**Tarea 5 – Artificial Intelligence, Machine Learning y Deep Learning**

**Diagnóstico de Cancer de Piel**



Escena de la Serie “Silicon Valley, S04E04”

Jian-Yang muestra su app de “Hot-Dog / Not Hot-Dog”

En resumen la inteligencia artificial es la capacidad de una máquina a tomar decisiones basada en modelos simulando la inteligencia humana.

Esto nos lleva a aprendizaje de máquina que es una parte de la inteligencia artificial en donde estudia técnicas estadísticas para dar esta habilidad de pensamiento a la máquina sin necesidad de ser programada. Para aprender, la máquina trata de predecir modelos basados en información (texto, imágenes, datos).

Desde hace unos años hacia acá se han utilizado técnicas de deep learning que es una sub parte de machine learning y utiliza redes neuronales para tratar de realizar el aprendizaje. A pesar de que las redes neuronales se hace analogía con las neuronas en el humano, el método de aprendizaje no es el mismo al que tiene el humano, es decir, en pocas palabras, deep learning es diferente a la manera de aplicar en neurociencia.

Podemos utilizar estas áreas para diferentes objetivos, las aplicaciones son muchas:

* Analizar datos para predecir un valor futuro
* Realizar estimaciones de cuando hacer mantenimiento
* Sistema de reconocimiento facial para acceso
* **Reconocedor de objetos**
* Hacer un chatbot (para servicio al cliente)
* Realizar que los robots aprendan tareas sin entrenarlos (una área de la robótica en actual estudio es Reinforcement Learning, ver el canal de OpenAI en Youtube)
* Vencer a los mejores jugadores de eSport, juegos en línea como DOTA
* **Diagnóstico de Imágenes médicas**

**Comprobación de Python 3.x y Dlib**

1.1 – Instale la máquina virtual que se le suministro con Oracle VirtualBox

1.2 – Abrir una terminal y ejecute el siguiente comando

workon facecourse-py3

1.3 – Una vez ejecutado navegar a la carpeta /home/ubuntu/cv4faces/week0/python

1.4 – Ejecutar: python facialLandmarkDetector.py

1.5 – Debe de observarse la imagen de una joven con la detección facial de los puntos más importantes de su cara

1.6 – Presione cualquier tecla sobre la imagen para cerrar la aplicación

1.7 – Salga del ambiente virtual

deactivate

**Instalando grupo de librerías necesarias**

*Nota: No instalaremos librerías que usan el GPU para que cualquier persona pueda realizar las tareas, usaremos el CPU que es más lento pero funcionará para todos los casos. Si desea instalar soporte de GPU debe seguir los pasos listados en la sección de Soporte de GPU****.***

2.1 – Activar el ambiente virtual

workon facecourse-py3

2.2.A – CPU Support: Instale las siguientes librerías

pip install --upgrade pip

pip install tensorflow tensorflow\_hub jupyter notebook

pip install –upgrade “tensorflow==1.7.\*”

sudo apt install curl

2.2.B – Soporte de GPU (saltar esta sección si no planea utilizar soporte de GPU)

2.2.1 Instalar lo siguiente si no posee nada instalado (obviar 2.1 para la VM actual)

sudo apt-get -y update

sudo apt-get -y upgrade

sudo apt-get -y install python-pip python-dev

sudo apt-get -y install -y python-numpy python-scipy ipython python-matplotlib python-sklearn python-wheel

sudo pip install --upgrade pip

2.2.2 Verificar si su tarjeta soporta CUDA (Compute Unified Device Architecture)

<https://developer.nvidia.com/cuda-gpus>

2.2.3 Instalación de CUDA

sudo apt-get install -y linux-image-extra-virtual linux-headers-`uname -r` linux-image-`uname -r`

wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86\_64/cuda-repo-ubuntu1604\_8.0.61-1\_amd64.deb

sudo dpkg -i cuda-repo-ubuntu1604\_8.0.61-1\_amd64.deb

sudo apt-get -y update

sudo apt-get -y upgrade

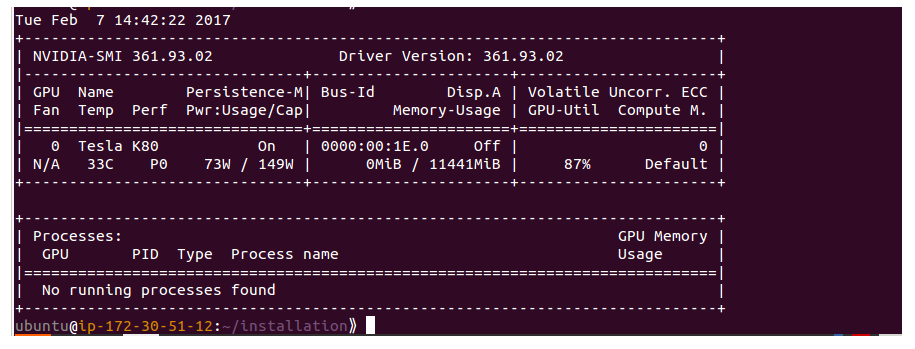
sudo apt-get install -y cuda

sudo sh -c "sudo echo '/usr/local/cuda/lib64' > /etc/ld.so.conf.d/cuda.conf"

sudo ldconfig

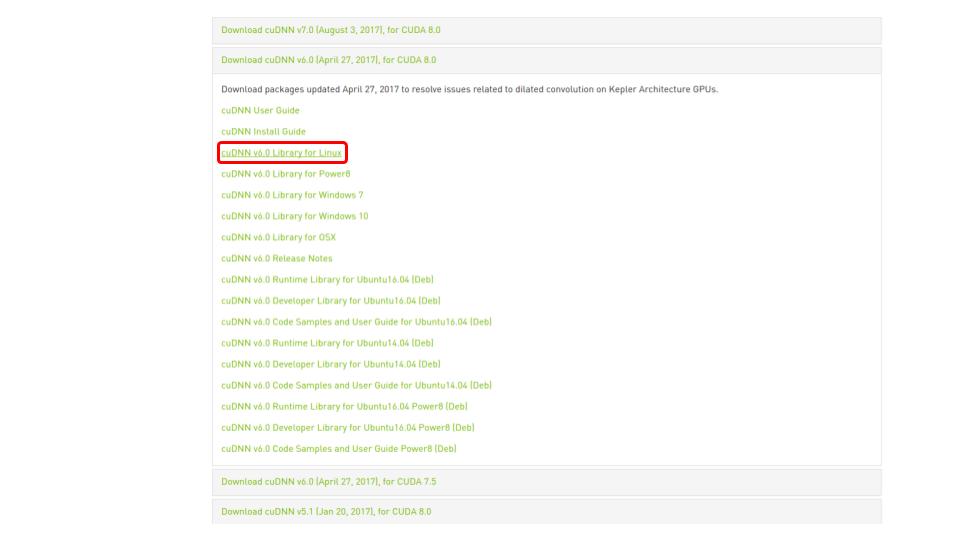
2.2.4 Verificar si CUDA está instalado o no

nvidia-smi



2.2.5 Instalación de Cudnn

Ir al sitio de NVIDIA, registrarse, contester preguntas y utilizar la version de Cudnn v6.0 para Linux y Cuda 8.



Si utiliza AWS (Amazon Web Services) necesita subir cudnn-8.0-linux-x64-v6.0.tgz usando rsync o scp siguiendo los siguientes pasos:

tar -zxf cudnn-8.0-linux-x64-v6.0.tgz

sudo cp -P cuda/lib64/\* /usr/local/cuda/lib64/

sudo cp cuda/include/\* /usr/local/cuda/include/

Añadiendo también al ~/.bashrc

vi ~/.bashrc

Export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64"

export CUDA\_HOME=/usr/local/cuda

Ahora corra el comando .bashrc y haga que los cambios se ejecuten en la sección actual

Source ~/.bashrc

Ahora con el sistema corriendo instalaremos la version de GPU de Tensorflow

2.2.6 – Activar el ambiente si no está active

workon facecourse-py3

2.2.7 Instalar Tensorflow GPU

pip install tensorflow-gpu==1.4

**Probando la instalación de Tensorflow**

3.1 En la ventana de commandos escribir:

python

>>>import tensorflow as tf

>>>tf.\_\_version\_\_

>>>'1.4.0' #For GPU version

>>>'1.5.0' #For CPU version

import tensorflow as tf

a=tf.Variable(1.0)

b=a+3

with tf.Session() as sess:

sess.run(tf.global\_variables\_initializer())

print(sess.run(b))

Resultado:

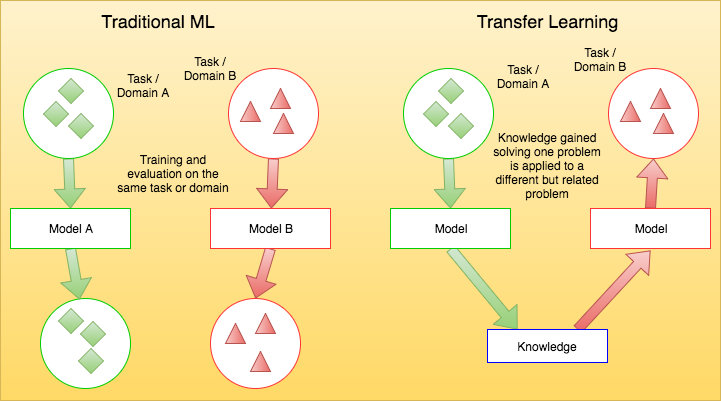
tensorflow/core/common\_runtime/gpu/gpu\_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: Tesla K80, pci bus id: 0000:00:1e.0)

4.0

**Introducción Rápida a un Clasificador de Objetos via Transfer Learning**

Transfer Learning es una técnica para generación de modelos generalmente utilizada cuando se tiene un dataset pequeño (nuestro caso) y un CPU en vez de un GPU (para demorar menos tiempo entrenando al modelo).

Más conciso, para entender que es transfer learning imagínese que el professor dicta una clase a sus estudiantes y el conocimiento previo del professor es adquirido por los estudiantes sin necesidad de pasar por el entrenamiento que tuvo el (ver figura inferior).



Los pasos para realizar transfer learning son:

* Preparar el dataset
* Cargar el modelo pre-entrenado
* Re-entrenar la capa final del modelo

**Copiando archivos de dataset, Tensorflow, Tensorflow\_Hub y preparando entrenamiento**

* 1. – Comparta la carpeta en el sistema y montela

4.2 – navegue a: cd /home/ubuntu

4.3 – cree una carpeta: mkdir <nombre\_de\_la\_carpeta\_en\_ubuntu>

ejm: mkdir sf\_Downloads

* 1. – monte la carpeta e ingrese el password

sudo mount –t vboxsf <nombre\_de\_la\_carpeta\_en\_windows\_donde\_esta\_la\_data> /home/ubuntu/<nombre\_de\_la\_carpeta\_en\_ubuntu>

ejm: sudo mount –t vboxsf Downloads /home/ubuntu/sf\_Downloads

* 1. – En mi caso, crearé una carpeta de proyecto en /home/ubuntu/ llamada skin\_cancer\_prj

mkdir skin\_cancer\_prj

* 1. – Descargue los archivos de imágenes del proyecto y ejemplos de tensorflow para el proyecto.

cd <carpeta\_a\_cambiar>

cp <path\_de\_archivos\_en\_carpeta\_compartida> <path\_del\_proyecto\_en\_linux>

ejm:

cd skin\_cancer\_prj

cp /home/ubuntu/sf\_Downloads/section\_02 .

* 1. – Descargue el repositorio de Tensorflow

git clone https://github.com/tensorflow/tensorflow

* 1. – Descargue el archivo de re-entrenamiento de tensorflow

git clone https://github.com/tensorflow/hub/

4.9 – Ajuste el archivo train.sh, que es un Shell script (para ejecutar automáticamente los pasos). El archivo train.sh está en:

cd /home/ubuntu/skin\_cancer\_prj/section\_2/01\_skin\_cancer\_classification

4.10 - Modifique el archivo como sigue:

cd hub

python examples/image\_retraining/retrain.py \

--bottleneck\_dir=../bottlenecks \

--how\_many\_training\_steps 4000 \

--model\_dir=../inception \

--output\_graph=../retrained\_graph.pb \

--output\_labels=../retrained\_labels.txt \

--summaries\_dir=../retrain\_logs \

--image\_dir ../section\_2/01\_skin\_cancer\_classification/data

**Previo a Entrenar - ¿Qué simbolizan estos parámetros?**

retrain.py: Es el archivo provisto por google para realizar transfer learning en sus proyectos. A pesar de que esta fue entrenado con imágenes que no son de detección de cáncer, como cualquier red neuronal, el sistema aprendió a clasificar imágenes y utilizaremos estos como pesos iniciales.

--bottleneck\_dir: bottleneck es la técnica de solamente utilizar todas las capas anteriores menos la última. ¿Porqué? Porque esta capa es utilizada para la clasificación de nuestras imágenes (que tipo de cáncer es, en este caso).

--how\_many\_training\_steps: es la cantidad de pasos a seguir, por defecto es 4000, que es el mismo numero que utilizamos

--model\_dir: especifica el modelo a utilizar, existen varios, PASCAL, COCO, VGG16, VGG19, Inception, Xception, etc. Cada uno con ventajas y desventajas, p.e. tiempo de entrenamiento y manera de detección

--output\_graph: gráfica de tensorboard que es la manera visual de ver la red

--output\_labels: como clasifica nuestras imagenes cancerígenas, son las etiquetas de tipos de melanoma, que en este caso, tenemos solo 3)

--summaries\_dir: información del entrenamiento

--image\_dir: donde se encuentran las imágenes para entrenar nuestro detector de objetos

Puede revisar el archivo retrain.py para que observe que otros argumentos además de estos puede alterar para el ajuste de entrenamiento de parámetros.

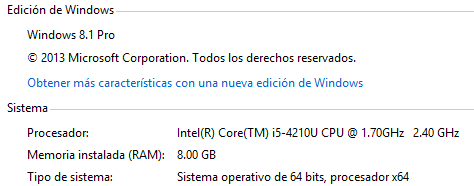
**Transfer Learning - Listos para reentrenar nuestro multi-clasificador de objetos!**

5.1 – Ejecute el siguiente comando en la carpeta de skin\_cancer\_prj:

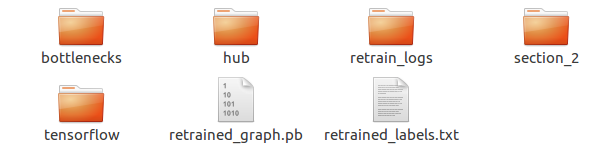
./section\_2/01\_skin\_cancer\_classification/train.sh

5.2 – El tiempo de entrenamiento en mi máquina virtual con máquina principal en Windows fue de “1 hora” aproximadamente con las siguientes especificaciones:





5.3 – Debe haber creado varias carpetas como sigue:



**Transfer Learning - Listos para verificar que tipo de cáncer es!**

6.1 – Abrir el archivo test.sh, modificarlo como sigue y salvar

cd tensorflow

python tensorflow/examples/label\_image/label\_image.py \

--graph=../retrained\_graph.pb \

--labels=../retrained\_labels.txt \

--input\_layer=Placeholder \

--output\_layer=final\_result \

--input\_mean=128 --input\_std=128 \

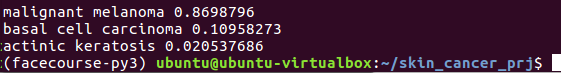
--image=../<nombre\_de\_la\_imagen>.jpg

6.2 – Ahora arrojar en la carpeta skin\_cancer\_prj una imagen de extensión .jpg que corresponda a un cáncer de los tres tipos

6.3 – Ejecutar el siguiente comando para predecir que tipo de cáncer es:

./section\_2/01\_skin\_cancer\_classification/test.sh

6.4 – Repita el paso con diferentes imágenes, por ejemplo, lunares, observará que detecta como tipo de cáncer, esto es, debido a que debemos tomar las nuevas imágenes de lunares y crearlas por ejemplo en la carpeta data/lunares y arrojar todos los lunares para volver a reentrenar.

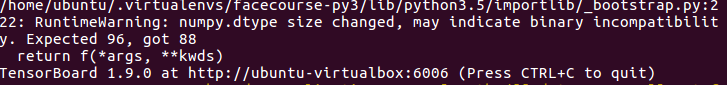


**Opcional - Tensorflow Graph – Un vistazo a la herramienta visual para observar el flujo y variables de la red neuronal.**

7.1 – Ejecute en la carpeta de skin\_cancer\_prj el comando siguiente:

python -m tensorboard.main --logdir ./retrain\_logs/

7.2 – La salida esperada debe ser:

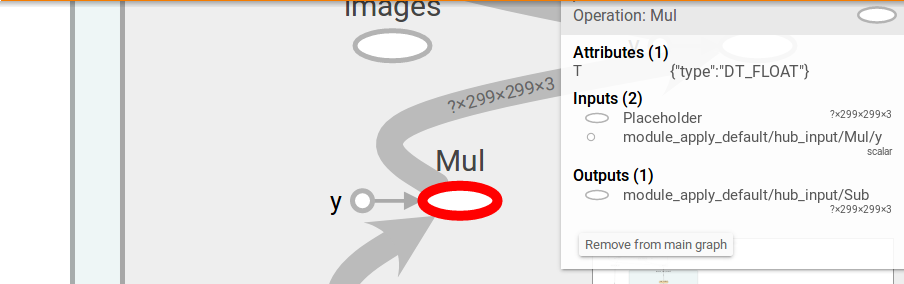


7.3 – Abrir un navegador web e insertar la dirección que describe Tensorboard, en mi caso:

http://ubuntu-virtualbox:6006/

7.4 – Ahora navegue por las ventanas de SCALAR y GRAPHS

En especial, esta herramienta me sirvió para detectar un problema que había con el archivo original de test.sh, en el cual utilizaba Mul (se debe investigar más el porque no funciona con Mul) como entrada, pero en realidad admite Placeholder como entrada



7.5 – Para detener Tensorboard usar CTRL+C sobre la ventana donde está iniciado el servidor.

**Reentrenamiento y Prueba del Modelo Utilizando MobileNet**

El modelo anterior funciona bien para un computador. Debido a que este modelo pesa más de 80MB no es tan provechoso utilizarlo en dispositivos móviles.

Utilizaremos el mismo proceso de reentrenamiento pero con el modelo MobileNet.

8.1 – Creamos una carpeta llamada mobilenet en /home/ubuntu/skin\_cancer\_prj/

8.2 – Creamos un nuevo archivo trainmobilenet.sh dentro del directorio

/home/ubuntu/skin\_cancer\_prj/section\_2/01\_skin\_cancer\_classification

Insertamos lo siguiente en el archivo:

cd hub

python examples/image\_retraining/retrain.py \

--how\_many\_training\_steps 4000 \

--bottleneck\_dir=../mobilenet/cancer\_bottlenecks\_mobilenet \

--tfhub\_module https://tfhub.dev/google/imagenet/mobilenet\_v1\_100\_224/feature\_vector/1 \

--output\_graph=../mobilenet/cancer\_retrained\_graph\_mobilenet.pb \

--output\_labels=../mobilenet/cancer\_retrained\_labels\_mobilenet.txt \

--summaries\_dir=../mobilenet/cancer\_retrain\_logs\_mobilenet \

--image\_dir=../section\_2/01\_skin\_cancer\_classification/data

8.3 – Relizamos el entrenamiento ubicados en la carpeta /home/ubuntu/skin\_cancer\_prj/

./section\_2/01\_skin\_cancer\_classification/trainmobilenet.sh

8.4 – Luego de entrenado creamos el archivo testmobilenet.sh

cd tensorflow

python –W ignore tensorflow/examples/label\_image/label\_image.py \

--graph=../mobilenet/cancer\_retrained\_graph\_mobilenet.pb \

--labels=../mobilenet/cancer\_retrained\_labels\_mobilenet.txt \

--input\_layer=Placeholder \

--output\_layer=final\_result \

--input\_mean=128 \

--input\_std=128 \

--input\_width=224 \

--input\_height=224 \

--image=../$1

8.5 - Note algo extraño, el argumento --image posee un $1, esto es una manera de llamar al argumento de entrada, en la siguiente sección se verá el ejemplo.

8.6 - Ejecutar el test de nuestros modelo con imágenes de enfermedades bajadas de internet y ejecutar el siguiente comando en la carpeta skin\_cancer\_prj

./section\_2/01\_skin\_cancer\_classification/testmobilenet.sh carcinoma.jpg

Observe que el declarer la imagen como $1 nos permite ingresar al shell script un argumento.

8.7 – Repita la operación para diferentes imagenes

**Tensorflow Mobile - Corriendo su aplicación en un dispositivo Android**

Existen dos maneras de realizar el reconocedor de objetos, una es con Tensorflow Lite y la que vamos a explorar es utilizando Tensorflow Mobile. Debido a que el modelo tardaría más en detectar que tipo de objeto (cáncer) es, hemos de usar Mobilenet que es optimizado y mejorado en precisión para dispositivos móviles.

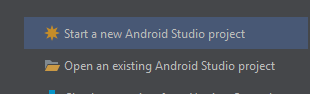
9.1 – Descargue e instale Android Studio en Windows

9.2 – Descargue e instale Git

9.2 – Descargue tensorflow de github en Windows con git

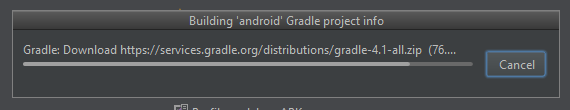
git clone http://www.github.com/tensorflow/tensorflow.git

9.3 – Navegue hasta la ventana de abrir un proyecto existente



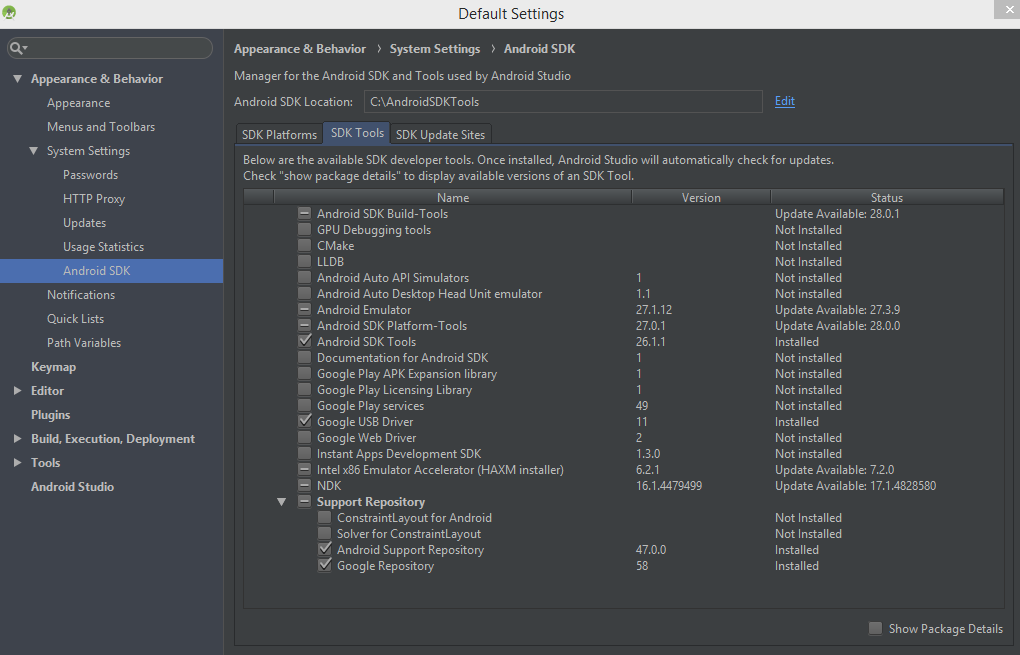
9.4 - Buscar el proyecto de ejemplo de Android ubicado en <PATH\_DE\_DESCARGA\_DE\_TENSORFLOW\_EN\_WINDOWS>\tensorflow\tensorflow\examples\android

9.5 - Esperar que cargue gradle

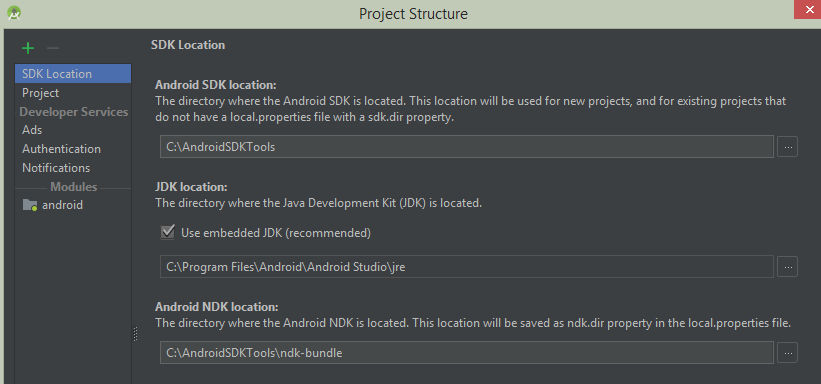


9.6 - Resuelva todos los problemas y dependencias que se le presenten

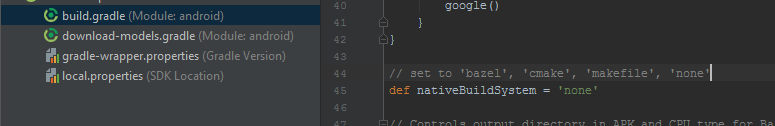
9.7 - Vaya a Tools/SDK Manager y verifique que tiene lo siguiente instalado



9.8 - Vaya a File/Project Structure y verifique lo siguiente



9.9 - Cambiar en build.gradle la línea def natifeBuildSystem a none

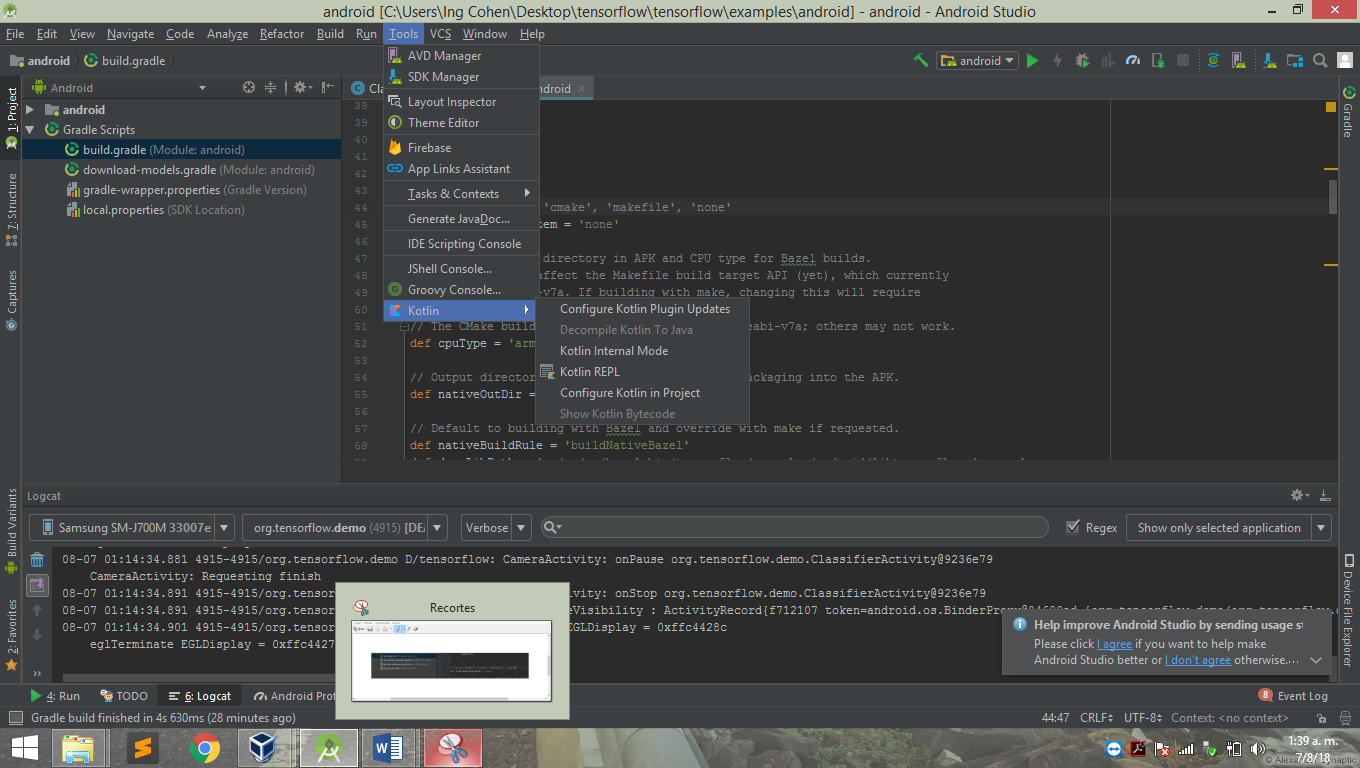


9.10 – Salve y esperar a que gradle realice sus operaciones

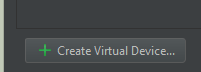
9.11 – Existen dos maneras de ejecutar la aplicación, una es creando un dispositivo virtual de Android si no posee uno y la otra es utilizando un dispositivo real. Nuestro objetivo es utilizar un dispositivo real pero explicaremos el paso también de como crear un dispositivo virtual. Puede saltar el paso 9.12 para ejecutar en un dispositvo Android real (explicado desde la sección 9.13)

9.12 - Crear un dispositivo virtual

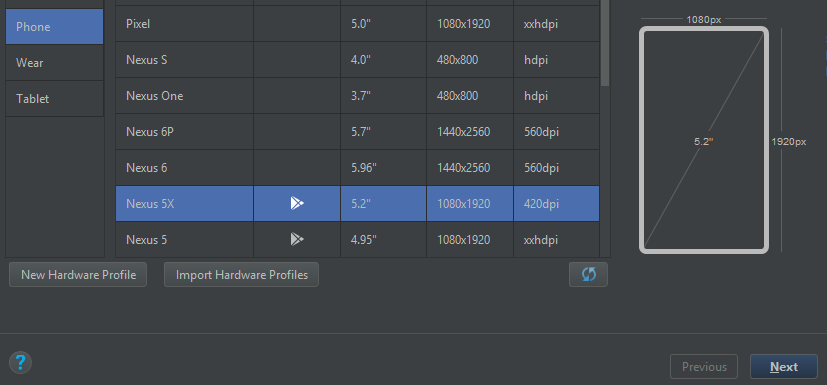
9.12.1 – Ir a Tools | AVD Manager



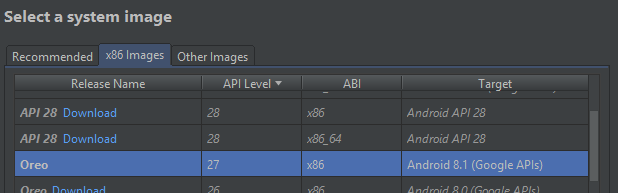
9.12.2 – Hacer clic en Crear dispositivo virtual



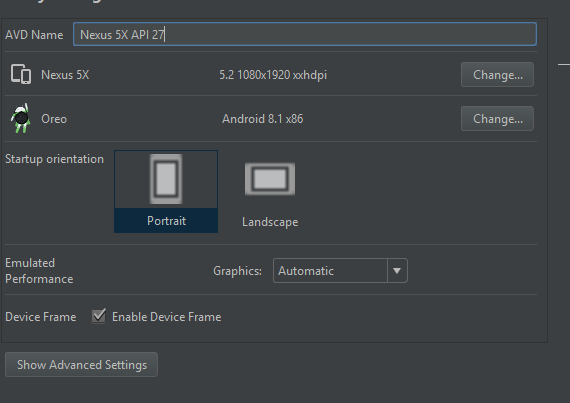
9.12.3 – Elija alguno de los dispositivos



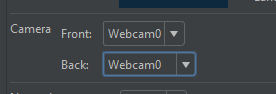
9.12.4 – Seleccione de la pestaña x86 la imagen de Oreo, en este caso.



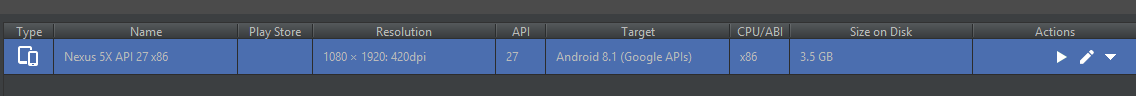
9.12.5 – En la siguiente ventana de Verificación de Configuración hacer click en “Show Advanced Settings”



9.12.6 – Seleccione la cámara como webcam0 si posee webcam y hacer click en finalizar



9.12.6 – Puede observar el dispositivo virtual ya creado



9.13 – Habilite las opciones del desarrollador de Android en su dispositivo real como sigue:

9.13.1 – Dirigirse a la sección de “Settings”

9.13.2 – Ir a “About Device”

9.13.3 – Ir a “Software Info”

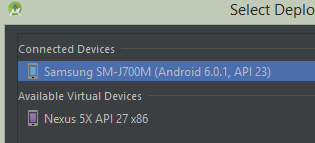
9.13.4 – Hacer click varias veces en el “Build Number” hasta que vea el mensaje de que lo felicita porque ya es un desarrollador de Android

9.13.5 – Regresar a Ajustes y se debe haber habilitado una opción adicional llamada “Developer Options”

9.13.6 – Dentro de este menú habilite “USB Debugging”

9.14 – Conecte el dispositivo Android a su computador, si aparece un mensaje hacer click en Aceptar para que Android Studio puede detectar correctamente su dispositivo.

9.15 - Luego probaremos las aplicaciones por defecto de Android… Ir a Android Studio y hacer click en Run | Run Android



9.16 – Seleccione el dispositivo a lanzar, en mi caso lanzaré la aplicación en el dispositivo real. Esto instalará 4 aplicaciones. Sea paciente y espere a que graddle haga su trabajo y estas aplicaciones se instalen en el dispositivo.

9.17 - la que nos interesa ens “TF Classify”. Abrala y apunte la cámara para ver que clasifica los objetos, aunque mal, pero funciona.

9.16 – Despues de un tiempo en.y verá el dispositivo en el emulador lanzado, seleccione su dispositvo real.

**Tensorflow Mobile – Portando nuestro algoritmo entrenado a Windows**

10.1 – Navegue hasta el interior de la carpeta en mobilenet

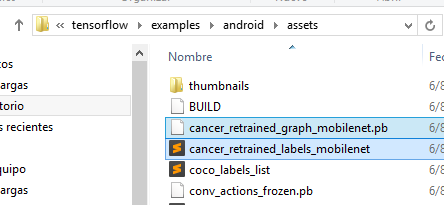
cd /home/ubuntu/skin\_cancer\_prj/mobilenet/

10.2 - Copie los modelos de de Tensorflow en su carpeta montada en Linux (que es la carpeta compartida en windows). Si pide contraseña ingrésela.

sudo cp cancer\_retrained\_graph\_mobilenet.pb ../../<carpeta\_montada\_de\_windows\_en\_linux>

sudo cp cancer\_retrained\_labels\_mobilenet.txt ../../ <carpeta\_montada\_de\_windows\_en\_linux>

10.2 – Copiados los modelos, muévalos a la carpeta assets del proyecto de Android



10.2 - Abrir ClassifierActivity.java, comente las líneas y cámbielas por estas:

// private static final int INPUT\_SIZE = 224;  
// private static final int IMAGE\_MEAN = 117;  
// private static final float IMAGE\_STD = 1;  
// private static final String INPUT\_NAME = "input";  
// private static final String OUTPUT\_NAME = "output";  
  
  
// private static final String MODEL\_FILE = "file:///android\_asset/tensorflow\_inception\_graph.pb";  
// private static final String LABEL\_FILE =  
// "file:///android\_asset/imagenet\_comp\_graph\_label\_strings.txt";  
 private static final int *INPUT\_SIZE* = 224;  
 private static final int *IMAGE\_MEAN* = 128;  
 private static final float *IMAGE\_STD* = 128;  
 private static final String *INPUT\_NAME* = "Placeholder";  
 private static final String *OUTPUT\_NAME* = "final\_result";  
  
 private static final String *MODEL\_FILE* = "file:///android\_asset/cancer\_retrained\_graph\_mobilenet.pb";  
 private static final String *LABEL\_FILE* =  
 "file:///android\_asset/cancer\_retrained\_labels\_mobilenet.txt";

10.3 – Observe que no haya cometido ningun error en la escritura de los archivos, de lo contrario no funcionará el app

10.4 – Salve todo en File | Save All

10.5 - Ejecutar la aplicación con Run | Run Android

10.6 - Éspere a que se instalen las aplicaciones



10.7 - Abrira la aplicación “TF Classify”

10.8 – Busque imágenes de estas enfermedades en <http://www.images.google.com/> , apunte con la cámara y observe como clasifica las enfermedades.



**Tarea para el restante 50% – Reentrenar la red neuronal para detectar lunares y piel**

Entrenar la red neuronal con MobileNet para detectar además de los canceres ya realizados; piel, lunares y pecas.

* Crear una carpeta llamada “lunar” con 100 imágenes de lunares
* Crear una carpeta llamada “sin lesion” con 100 imágenes de piel
  + sin heridas
  + piel de modelo de pasarela
* Crear una carpeta llamada “pecas” con 100 imágenes de pecas
* Reentrenar el modelo
* Probar 1 imagen de cada cáncer, lunar, pecas y piel limpia
* Mostrar los resultados

Nota: Buscar un método de realizar web scrapping para bajar automáticamente imágenes de google. Por ejemplo, pueden utilizar el siguiente código en python:

<https://github.com/hardikvasa/google-images-download>

**Recomendaciones**

Si está interesado en el tema puede buscar otros recursos de explicación y tecnologías como:

Los modelos COCO, R-CNN, PASCAL, VGG16, VGG19, Imagenet, Xception, Inception, Mobilenet

Los frameworks Keras, Tensorflow, Pytorch, scikit-learn, NVIDIA Digits, CUDA

Tensorflow, Tensorflow Lite, Tensorflow Mobile, Tensorflow Serving, Tensorflow JS

Cursos como fast.ai, Coursera Deep Learning, Coursera Machine Learning, Coursera Google Cloud Deep Learning, Udacity Intro to Deep Learning, Standford CS231n, Intro to Machine Learning y Tensorflow for Poets en colab

GPU Libre de cargos para entrenamiento en google colabs, floydhub

Nivel competitivo en Kaggle

Y libros varios en la librería online de Safari afiliados los estudiantes de la universidad latina.