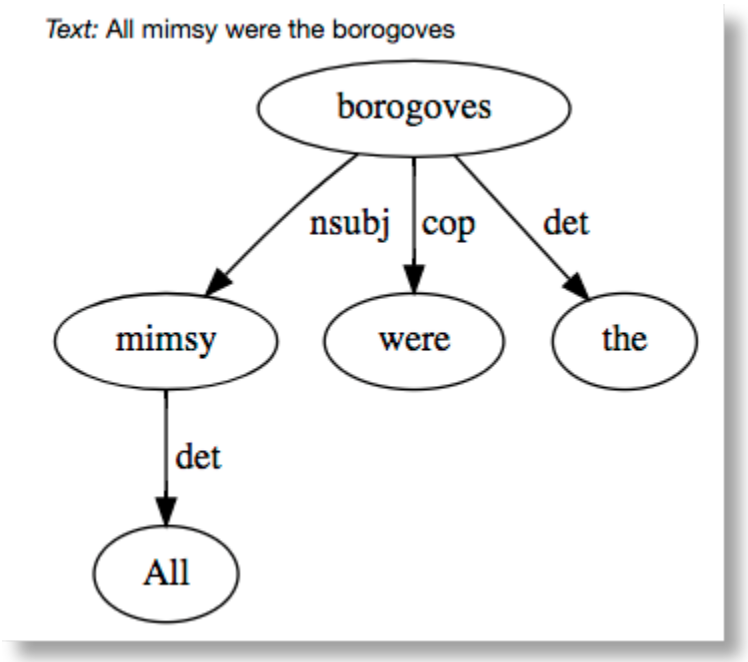
Estos son solo algunos de los modelos lanzados:

La API de Detección de Objetos: Sigue siendo un desafío central de aprendizaje automático crear modelos precisos de aprendizaje automático capaces de localizar e identificar múltiples objetos en una sola imagen. La API de Detección de Objetos TensorFlow de fuente abierta recientemente ha producido resultados de vanguardia (y se ha colocado primero en el desafío de detección de COCO).



tf-seq2seq: Google anunció previamente Google Neural Machine Translation (GNMT), un modelo de secuencia a secuencia (seq2seq) que ahora se utiliza en los sistemas de producción Google Translate. tf-seq2seq es un marco de código abierto seq2seq en TensorFlow que hace que sea fácil experimentar con modelos seq2seq y lograr resultados de última generación.

ParseySaurus es un conjunto de modelos prediseñados que reflejan una actualización a SyntaxNet. Los nuevos modelos usan una representación de entrada basada en caracteres y son mucho mejores para predecir el significado de palabras nuevas basadas tanto en su ortografía como en cómo se usan en el contexto. Son mucho más precisos que sus predecesores, particularmente para los idiomas en los que puede haber docenas de formas para cada palabra y muchas de estas formas podrían nunca ser observadas durante el entrenamiento, incluso en un corpus muy grande.



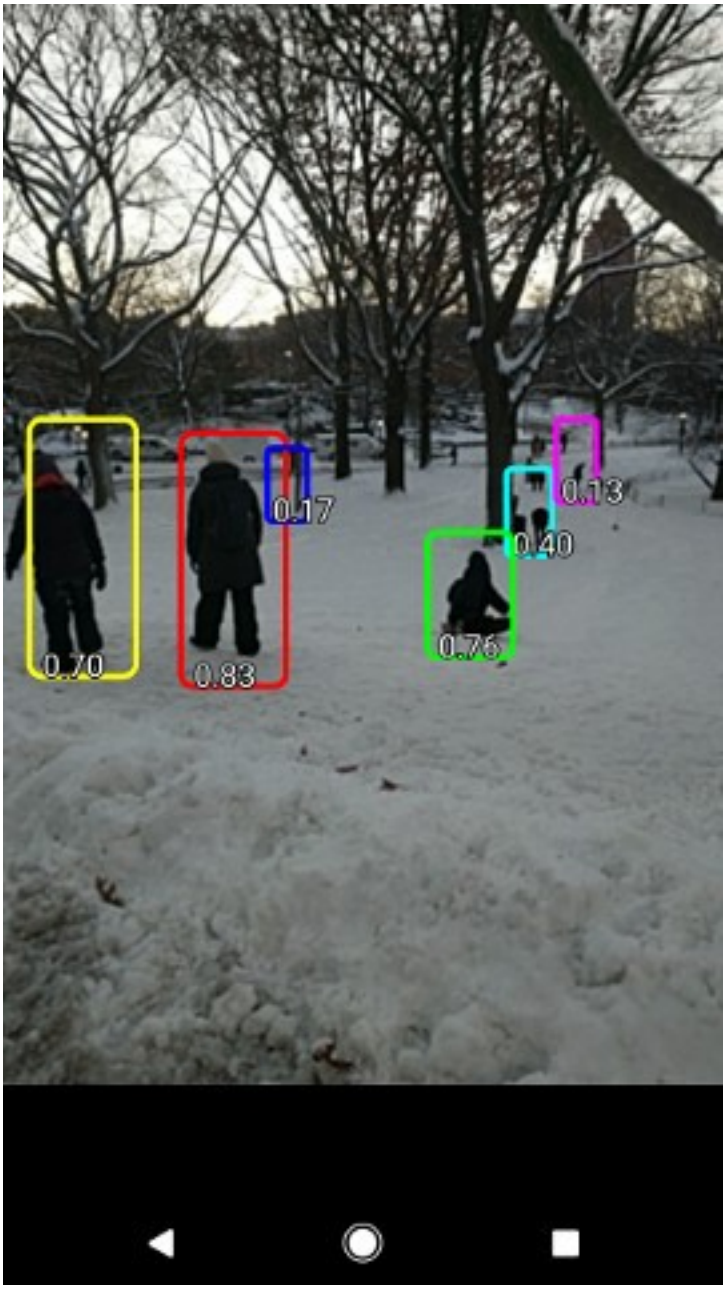
Multistyle Pastiche Generator del Proyecto Magenta : «Transferencia de estilo» es lo que sucede bajo aplicaciones que utilizan el estilo de una pintura a una foto. Este modelo Magenta amplía la transferencia de estilo de imagen al crear una red única crear una red única que puede realizar más de una estilización de una imagen.

Uso de TensorFlow en dispositivos móviles

TensorFlow está trabajando para ayudar a los desarrolladores a crear aplicaciones móviles ligeras, tanto al continuar reduciendo la huella del código como al respaldar la cuantificación de este. Uno de los proyectos de TensorFlow, MobileNet , está desarrollando un conjunto de modelos de visión por computador especialmente diseñados para abor-

dar las ventajas y desventajas de velocidad / precisión que deben considerarse en los dispositivos móviles o en las aplicaciones integradas. Los modelos de MobileNet se pueden encontrar en el repositorio de modelos TensorFlow también.

Una de las demostraciones más recientes de Android, TF Detect, usa un modelo de MobileNet entrenado usando la API de detección de objetos Tensorflow.



CAPÍTULO 3

El ecosistema TensorFlow

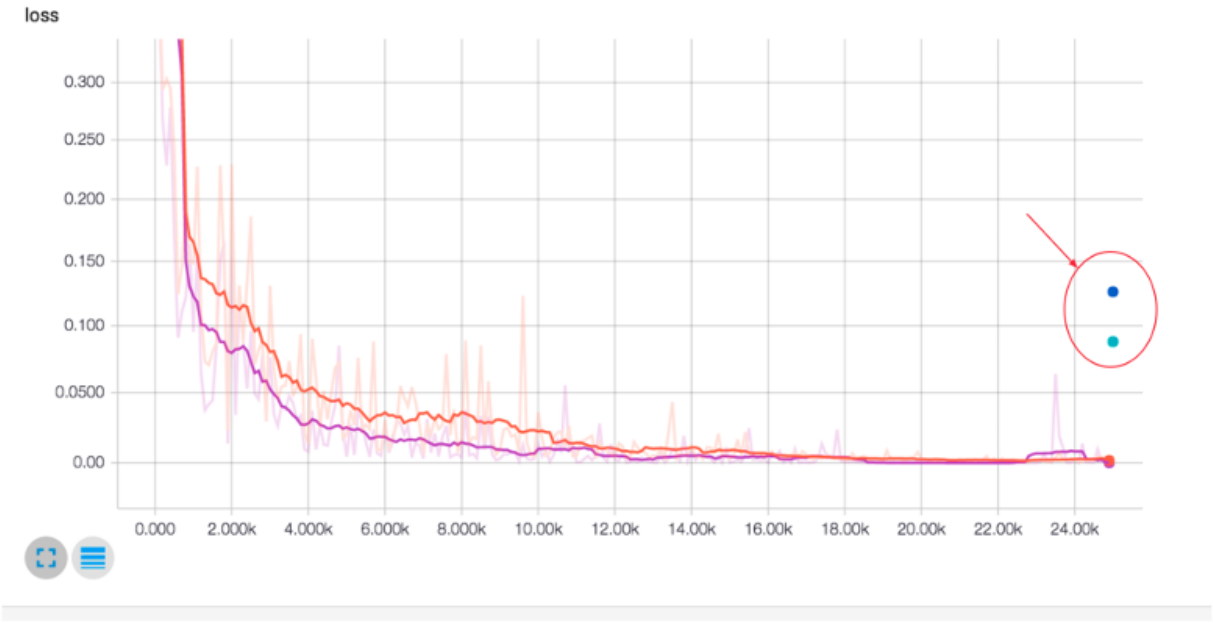
El ecosistema TensorFlow incluye muchas herramientas y bibliotecas, mencionaremos las principales:

TensorBoard: es un conjunto de aplicaciones web para inspeccionar, visualizar y comprender las ejecuciones y gráficos de TensorFlow. Podemos usar TensorBoard para ver los gráficos del modelo TensorFlow y acercar los detalles de las subsecciones.

Puede trazar métricas como la pérdida y la precisión durante una ejecución de entrenamiento; mostrar las visualizaciones del histograma de cómo un tensor está cambiando con el tiempo; mostrar datos adicionales, como imágenes; recopilar metadatos de tiempo de ejecución para una ejecución, como el uso total de la memoria y las formas del tensor para los nodos.

TensorBoard funciona leyendo los archivos de TensorFlow que contienen información resumida sobre el proceso de capacitación. Puede generar estos archivos cuando ejecuta trabajos de TensorFlow.

Puede usar TensorBoard para comparar ejecuciones de entrenamiento, recopilar estadísticas de tiempo de ejecución y generar histogramas.



Comparing two different optimizers for DNNClassifier

Una característica particularmente fascinante de TensorBoard es su visualizador de incrustaciones. Las incrustaciones son omnipresentes en el aprendizaje automático, y en el contexto de TensorFlow, a menudo es natural ver los tensores como puntos en el espacio, por lo que casi cualquier modelo de TensorFlow dará lugar a varias incrustaciones.

Datalab: Jupyter notebooks es una manera fácil de explorar interactivamente los datos, definir modelos TensorFlow e iniciar entrenamientos. Utilizando herramientas y productos de Google Cloud Platform como parte del flujo de trabajo, tal vez usando Google Cloud Storage o BigQuery para conjuntos de datos, o Apache Beam para preprocesamiento de datos, entonces Google Cloud Datalab proporciona un entorno basado en Jupyter con todas estas herramientas (y otros como NumPy, pandas, scikit-learn y Matplotlib), junto con TensorFlow, preinstalados y agrupados. Datalab es de código abierto.

Facets: El poder del aprendizaje de máquinas proviene de su capacidad de aprender patrones a partir de grandes cantidades de datos, por lo que comprender sus datos puede ser fundamental para construir un poderoso sistema de aprendizaje automático. Facets es una herramienta de visualización de datos de código abierto recientemente lanzada que ayuda a comprender sus conjuntos de datos de aprendizaje automático y obtener una idea de la forma y características de cada característica y ver de un vistazo cómo interactúan las características entre sí. Por ejemplo, puede ver sus conjuntos de datos de entrenamiento y prueba.

Otra herramienta de diagnóstico útil es el depurador TensorFlow , tfdbg , que permite ver la estructura interna y los estados de ejecución de los gráficos TensorFlow durante el entrenamiento y la inferencia.

Hay muchas otras herramientas y bibliotecas repositorios org de TensorFlow GitHub para conocerlos.

CAPÍTULO 4

Usos

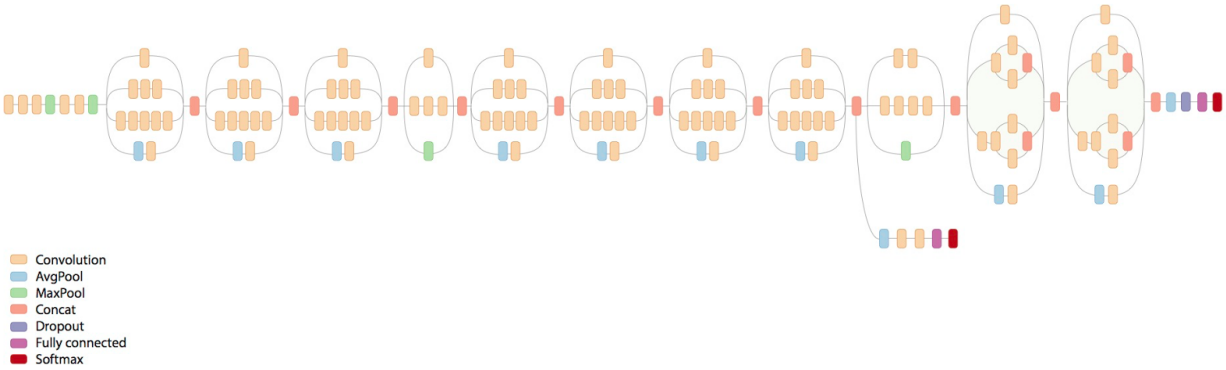
RankBrain

Organización : Dominio de Google : recuperación de información.

Descripción : despliegue a gran escala de redes neuronales profundas para el ranking de búsqueda en Google .

Más información : Google da vuelta a su búsqueda lucrativa a las máquinas de AI

Inception Image Classification Model

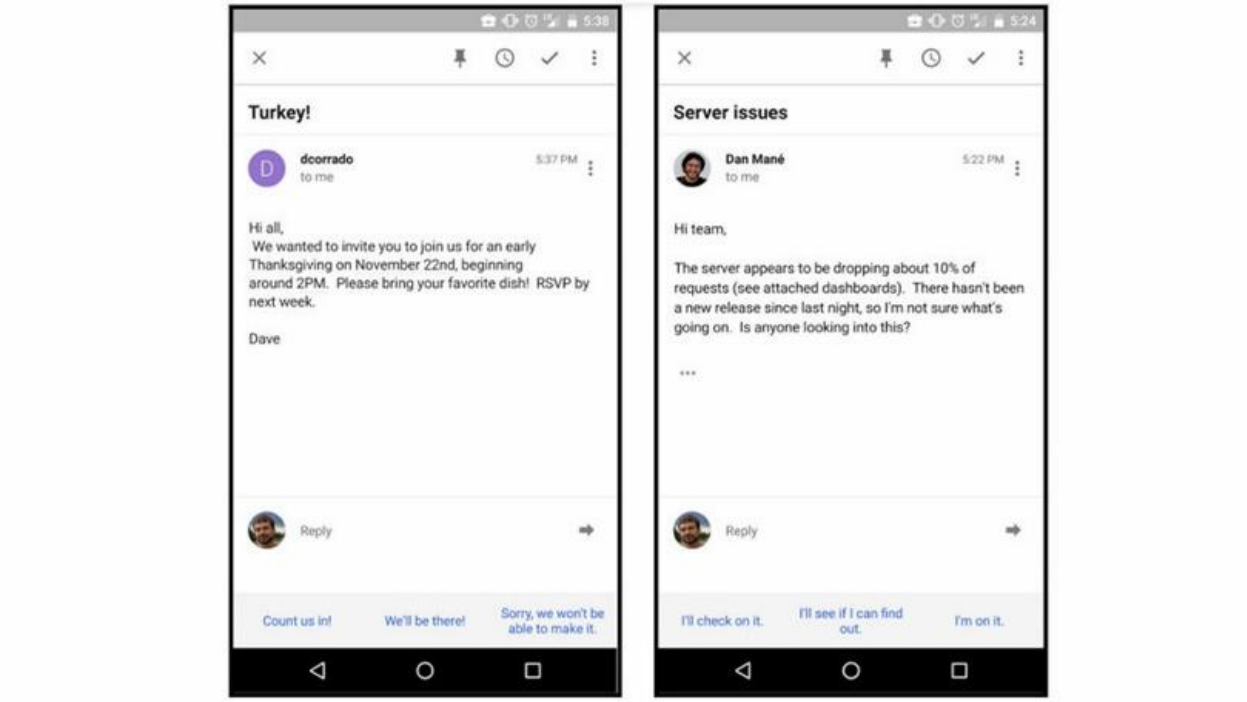


Organización : Google

Descripción : modelo básico y seguimiento de la investigación en modelos de visión por computadora de alta precisión, empezando por el modelo que ganó el desafío de clasificación de imágenes Imagenet 2014.

Más información : modelo de línea de base descrito en papel Arxiv

SmartReply

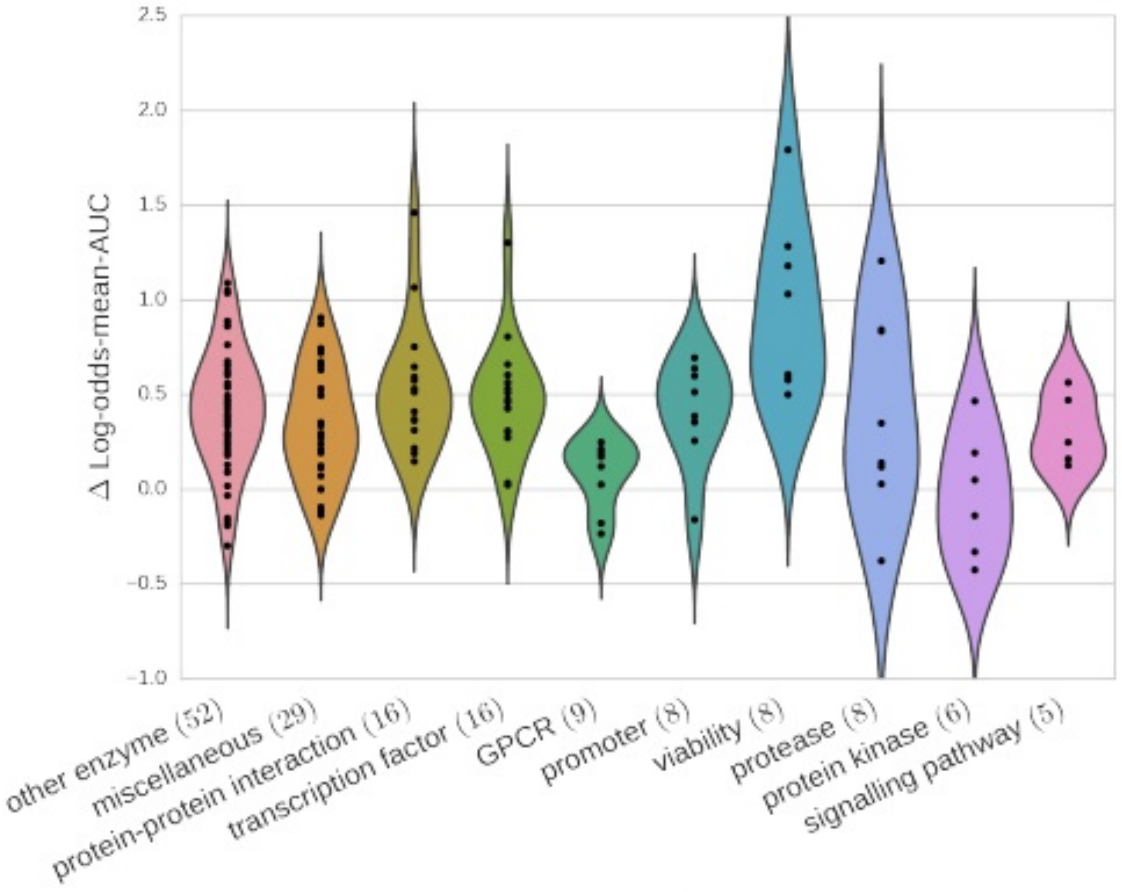


Organización : Google.

Descripción : Modelo Deep LSTM para generar automáticamente respuestas por correo electrónico.

Más información : Publicación de blog de investigación de Google

Massively Multitask Networks for Drug Discovery



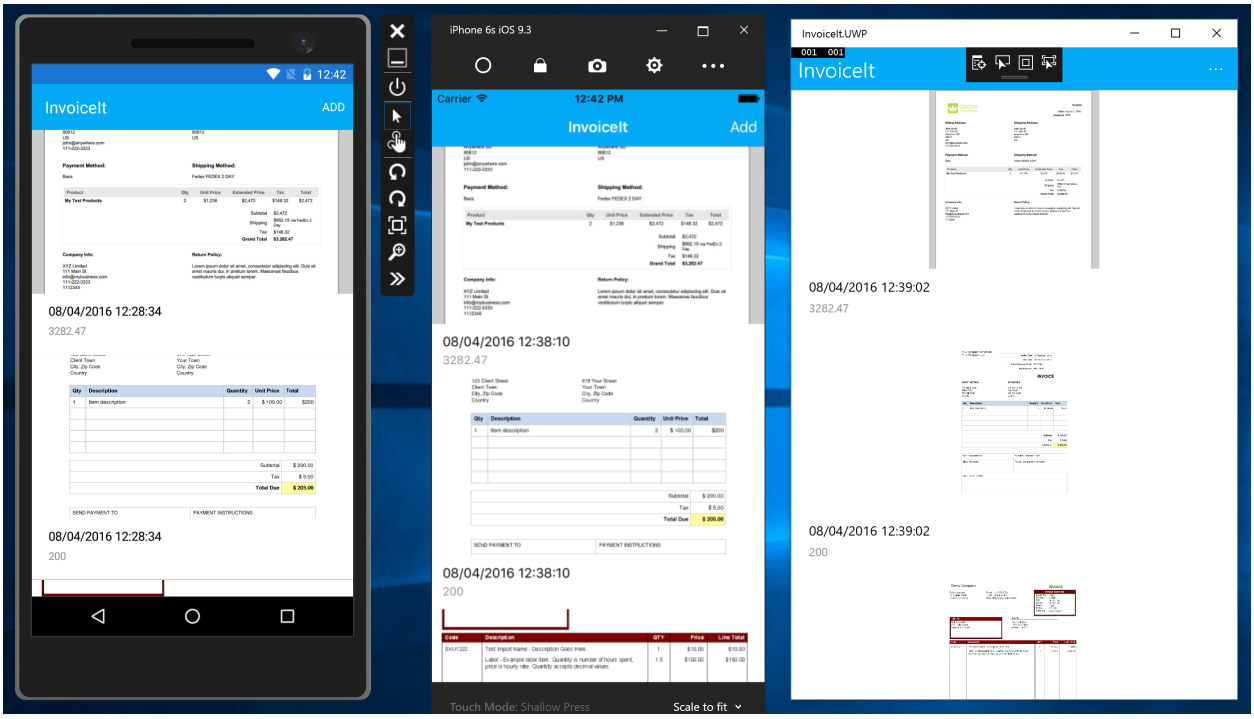
Organización : Google y la Universidad de Stanford.

Dominio : descubrimiento de fármacos.

Descripción : Un modelo de red neuronal profunda para identificar candidatos de fármacos prometedores.

Más información : papel Arxiv

On-Device Computer Vision for OCR



Organización : Google.

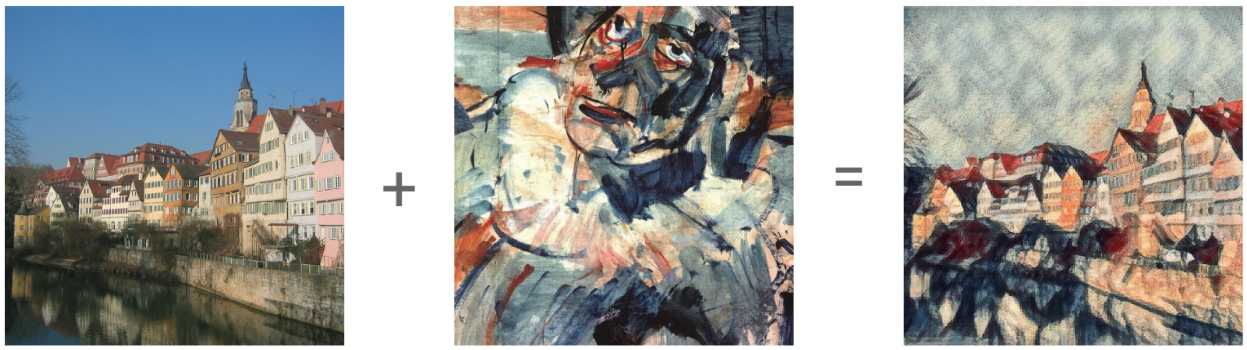
Descripción : modelo de visión artificial en el dispositivo para hacer reconocimiento óptico de caracteres para permitir la traducción en tiempo real.

Más información : publicación de blog de Google Research

CAPÍTULO 5

Transferencia de estilo artístico

Uno de los desarrollos más interesantes en el aprendizaje profundo que ha surgido recientemente es la transferencia de estilo artístico , o la capacidad de crear una nueva imagen, conocida como pastiche , basada en dos imágenes de entrada: una que representa el estilo artístico y otra que representa el contenido.



Content Style Pastiche

Usando esta técnica, podemos generar bellas obras de arte nuevas en una variedad de estilos.

Utilizaremos una red neuronal de transferencia de estilo artístico en una aplicación de Android en solo 9 líneas de código . Usar las técnicas descritas en este código para implementar cualquier red TensorFlow.

Usaremos:

Uso de las bibliotecas Java y nativas Android de TensorFlow en su aplicación.

Importación de un modelo capacitado de TensorFlow en una aplicación de Android.

Realizar inferencia en una aplicación de Android.

Accediendo a tensores específicos en un gráfico de TensorFlow.

Necesitamos:

Un dispositivo Android que ejecuta Lollipop (API 21, v5.0) con una cámara compatible con Camera2 API (introducido en API 21)

Android Studio v2.2 o superior

Incluyendo v23 (Marshmallow) o superior de las herramientas de compilación SDK

Obtener el código:

Hay dos formas de obtener la fuente de este codelab: descargue un archivo ZIP que contenga el código o clónelo desde GitHub.

Descargar ZIP

Verifique el código de GitHub:

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

Abra Android Studio y seleccione Importar proyecto. En el cuadro de diálogo de archivo, deberá navegar hasta «android» directorio dentro del directorio que descargó en el paso anterior. Por ejemplo, si revisó el código en su directorio personal, $HOME/tensorflow-style-transfer-android/android.

Si se le solicita, debe aceptar la sugerencia de usar el Gradle wrapper y rechazar usar Instant Run.

Importante:

> Es necesario importar / abrir el directorio android, no tensorflow-style-transfer-android directory

Una vez que Android Studio haya importado el proyecto, use el buscador de archivos para abrir la clase StylizeActivity. Aquí es donde trabajaremos: si puede cargar el archivo, OK, pasemos a la siguiente sección.

Cargue el esqueleto de la aplicación de Android

El esqueleto de esta aplicación contiene una aplicación de Android que toma fotogramas de la cámara del dispositivo y los muestra en una vista de la actividad principal.

Controles de interfaz de usuario:

El primer botón, etiquetado con un número (256 de forma predeterminada) controla el tamaño de la imagen para mostrar (y finalmente se ejecuta a través de la red de transferencia de estilo). Los números más pequeños significan imágenes más pequeñas, que serán más rápidas de transformar, pero de menor calidad. Por el contrario, las imágenes más grandes contendrán más detalles, pero tomarán más tiempo en transformarse.

El segundo botón, etiquetado save, guardará el marco actual en su dispositivo para que lo use más adelante.

Las miniaturas representan los estilos posibles que puede usar para transformar la alimentación de la cámara. Cada imagen es un control deslizante y puede combinar varios controles deslizantes que representarán las proporciones de cada estilo que desee aplicar a los marcos de la cámara. Estas relaciones, junto con el marco de la cámara, representan las entradas en la red.

El código de la aplicación incluye algunos helpers que se requieren para la interfaz entre TensorFlow nativo y Android Java. Los detalles de su implementación no son importantes, pero debe comprender lo que hacen.

> StylizeActivity.onPreviewSizeChosen(. . .)

El esqueleto de la aplicación utiliza un fragmento de cámara personalizado que llamará a este método una vez que se hayan otorgado los permisos y la cámara esté disponible para su uso.

> StylizeActivity.setStyle(. . .)

Esto mantiene los controles deslizantes de estilo normalizados de modo que sus valores se suman a 1.0, en línea con lo que nuestra red espera.

> StylizeActivity.renderDebug(. . .)

Proporciona una superposición de depuración al presionar los botones para subir o bajar el volumen en el dispositivo, incluida la salida de TensorFlow, las métricas de rendimiento y la imagen original sin estilo.

> StylizeActivity.stylizeImage(. . .)

Aquí es donde haremos nuestro trabajo. El código provisto realiza alguna conversión entre matrices de enteros (proporcionadas por getPixels() método de Android) del formulario [0xRRGGBB, . . .] a arreglos floats [0.0, 1.0] del formulario [r, g, b, r, g, b, . . .].

> ImageUtils.*

Proporciona algunos helpers para transformar imágenes. La cámara proporciona datos de imagen en YUV space (ya que es el más ampliamente compatible), pero la red espera RGB , por lo que ofrecemos helpers para convertir la imagen. La mayoría de estos se implementan en C ++ nativo para la velocidad; el código está en el jni directorio, pero para este laboratorio se proporciona a través de los libtensorflow_demo.so binarios preconstruidos en el libsdirectorio (definido como jniLibs en Android Studio). Si no están disponibles, el código recurrirá a una implementación de Java.

Acerca de esta red

Si bien no es crítico entender cómo funciona esta red para usarla o importarla, aquí se proporcionan algunos antecedentes. SE PUEDE SALTAR ESTA SECCIÓN

La red que estamos importando es el resultado de varios desarrollos importantes. El primer papel de transferencia de estilo neuronal (Gatys, et al., 2015) introdujo una técnica que explota las propiedades de las redes de clasificación de imágenes convolucionales, donde las capas inferiores identifican bordes y formas simples (componentes de estilo) y los niveles superiores identifican contenido más complejo para generar un «pastiche» Esta técnica funciona en dos imágenes, pero es lenta en su ejecución.

Desde entonces, se han propuesto varias mejoras, incluida una que compensa las redes de preentrenamiento para cada estilo (Johnson, et al., 2016) , lo que genera generación de imágenes en tiempo real.

Finalmente, la red que utilizamos en este laboratorio (Dumoulin, et al., 2016) intuyó que diferentes redes que representan diferentes estilos probablemente estarían duplicando mucha información, y propuso una red única entrenada en múltiples estilos. Un efecto secundario interesante de esto fue la capacidad de combinar estilos, que estamos usando aquí.

Para una comparación dela técnica de estas redes, así como la revisión de otras, consulte el artículo de revisión de Cinjon Resnick .

Dentro de la red El código original de TensorFlow que generó esta red está disponible en la página GitHub de Magenta, específicamente el modelo de transformación de imágenes estilizadas (README).

Antes de usarlo en un entorno con recursos limitados, como una aplicación móvil, este modelo se exportó y transformó para usar tipos de datos más pequeños y eliminar cálculos redundantes. Puede leer más sobre este proceso en el documento Graph Transforms.

El resultado final es el stylize_quantized.pb archivo, que se muestra a continuación, que usará en la aplicación. El nodo transformador contiene la mayor parte del gráfico, haga clic en la versión interactiva para expandirlo.

Agregar dependencias al proyecto:

Para agregar las bibliotecas de inferencia y sus dependencias a nuestro proyecto, debemos agregar la biblioteca de inferencia de Android TensorFlow y la API de Java, que está disponible en JCenter (en Archivos, tensorflow-android) o puede compilarlo desde la fuente TensorFlow.

1.Abrir build.gradle en Android Studio. 2.Agregue la API al proyecto agregándola a «dependencies block» dentro del android block (nota: este no es el buildscript block).

build.gradle

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

3.Haga clic en el botón de Gradle sync para que estos cambios estén disponibles en el␣
↪IDE.
```

La interfaz de inferencia de TensorFlow

Al ejecutar el código de TensorFlow, normalmente necesitaría administrar tanto un gráfico computacional como una sesión (como se describe en los documentos de Getting Started); sin embargo, dado que los desarrolladores de Android probablemente deseen realizar inferencias sobre un gráfico preconstruido, TensorFlow proporciona una interfaz Java que maneja la gráfica y la sesión: TensorFlowInferenceInterface.

Si necesita más control, la API de TensorFlow Java proporciona Session y los Graph objetos que puede conocer de la API de Python.

La red de transferencia de estilo

Hemos incluido la red de transferencia de estilo descrita en la última sección del assets, directorio del proyecto , por lo que estará disponible para el uso. También puede descargarlo directamente o compilarlo desde el proyecto Magenta .

Puede valer la pena abrir el visor gráfico interactivo para que pueda ver los nodos a los que haremos referencia en breve (Sugerencia : abra el nodo transformado haciendo clic en el ícono + que aparece una vez que se desplaza).

Grafico interactivo

Agregue el código de inferencia

En StylizeActivity.java, agregue los siguientes campos, cerca de la parte superior de la clase (por ejemplo, justo antes de la NUM_STYLES)

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;
```

cada uno de estos nodos corresponde a un nodo del mismo nombre en el gráfico. Intente encontrarlos en la herramienta gráfica interactiva anterior. Donde vea un / (carácter de barra) tendrá que expandir un nodo para ver sus elementos secundarios.

En la misma clase, encuentre el método onPreviewSizeChosen y construya el TensorFlowInferenceInterface. Utilizamos este método para la inicialización, ya que se llama una vez que se otorgan los permisos al sistema de archivos y a la cámara.

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

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

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

Importante : si recibe una advertencia sobre » No se puede encontrar el símbolo . . . «, deberá agregar las declaraciones de importación en este archivo. Android Studio puede hacer esto para usted si se mueve el cursor sobre el texto de error en rojo, pulse Alt-Intro , y selecciona Importar . . .

Ahora encuentre el método stylizeImage, agregue el código para pasar nuestro mapa de bits de la cámara y los estilos elegidos a TensorFlow y tome la salida del gráfico. Esto va entre los dos bucles.

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) {
 // ...
}
```

StylizeActivity.java

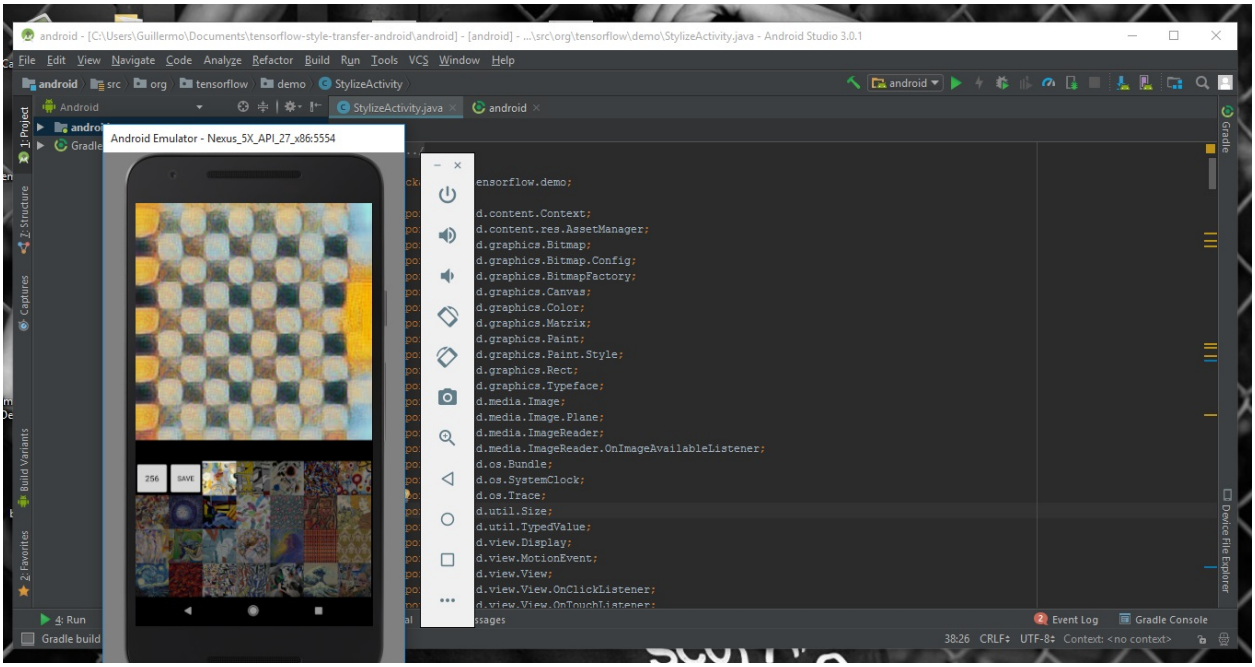
```
private void renderDebug(final Canvas canvas) {
 // ... provided code that does some drawing ...

 // 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 ...
}
```

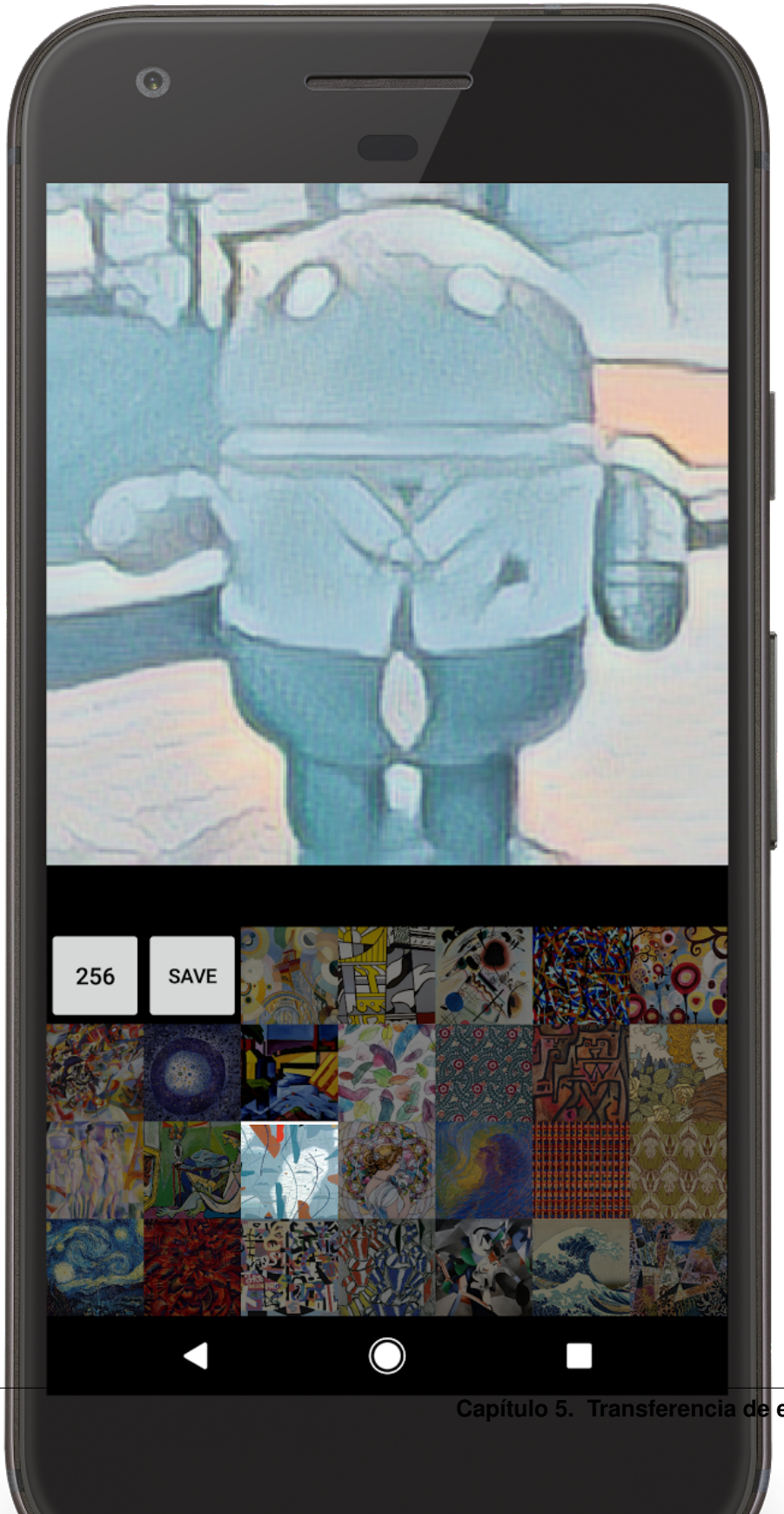
Importante : si recibe una advertencia sobre » No se puede encontrar el símbolo . . . «, deberá agregar las declaraciones de importación en este archivo. Android Studio puede hacer esto para usted si se mueve el cursor sobre el texto de error en rojo, pulse Alt-Intro , y selecciona Importar . . .

Finalmente

En Android Studio, presione el botón Ejecutar y espere a que se construya el proyecto.

¡Ahora debería ver la transferencia de estilos en su dispositivo!


Android Emulator - Nexus_5X_API_27_x86:5554
StylizeActivity.java
android
Event Log
Gradle Console


256
SAVE

CAPÍTULO 6

Reconocimiento de imágenes TensorFlow

Conoceremos:

Cómo entrenar un modelo de reconocimiento de imágenes personalizado.

Cómo optimizar tu modelo

Cómo comprimir tu modelo

Cómo ejecutarlo en una aplicación de Android prefabricada.

Haremos:

Una aplicación de cámara simple que ejecuta un programa de reconocimiento de imágenes TensorFlow para identificar flores.

Preparar:

La mayoría de este tutorial usará el terminal.

Instalar TensorFlow

Antes de que podamos comenzar con el tutorial, debes instalar tensorflow.

Clona el repositorio de Git:

```
git clone https://github.com/googlecodelabs/tensorflow-for-poets-2
```

Ahora cd en el directorio del clon que acaba de crear. Ahí es donde trabajarás para el resto de este codelab:

```
cd tensorflow-for-poets-2
```

El repositorio contiene tres directorios: android/,scripts/,tf_files/

Checkout branch con los archivos requeridos:

```
git checkout end_of_first_codelab
ls tf_files/
```

Luego, verifique el modelo antes de comenzar a modificarlo.

El directorio scripts/ contiene un simple script de línea de comando, label_image.py para probar la red. Ahora probaremos label_image.py en esta imagen de algunas margaritas:

Ahora prueba el modelo. Si está utilizando una arquitectura diferente, deberá establecer el indicador «–input_size»:

```
python -m scripts.label_image \
  --graph=tf_files/retrained_graph.pb  \
  --image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg
```

El script imprimirá la probabilidad que el modelo ha asignado a cada tipo de flor algo como esto:

```
Evaluation time (1-image): 0.140s

daisy 0.7361
dandelion 0.242222
tulips 0.0185161
roses 0.0031544
sunflowers 8.00981e-06
```

Los dispositivos móviles tienen limitaciones importantes, por lo que vale la pena considerar cualquier procesamiento previo que se pueda hacer para reducir la huella de una aplicación.

Bibliotecas limitadas en dispositivos móviles

Una forma en que la biblioteca de TensorFlow se mantiene pequeña, para dispositivos móviles, solo admite el subconjunto de operaciones que se usan comúnmente durante la inferencia. Este es un enfoque razonable, ya que la capacitación rara vez se lleva a cabo en plataformas móviles. Del mismo modo, también excluye el soporte para operaciones con grandes dependencias externas. Puede ver la lista de operaciones compatibles en el archivo tensorflow/contrib/makefile/tf_op_files.txt.

Por defecto, la mayoría de los gráficos contienen operaciones de entrenamiento que la versión móvil de TensorFlow no admite. . TensorFlow no cargará un gráfico que contenga una operación no admitida (incluso si la operación no admitida es irrelevante para la inferencia).

Optimizar para inferencia

Para evitar problemas causados por operaciones de entrenamiento no compatibles, la instalación de TensorFlow incluye una herramienta optimize_for_inferenceque elimina todos los nodos que no son necesarios para un conjunto determinado de entradas y salidas.

El script también hace algunas otras optimizaciones que ayudan a acelerar el modelo, como la fusión de operaciones explícitas de normalización por lotes en los pesos convolucionales para reducir la cantidad de cálculos. Esto puede dar una velocidad del 30 %, dependiendo del modelo de entrada. Así es como ejecuta el script:

```
      python -m tensorflow.python.tools.optimize_for_inference \
--input=tf_files/retrained_graph.pb \
--output=tf_files/optimized_graph.pb \
--input_names="input" \
--output_names="final_result"
```

La ejecución de este script crea un nuevo archivo en tf_files/optimized_graph.pb.

Verificar el modelo optimizado

Para comprobar que optimize_for_inference no ha alterado la salida de la red, compare la label_imagesalida retrained_graph.pbcon la de optimized_graph.pb:

```
    python -m scripts.label_image \
      --graph=tf_files/retrained_graph.pb\
      --image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg

      python -m scripts.label_image \
--graph=tf_files/optimized_graph.pb \
--image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg
```

Ahora ejecútalo tú mismo para confirmar que ves resultados similares.

Investigue los cambios con TensorBoard

Si siguió el primer tutorial, debería tener un tf_files/training_summaries/directorio (de lo contrario, simplemente cree el directorio emitiendo el siguiente comando Linux:) mkdir tf_files/training_summaries/.

Los siguientes dos comandos matarán cualquier instancia de ejecución de TensorBoard y lanzarán una nueva instancia, en segundo plano mirando ese directorio:

```
pkill -f tensorboard
tensorboard --logdir tf_files/training_summaries &
```

TensorBoard, que se ejecuta en segundo plano, ocasionalmente puede imprimir la siguiente advertencia en su terminal, que puede ignorar de forma segura

WARNING:tensorflow:path ../external/data/plugin/text/runs not found, sending 404.

Ahora agregue sus dos gráficos como registros de TensorBoard:

```
python -m scripts.graph_pb2tb tf_files/training_summaries/retrained \
  tf_files/retrained_graph.pb

python -m scripts.graph_pb2tb tf_files/training_summaries/optimized \
  tf_files/optimized_graph.pb
```

Ahora abra TensorBoard , y vaya a la pestaña «Gráfico». Luego, desde la lista de selección etiquetada como «Ejecutar» en el lado izquierdo, seleccione «Retrained».

Explore el gráfico un poco, luego seleccione «Optimizado» en el menú «Ejecutar».

Desde aquí puede confirmar que algunos nodos se han fusionado para simplificar el gráfico. Puede expandir los distintos bloques haciendo doble clic en ellos.

Explore el gráfico un poco, luego seleccione «Optimizado» en el menú «Ejecutar».

El modelo recapacitado todavía tiene un tamaño de 84MB en este punto. Ese gran tamaño de descarga puede ser un factor limitante para cualquier aplicación que lo incluya.

Cada sistema de distribución de aplicaciones móviles comprime el paquete antes de la distribución. Así que prueba cuánto se puede comprimir el gráfico con el comando gzip:

```
gzip -c tf_files/optimized_graph.pb > tf_files/optimized_graph.pb.gz

gzip -l tf_files/optimized_graph.pb.gz

compressed        uncompressed  ratio uncompressed_name
    5028302             5460013   7.9% tf_files/optimized_graph.pb
```

Por sí solo, la compresión no es de gran ayuda. Esto solo reduce un 8 % el tamaño del modelo. Si está familiarizado con el funcionamiento de las redes neuronales y la compresión, esto no debería sorprender.

La mayor parte del espacio ocupado por el gráfico se basa en los pesos, que son bloques grandes de números de coma flotante. Cada peso tiene un valor de coma flotante ligeramente diferente, con muy poca regularidad.

Pero la compresión funciona explotando la regularidad en los datos, lo que explica la falla aquí.

Ejemplo: Cuantizar una imagen

Las imágenes también pueden considerarse grandes bloques de números. Una técnica simple para comprimir imágenes para reducir el número de colores. Hará lo mismo con los pesos de su red, después de que demuestre el efecto en una imagen.

A continuación he utilizado la utilidad de ImageMagick «convert» para reducir una imagen a 32 colores. Esto reduce el tamaño de la imagen en más de un factor de 5 (png ha incorporado la compresión), pero ha degradado la calidad de la imagen.

Color de 24 bits: 290 KB

32 colores: 55KB

Aplicar un proceso casi idéntico a los pesos de tu red neuronal tiene un efecto similar. Le da mucha más repetición para que el algoritmo de compresión lo aproveche, mientras que reduce la precisión en una pequeña cantidad (típicamente menos de un 1 % de caída en la precisión).

Lo hace sin ningún cambio en la estructura de la red, simplemente cuantifica las constantes en su lugar.

Ahora use quantize_graph secuencia de comandos para aplicar estos cambios:

(Este script es del repositorio de TensorFlow , pero no está incluido en la instalación predeterminada):

```
      python -m scripts.quantize_graph \
--input=tf_files/optimized_graph.pb \
--output=tf_files/rounded_graph.pb \
--output_node_names=final_result \
--mode=weights_rounded
```

Ahora intente comprimir este modelo cuantificado:

```
gzip -c tf_files/rounded_graph.pb > tf_files/rounded_graph.pb.gz

gzip -l tf_files/rounded_graph.pb.gz

  compressed        uncompressed  ratio uncompressed_name
    1633131             5460032  70.1% tf_files/rounded_graph.pb
```

Debería ver una mejora significativa. Obtengo una compresión del 70 % en lugar del 8 % que proporciona gzip para el modelo original.

Ahora, antes de continuar, verifique que el proceso de cuantificación no haya tenido un efecto demasiado negativo en el rendimiento del modelo.

Primero compare manualmente los dos modelos en una imagen de ejemplo.:

```
      python -m scripts.label_image \
        --image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg \
        --graph=tf_files/optimized_graph.pb

        python -m scripts.label_image \
--image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg \
--graph=tf_files/rounded_graph.pb
```

A continuación, verifique el cambio en una porción más grande si los datos para ver cómo afectan el rendimiento general.:

```
Nota: Si comenzó con la end_of_first_codelab, en lugar de trabajar a través de␣
↪`TensorFlow for Poets <https://codelabs.developers.google.com/codelabs/tensorflow-
↪for-poets/index.html>`_, no tendrá el conjunto completo de fotos. La evaluación del␣
↪modelo a continuación fallará. Usted debe:

Pase a la siguiente sección.
Descargue las fotos con el siguiente comando (200MB):
curl http://download.tensorflow.org/example_images/flower_photos.tgz \

| tar xz -C tf_files
```

Primero evalúe el rendimiento del modelo de referencia en el conjunto de validación. Las últimas dos líneas del resultado muestran el rendimiento promedio. Puede tomar uno o dos minutos recuperar los resultados.:

```
python -m scripts.evaluate  tf_files/optimized_graph.pb
```

optimized_graph.pb las puntuaciones tienen una precisión del 90.9 % y 0.270 para el error de entropía cruzada.

Ahora compare eso con el rendimiento del modelo en rounded_graph.pb:

```
python -m scripts.evaluate  tf_files/rounded_graph.pb
```

Debería ver un cambio de menos del 1 % en la precisión del modelo.

Estas diferencias están lejos de ser estadísticamente significativas. El objetivo es simplemente confirmar que este cambio no haya roto claramente el modelo.

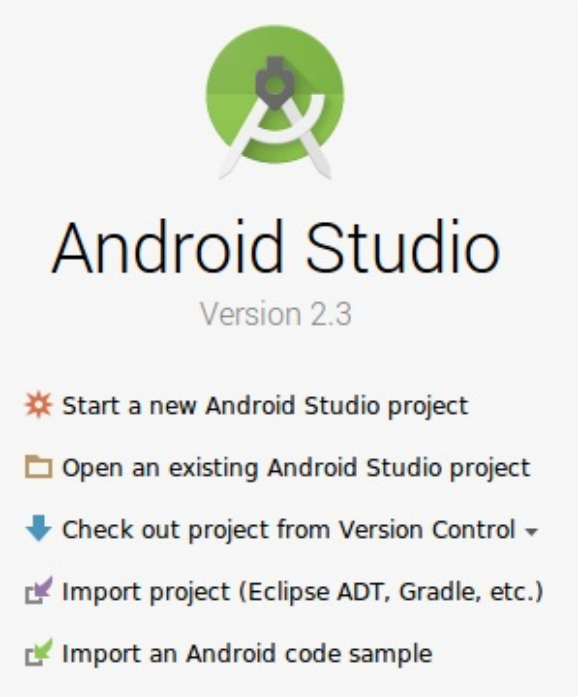
Instalar AndroidStudio

Si aún no lo tiene instalado, vaya a instalar AndroidStudio 3.0+ .

Abra el proyecto con AndroidStudio

Abra un proyecto con AndroidStudio siguiendo estos pasos:

1. Abra AndroidStudio . Después de cargar, seleccione " Abrir un proyecto existente de Android Studio" en esta ventana emergente:



En el selector de archivos, elija tensorflow-for-poets-2/android/tfmobile desde su directorio de trabajo.

Obtendrá una ventana emergente de «Gradle Sync», la primera vez que abre el proyecto, y le pregunta sobre el uso de gradle wrapper. Haga clic en Aceptar».

Prueba ejecutar la aplicación

La aplicación se puede ejecutar en un dispositivo Android real o en el emulador de AndroidStudio.

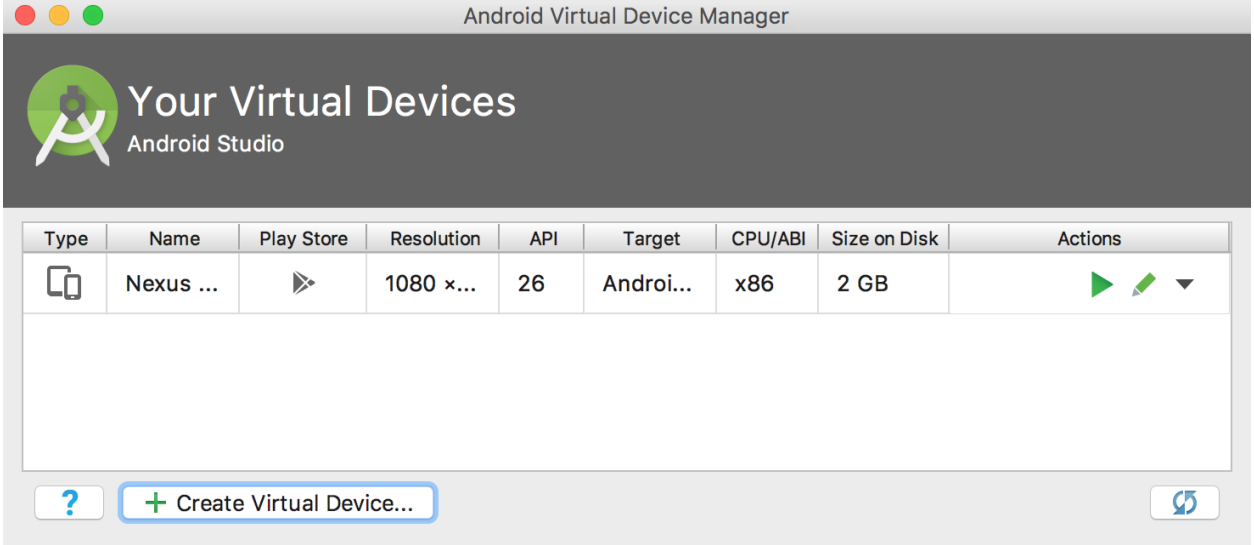
Configura un dispositivo Android

No puede cargar la aplicación de Android Studio en su teléfono a menos que active «modo desarrollador» y «Depuración USB». Este es un proceso de configuración de una sola vez.

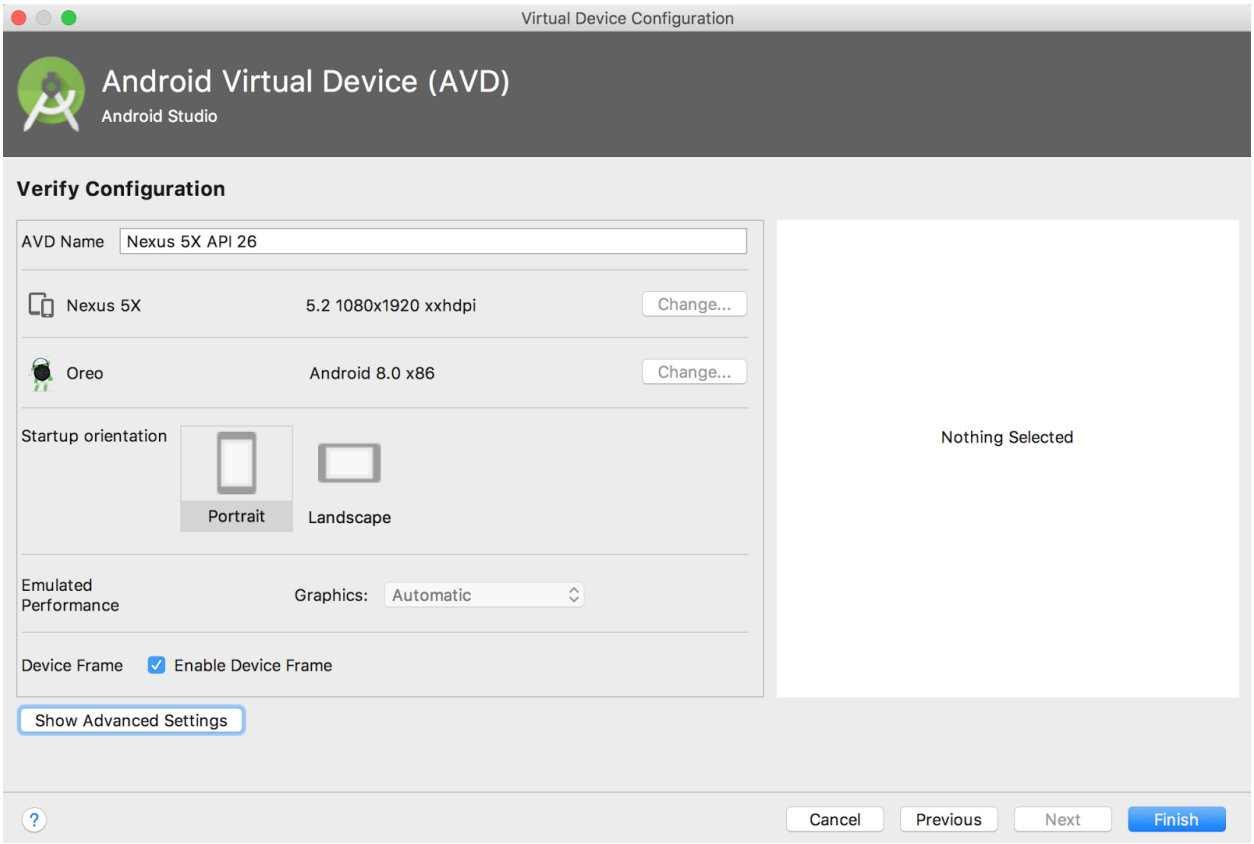
Siga estas instrucciones.

Para hacer esto, necesita crear un nuevo dispositivo en el «Administrador de dispositivos virtuales de Android», al que puede acceder con este botón:

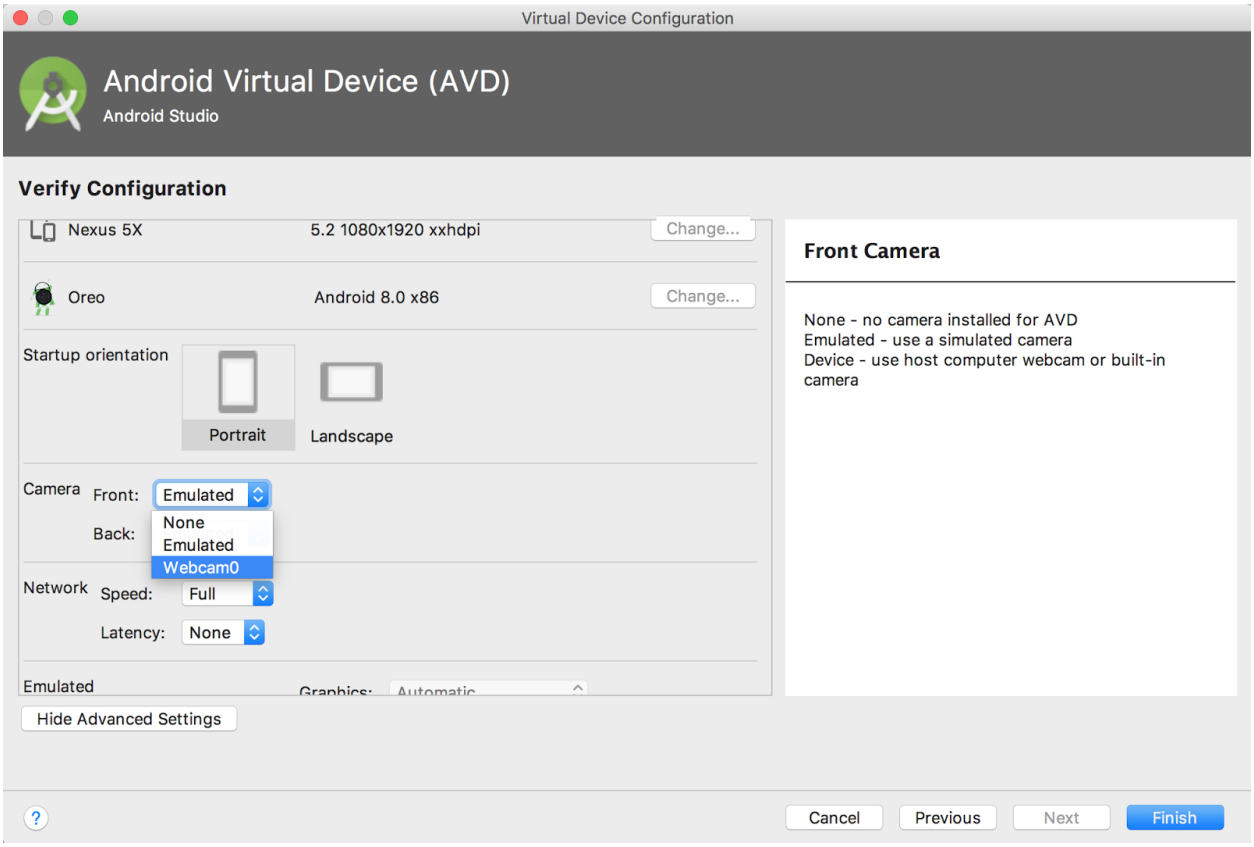
Desde la página principal de ADVM, seleccione «Crear dispositivo virtual»:



Luego, en la página «Verificar configuración», la última página de la configuración del dispositivo virtual, seleccione «Mostrar configuración avanzada»:



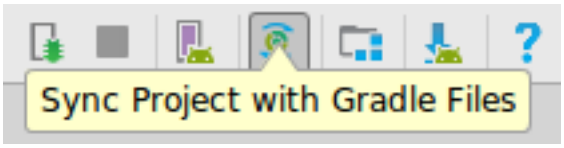
Con la configuración avanzada que se muestra, puede configurar la fuente de la cámara para usar la cámara web de la computadora host:



Prueba Crea e instala la aplicación

Antes de realizar cualquier cambio en la aplicación, ejecutemos la versión que se envía con el repositorio.

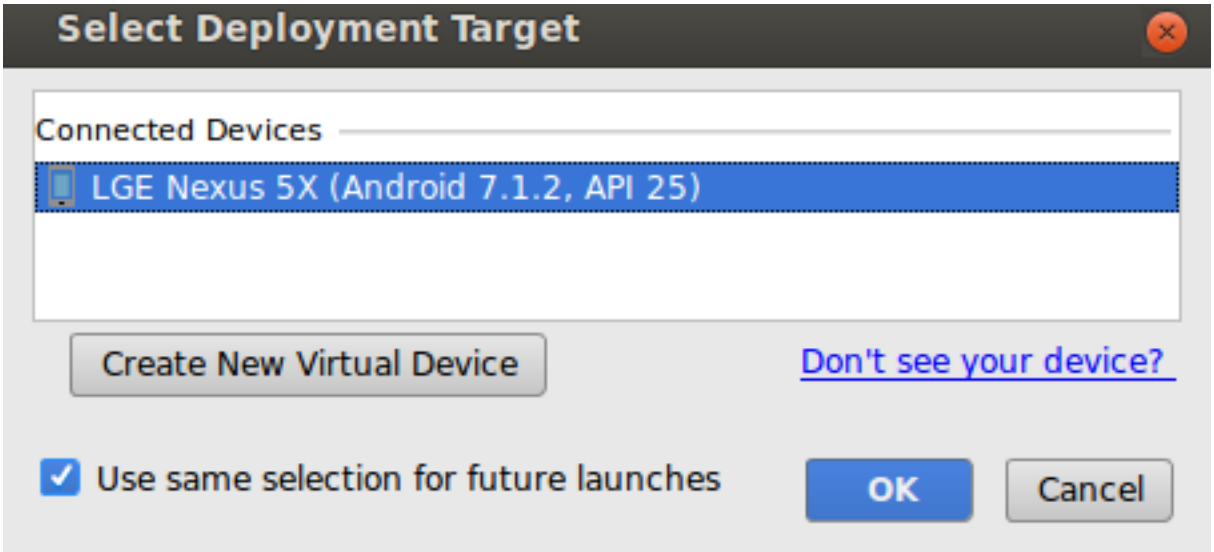
Ejecute una sincronización de Gradle:



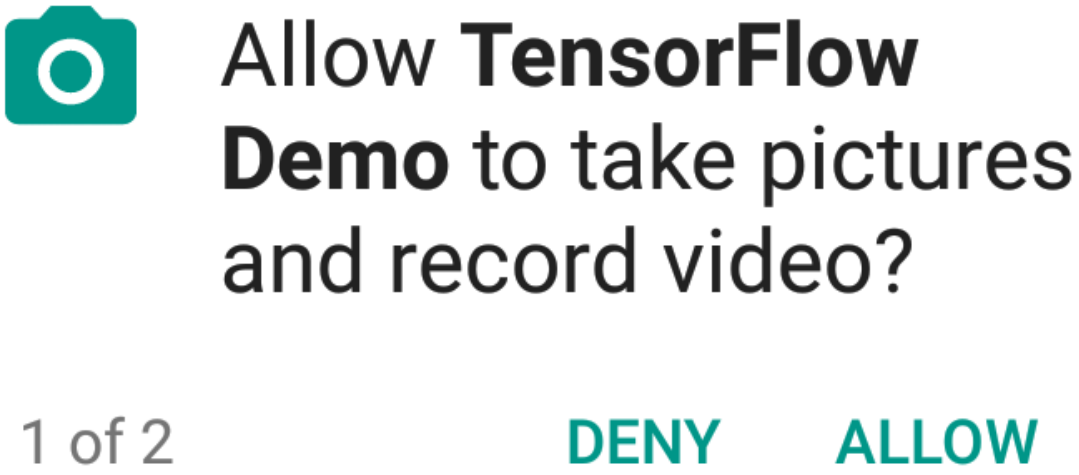
y luego pulse reproducir, en Android Studio para iniciar el proceso de compilación e instalación.

Android Studio puede solicitarle que habilite la ejecución instantánea. Esto no es recomendable ya que aún no es totalmente compatible con el NDK de Android, que se usa para construir las bibliotecas de inferencia de TensorFlow.

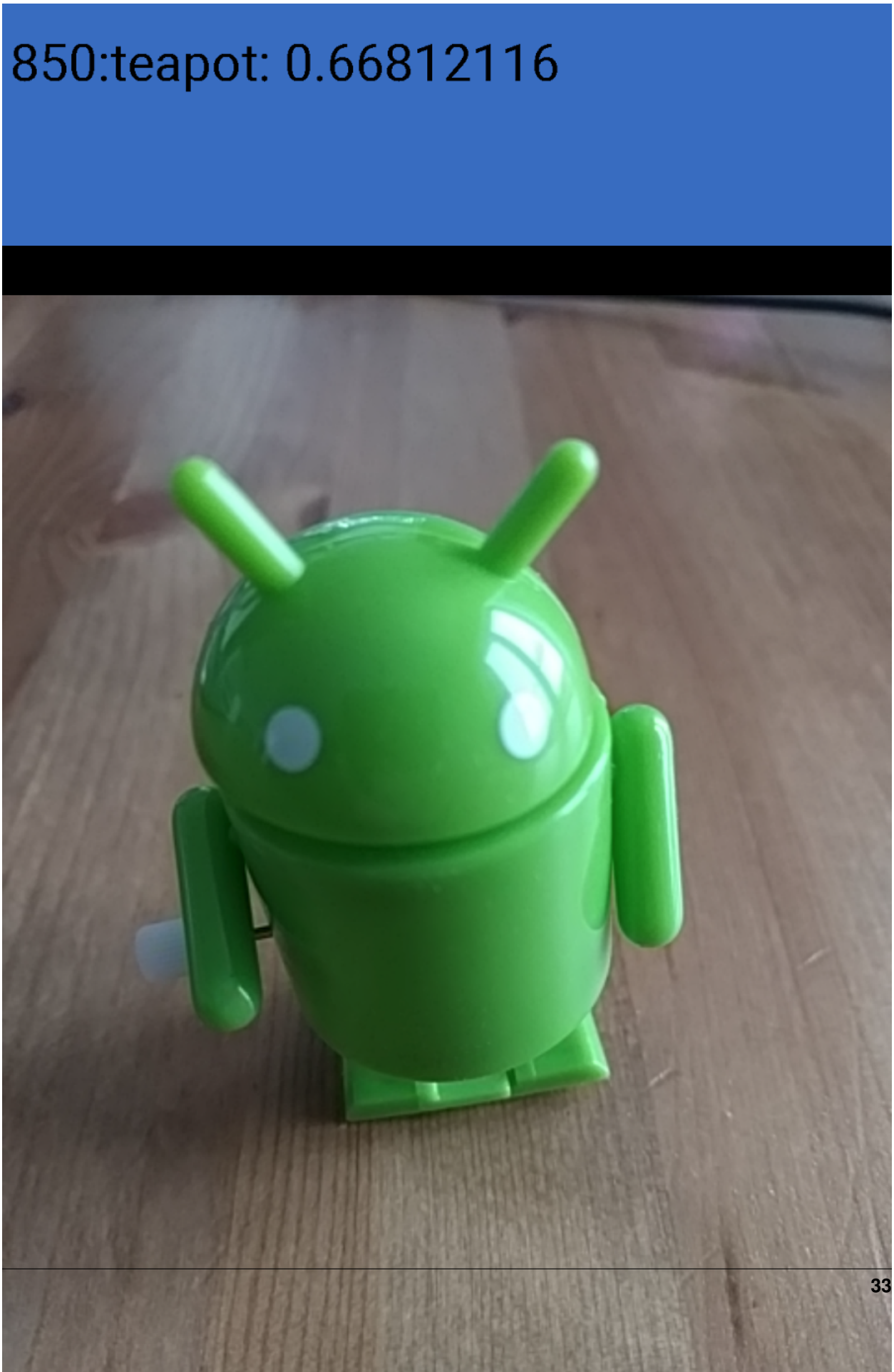
A continuación, deberá seleccionar su teléfono desde esta ventana emergente:



Ahora permite que Tensorflow Demo acceda a tu cámara y a tus archivos:



Ahora que la aplicación está instalada, haga clic en el ícono de la aplicación . image:: img/tf37.png para iniciarla. Esta versión de la aplicación usa MobileNet estándar, entrenado previamente en las categorías de 1000 ImageNet. Debería verse algo como esto («Android» no es una de las categorías disponibles)


850:teapot: 0.66812116

Ejecuta la aplicación personalizada

La configuración predeterminada de la aplicación clasifica las imágenes en una de las 1000 clases de ImageNet, utilizando la red móvil estándar.

Ahora modifiquemos la aplicación para que la aplicación use nuestro morel reciclado para nuestras categorías de imágenes personalizadas.

Agregue sus archivos modelo al proyecto

El proyecto de demostración está configurado para buscar a graph.pby labels.txt archivos en android/tfmobile/assetsdirectorio. Reemplace esos dos archivos con sus versiones. El siguiente comando realiza esta tarea:

cp tf_files/rounded_graph.pb android/tfmobile/assets/graph.pb cp tf_files/retrained_labels.txt android/tfmobile/assets/labels.txt

Cambiar output_name en ClassifierActivity.java

La interfaz TensorFlow utilizada por la aplicación requiere que solicite sus resultados por su nombre. La aplicación está configurada actualmente para leer el resultado de la línea base MobileNet, nombrado «MobilenetV1/Predictions/Softmax». El nodo de salida para nuestro modelo tiene un nombre diferente: «final_result». Abra ClassifierActivity.java y actualice OUTPUT_NAME de la siguiente manera:

ClassifierActivity.java:

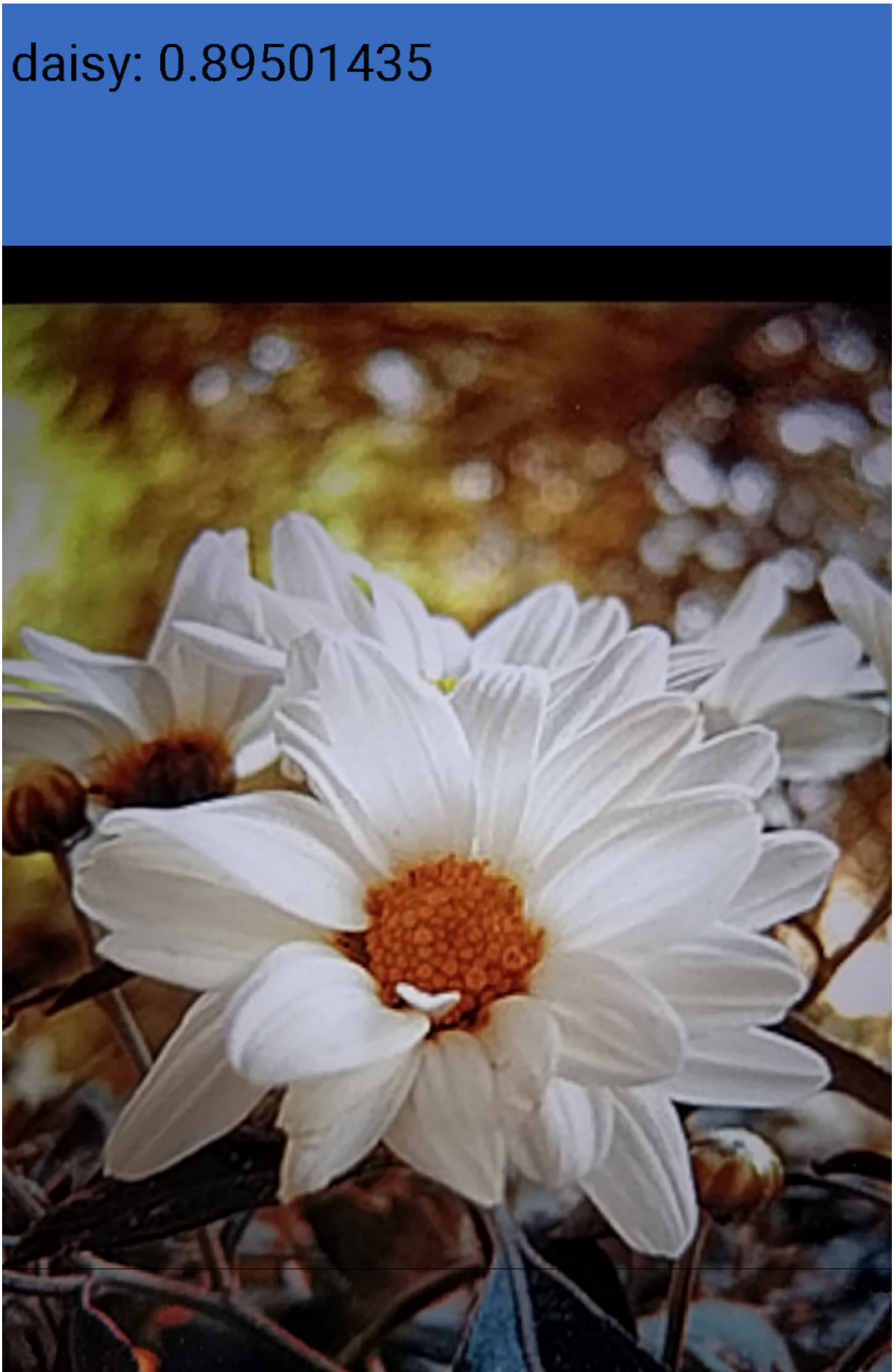
```
private static final String INPUT_NAME = "input";
private static final String OUTPUT_NAME = "final_result";
```

Ejecuta tu aplicación

En Android Studio, ejecute una sincronización de Gradle:

para que el sistema de compilación pueda encontrar sus archivos y luego pulse reproducir.

Debería verse algo como esto:


daisy: 0.89501435

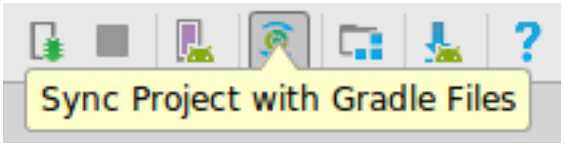
Puede mantener pulsados los botones de encendido y de reducción de volumen para tomar una captura de pantalla.

Ahora intente una búsqueda web de flores, señale la cámara en la pantalla de la computadora y vea si esas imágenes están clasificadas correctamente.

O haga que un amigo le tome una foto y descubra qué tipo de flor es.

Si obtiene un error de sincronización de Gradle:

Es porque Gradle no pudo encontrar android/tfmobile/assets/graph.pb, o android/tfmobile/assets/labels.txt. Verifique las ubicaciones de esos archivos y vuelva a ejecutar la sincronización gradle haciendo clic en el botón «Sincronizar proyecto con archivos Gradle» desde la barra de herramientas:



Ahora que tiene la aplicación ejecutándose, veamos el código específico de TensorFlow.

TensorFlow-Android AAR

Esta aplicación utiliza un archivo Android (AAR) precompilado para sus dependencias TensorFlow. Este AAR está alojado en jcenter. El código para construir el AAR vive en tensorflow.contrib.android.

Las siguientes líneas en el archivo build.gradle incluyen el AAR en el proyecto.

build.gradle:

```
repositories {
   jcenter()
}

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

Uso de la interfaz de inferencia de TensorFlow

El código que interactúa con TensorFlow está contenido en TensorFlowImageClassifier.java.

Crea la interfaz

El primer bloque de interés simplemente crea a TensorFlowInferenceInterface, que carga el TensorFlowgráfico nombrado usando el assetManager.

Esto es similar a a tf.Session(para aquellos familiarizados con TensorFlow en Python).

TensorFlowImageClassifier.java:

```
// load the model into a TensorFlowInferenceInterface.
c.inferenceInterface = new TensorFlowInferenceInterface(
    assetManager, modelFilename);
```

Inspeccione el nodo de salida

Este modelo se puede volver a entrenar con diferentes números de clases de salida. Para garantizar que creamos una matriz de salida con el tamaño correcto, inspeccionamos las operaciones de TensorFlow:

TensorFlowImageClassifier.java:

```
// Get the tensorflow node
final Operation operation = c.inferenceInterface.graphOperation(outputName);

// Inspect its shape
final int numClasses = (int) operation.output(0).shape().size(1);

// Build the output array with the correct size.
c.outputs = new float[numClasses];
```

Alimentar en la entrada

Para ejecutar la red, necesitamos alimentar nuestros datos. Usamos el feedmétodo para eso. Para usar feed,debemos pasar:

el nombre del nodo para alimentar los datos

los datos para poner en ese nodo

la forma de los datos

Las siguientes líneas ejecutan el método de alimentación.

TensorFlowImageClassifier.java:

```
inferenceInterface.feed(
    inputName,   // The name of the node to feed.
    floatValues, // The array to feed
    1, inputSize, inputSize, 3 ); // The shape of the array
```

Ejecute el cálculo

Ahora que las entradas están en su lugar, podemos ejecutar el cálculo.

Tenga en cuenta que este run, método toma una matriz de nombres de salida porque es posible que desee extraer más de una salida. También acepta un indicador booleano para controlar el registro.

TensorFlowImageClassifier.java.

```
inferenceInterface.run(
    outputNames, // Names of all the nodes to calculate.
    logStats);   // Bool, enable stat logging.
```

Obtener la salida

Ahora que la salida se ha calculado, podemos sacarla del modelo en una variable local. La outputsmatriz aquí es la que dimensionamos mediante la inspección de la salida Operationanterior.

Llame a este método de búsqueda una vez por cada salida que desee recuperar.

TensorFlowImageClassifier.java:

```
inferenceInterface.fetch(
    outputName,  // Fetch this output.
    outputs);    // Into the prepared array.
```

¿Qué sigue?

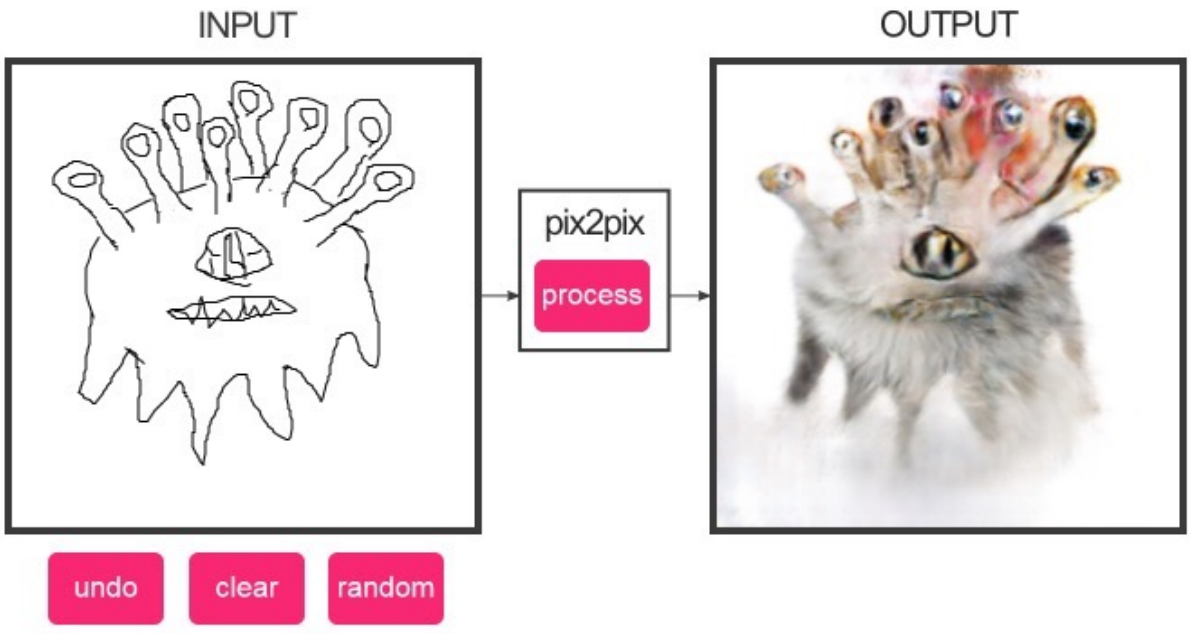
Hay muchas opciones:

Revise otros ejemplos de mobile-tensorflow . Los otros ejemplos de Android incluyen aplicaciones que hacen estilización de imagen y detección de peatones . El ejemplo de estilización también está disponible como un codelab . Si desea obtener más información sobre TensorFlow en general consulte cómo comenzar.

CAPÍTULO 7

Demostración de imagen a imagen

El modelo pix2pix funciona entrenando en pares de imágenes, como la construcción de etiquetas de fachada para construir fachadas, y luego intenta generar la imagen de salida correspondiente desde cualquier imagen de entrada que le proporcione. La idea es directamente del papel pix2pix , que es una buena lectura.



Requisitos previos:

Tensorflow 1.0.0

Recomendado:

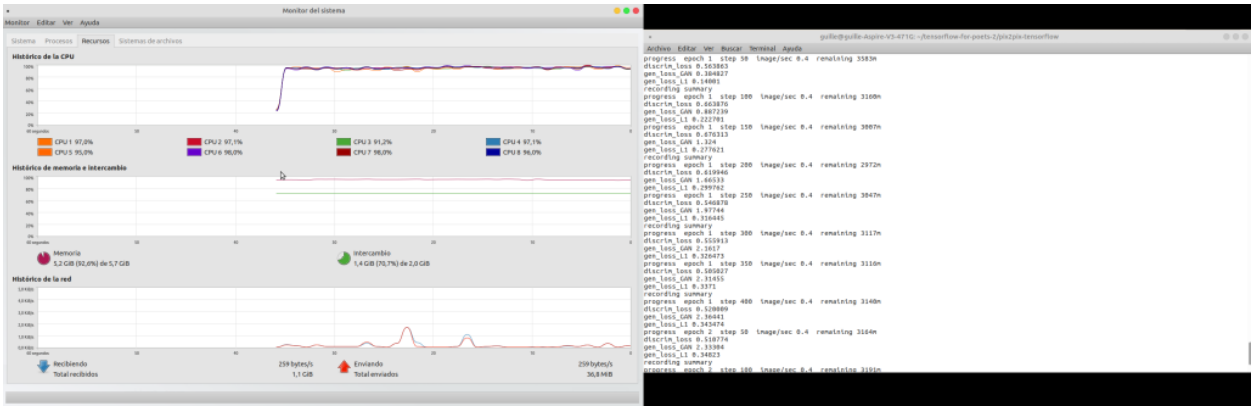
Linux con Tensorflow GPU edition + cuDNN

Instalación tensorflow.

cuDNN para mayor performance.

Empezando:

```
# clonar este repositorio
git clone https://github.com/affinelayer/pix2pix-tensorflow.git
cd pix2pix-tensorflow
 # descarga el conjunto de datos de fachadas CMP (generado a partir de http://cmp.
↪felk.cvut.cz/~tylecr1/facade/)
python tools/download-dataset.py facades

# entrenar el modelo (esto puede demorar 1-8 horas dependiendo de la GPU, en la CPU␣
↪estará esperando un poco)

python pix2pix.py \
  --mode train \
  --output_dir facades_train \
  --max_epochs 200 \
  --input_dir facades/train \
  --which_direction BtoA

# prueba el modelo

python pix2pix.py \
  --mode test \
  --output_dir facades_test \
  --input_dir facades/val \
  --checkpoint facades_train
```



La ejecución de prueba generará un archivo HTML en el facades_test/index.html que se muestran los conjuntos de imágenes de entrada / salida / destino.

Si tiene instalado Docker, puede usar la imagen proporcionada de Docker para ejecutar pix2pix sin instalar la versión correcta de Tensorflow:

```
# entrenar el modelo
python tools/dockrun.py python pix2pix.py \
      --mode train \
      --output_dir facades_train \
      --max_epochs 200 \
      --input_dir facades/train \
      --which_direction BtoA
# prueba el modelo
```

```
python tools/dockrun.py python pix2pix.py \
      --mode test \
      --output_dir facades_test \
      --input_dir facades/val \
      --checkpoint facades_train
```

Más Información.

CAPÍTULO 8

Agradecimientos y links de interés

Agradecimientos a los autores de las siguientes publicaciones por aportar a la academia y al conocimiento de las tecnologias.

TensorFlow Tutorial para principiantes

What is the TensorFlow machine intelligence platform?

Image-to-Image Demo

Conceptos básicos tensorflow ppt

Reconocimiento de imágenes TensorFlow

Artistic style transfer

Links de interés

Tensorflow Getting Started

¿Cuáles son algunas de las mejores aplicaciones que usan TensorFlow de Google?

Tensorflow Tutorials using Jupyter Notebook

TensorFlow Serving es una biblioteca de software de código abierto para servir modelos de aprendizaje automático

Ejecutando sus modelos en producción con TensorFlow Serving

PyGamePlayer

TensorKart

Alojar un modelo de Keras con el Backend de Tensorflow en Azure App Services

Redes neuronales

¡Usando TensorFlow para clasificar a los perritos calientes!

(EN DESARROLLO) Tensile extensión de PHP le proporciona un sistema orientado a objetos para utilizar Tensorflow Machine Learning de Google (http://tensorflow.org) desde sus aplicaciones PHP. Todavía no está en estado completo funcionando y compilaciones experimentales.

Deep Text Corrector usa TensorFlow para entrenar modelos de secuencia a secuencia que son capaces de corregir automáticamente pequeños errores gramaticales en el inglés escrito de conversación

Aplicación Mini AI con TensorFlow y Shiny

Creating REST API for TensorFlow models

Inmersión profunda en la detección de objetos con imágenes abiertas, utilizando Tensorflow

Usar TensorFlowSharp en Unity (Experimental)

Tensorflow Object Detection API

DevDocs combina múltiples documentos de API en una interfaz rápida, organizada y con capacidad de búsqueda.

Node.js meets OpenCV's Deep Neural Networks—Fun with Tensorflow and Caffe

Proyector tensorflow

Interactive Abstract Pattern Generation Javascript Demo

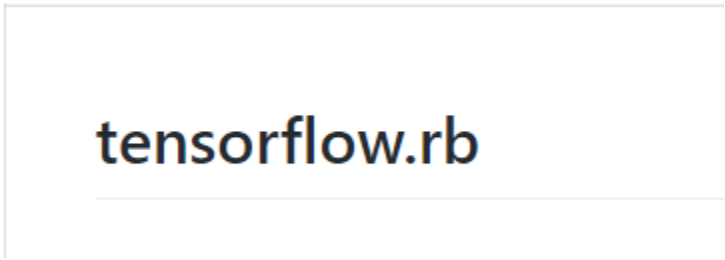
Fast PixelCNN++: speedy image generation

Recopilación realizada por Guillermo Lemunao.

CAPÍTULO 9

Otros

API de Ruby para utilizar TensorFlow.



TensorFlow Node.js JavaScript y una API de alto nivel para los usuarios de Node. js.

Repos divertidos:

https://github.com/kevinhughes27/TensorKart

- https://github.com/DanielSlater/PyGamePlayer

Games

- Pong
- Tetris
- Mini Pong - modified version of pong to run in lower resolutions
- Half Pong - simplified version of pong with just one bar