



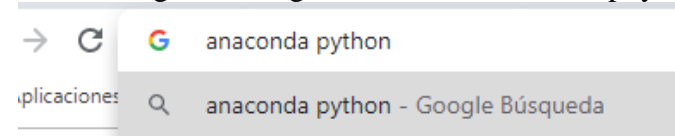
TUTORIAL DE COMO GENERAR ROSTROS ARTIFICIALES USANDO GAN (GENERATIVE ADVERSARIAL NETS).

El desarrollo de este tutorial se realiza usando la plataforma de código abierto ANACONDA, con el IDE JUPITER NOTEBOOK programando en PYTHON.

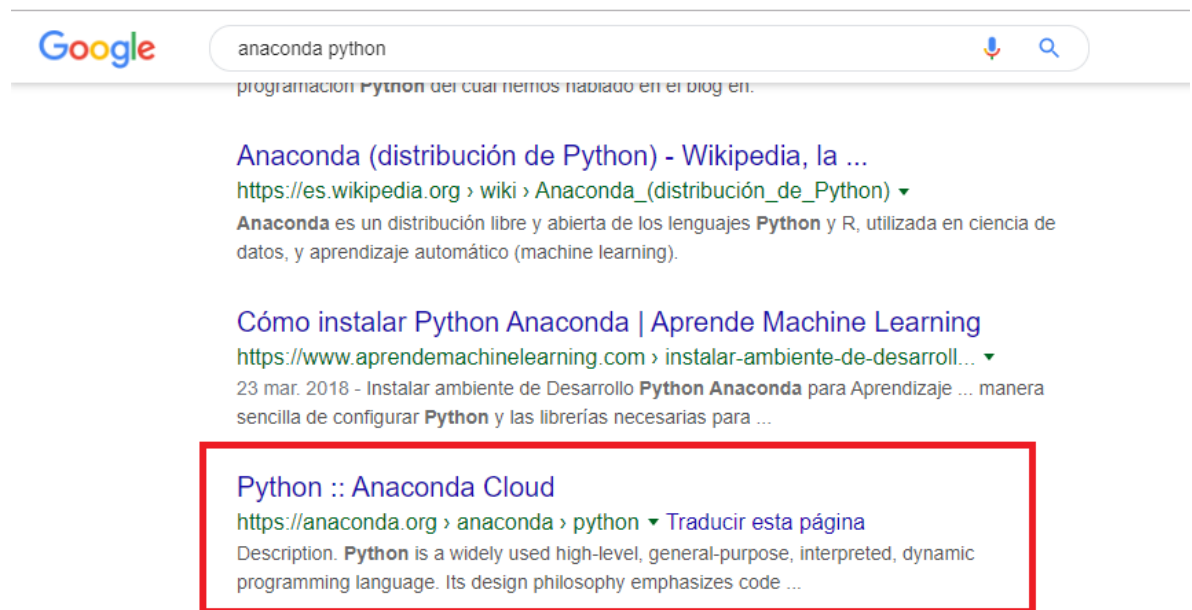
INSTALACION DE ANACONDA.

Anaconda es una plataforma de código abierto que se dedica a la ciencia de los datos, aprendizaje automático y big data. El proceso de instalación se describe a continuación:

1. En el navegador Google buscamos: anaconda phyton



Del resultado de la búsqueda clic en: anaconda Cloud





El siguiente paso es descargar el instalador así: clic en **DOWNLOAD ANACONDA**

Al dar clic se desplegará una nueva ventana donde se descarga el ejecutable para Windows dando clic en **DOWLOAD**, después de unos segundos el ejecutable empezará a descargarse.

Universidad de Nariño- Facultad de Ingenierías
Departamento de electrónica
Electiva I: Procesamiento de imágenes.
Profesor: Wilson Achicanoy.

David Aldemar Cordoba Vargas

Cuando la descarga haya finalizado se obtendrá un ejecutable de este tipo:



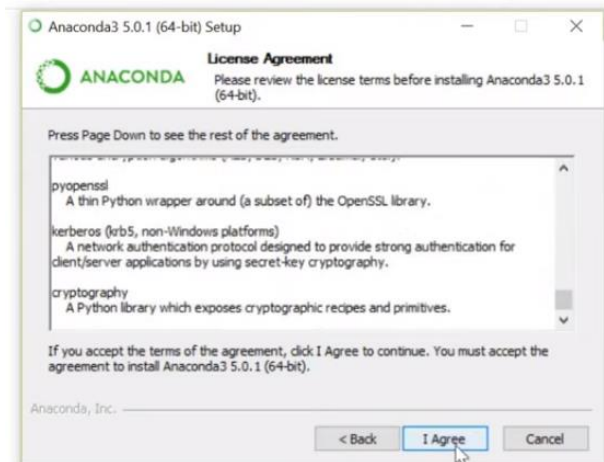
Anaconda3-2019.07-Windows-x86_64.exe

El ejecutable también se encuentra en los archivos de este tutorial en la carpeta GANS\ANACONDA.

Proceso siguiente a la descarga es dar doble clic este ejecutable que acabamos de descargar y seguimos la siguiente rutina:



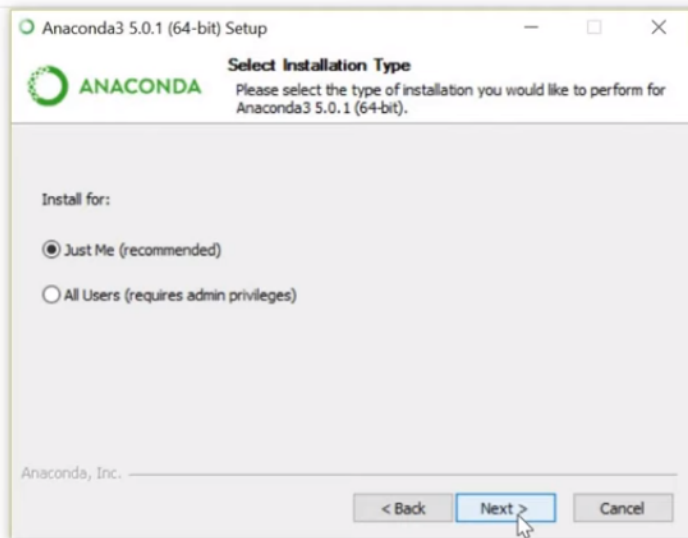
Se da clic en siguiente(Next).



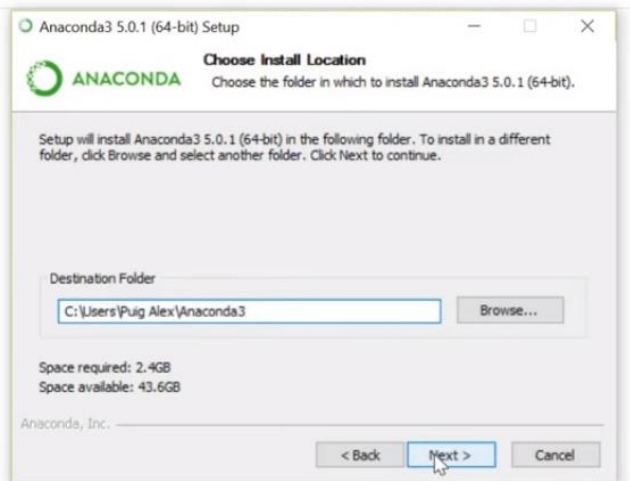
Se lee las condiciones y se procede a dar clic en De acuerdo (I Agree).



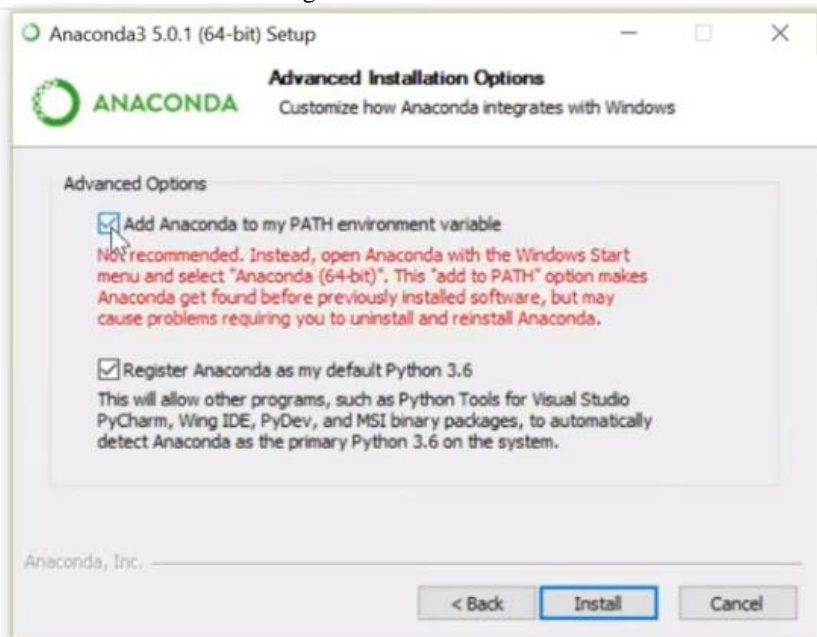
Se instala solo para nuestro Usuario.



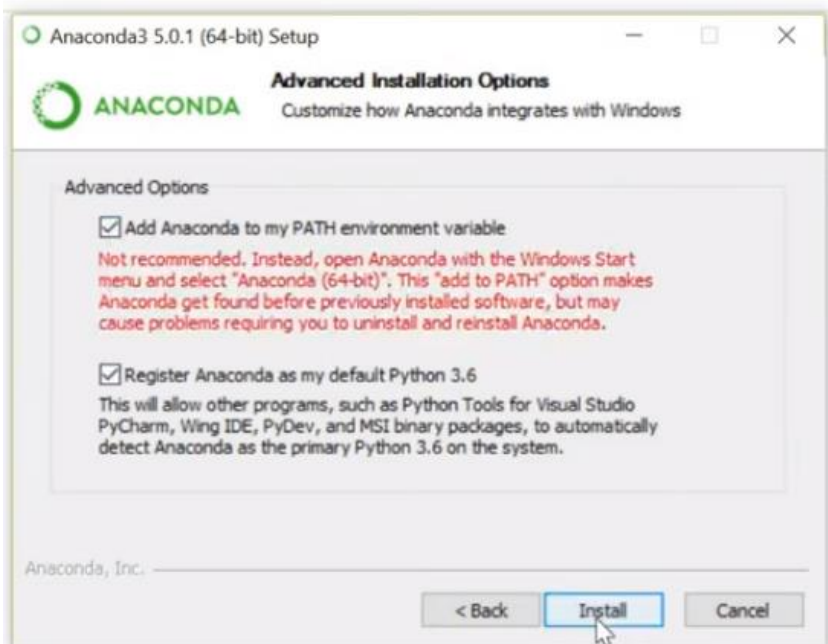
Se elige donde instalar o se deja por defecto la dirección de instalación, después clic en siguiente (Next)



Se da clic en el primer recuadro, esto para la instalación de paquetes en el futuro sin tener problemas.



Posterior a eso se da clic en instalar.



Universidad de Nariño- Facultad de Ingenierías

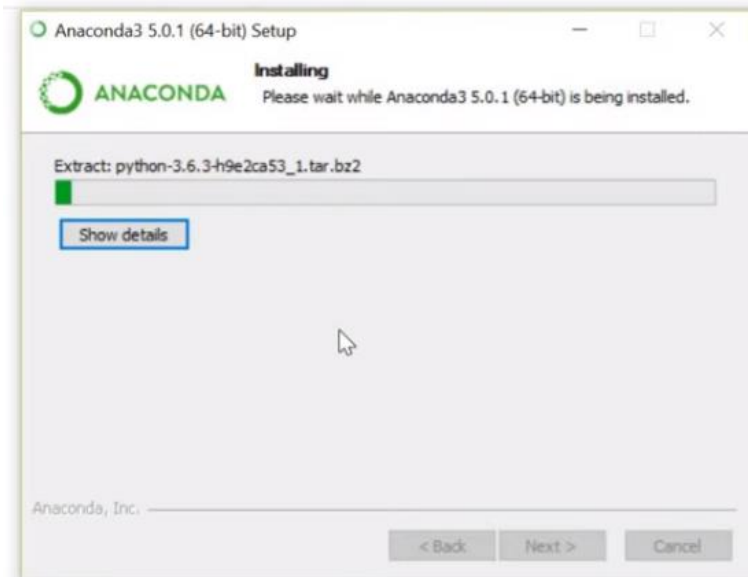
Departamento de electrónica

Electiva I: Procesamiento de imágenes.

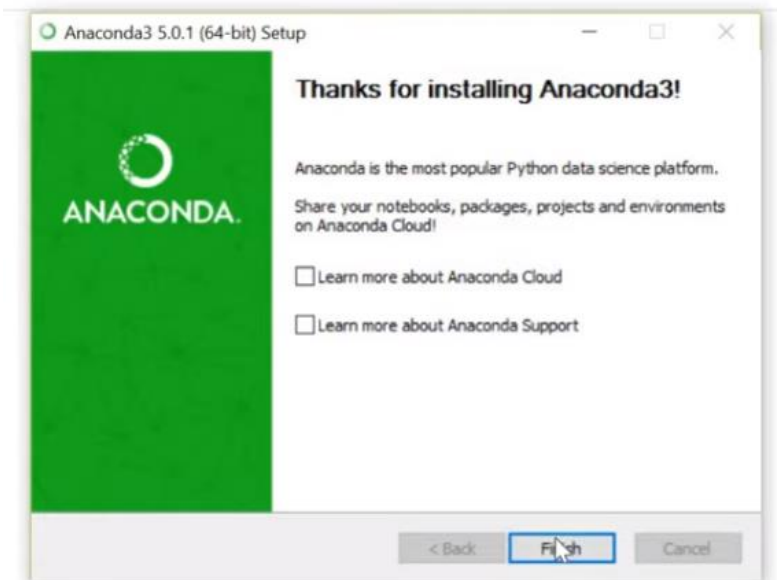
Profesor: Wilson Achicanoy.

David Aldemar Cordoba Vargas

Inicia el proceso de instalación, se debe esperar unos minutos hasta que el proceso termine



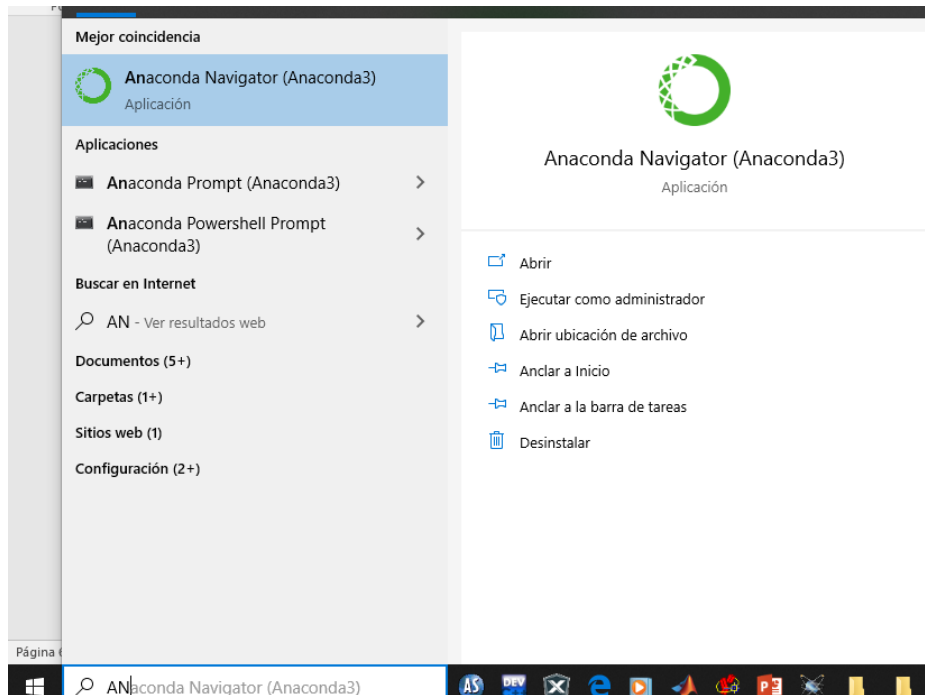
Cuando termine de instalar se debe quitar el chek de los dos recuadros de la ventana como se indica abajo, clic en finalizar(Finish).



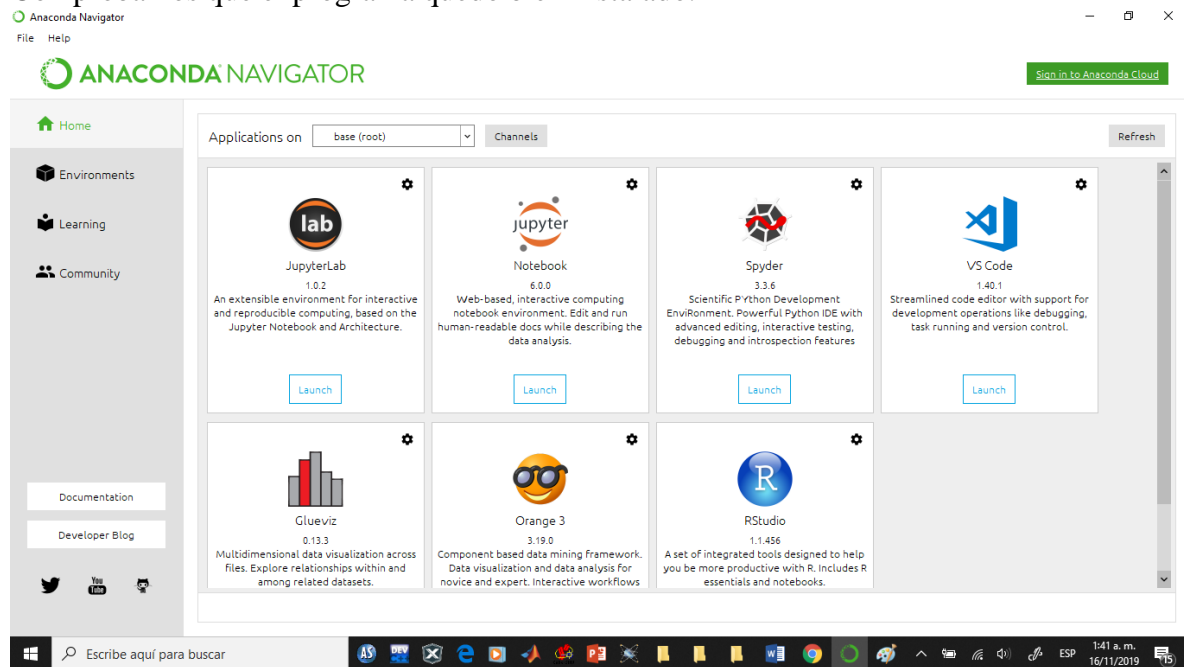


2. PARA USAR ANACONDA NAVIGATOR

Buscamos en el inicio Anaconda Navigator y abrimos el interfaz.



Comprobamos que el programa quedo bien instalado.

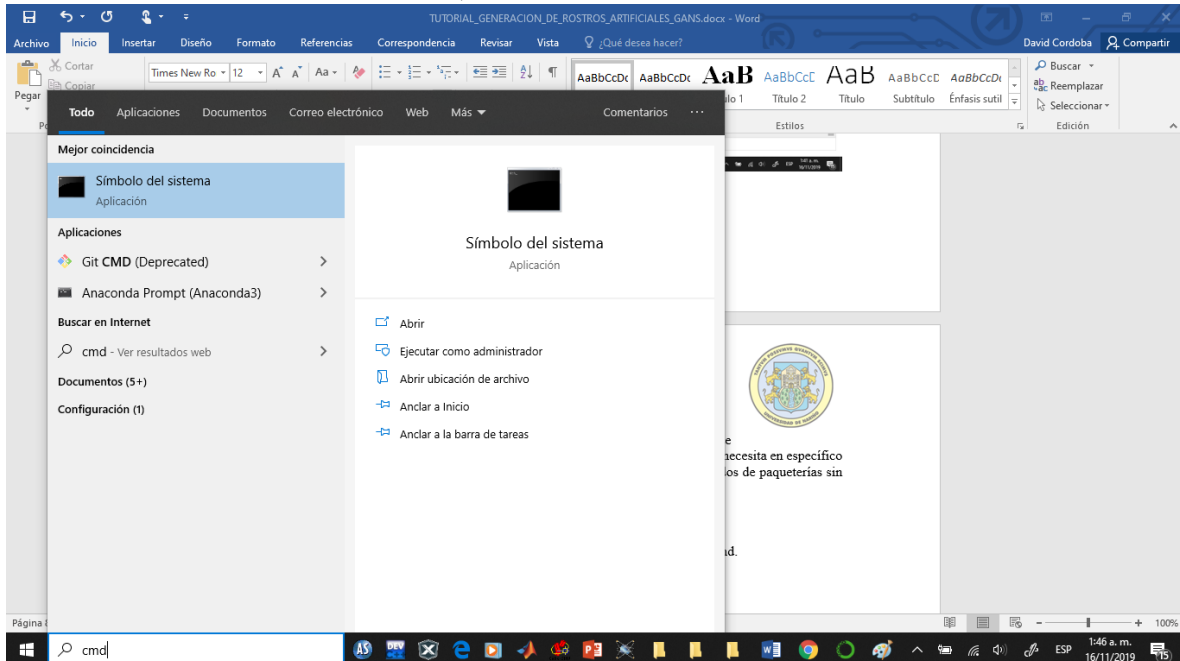




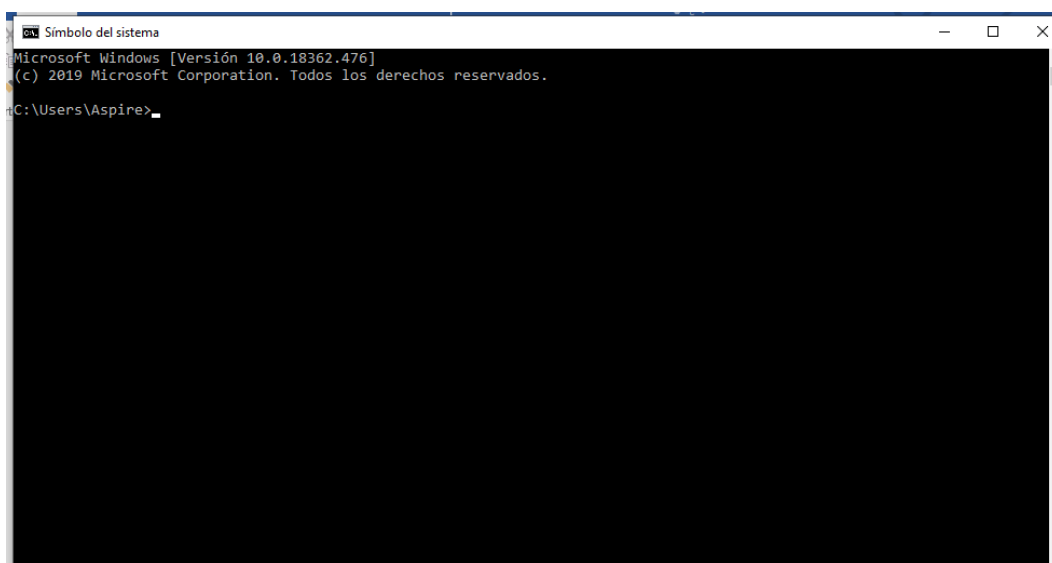
NOTA: Anaconda tiene la posibilidad de crear varios entornos de desarrollo, de manera que cada uno tiene las paqueterías que cada uno necesita en específico y esto es realmente útil, pues evita que usemos ambientes sobrecargados de paqueterías sin usar.

3. CREAR AMBIENTE

Para crear el ambiente en anaconda, en la barra de inicio buscamos: cmd.



Clic en la terminal del sistema:

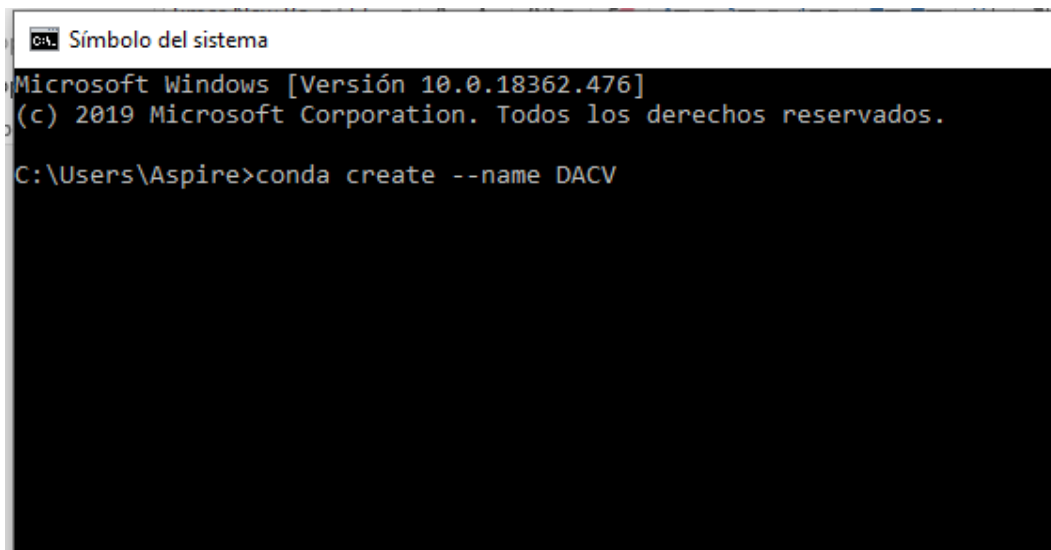




Cuando la terminal ya se encuentre abierta, el comando que se utilizara para la creación de un nuevo ambiente es:

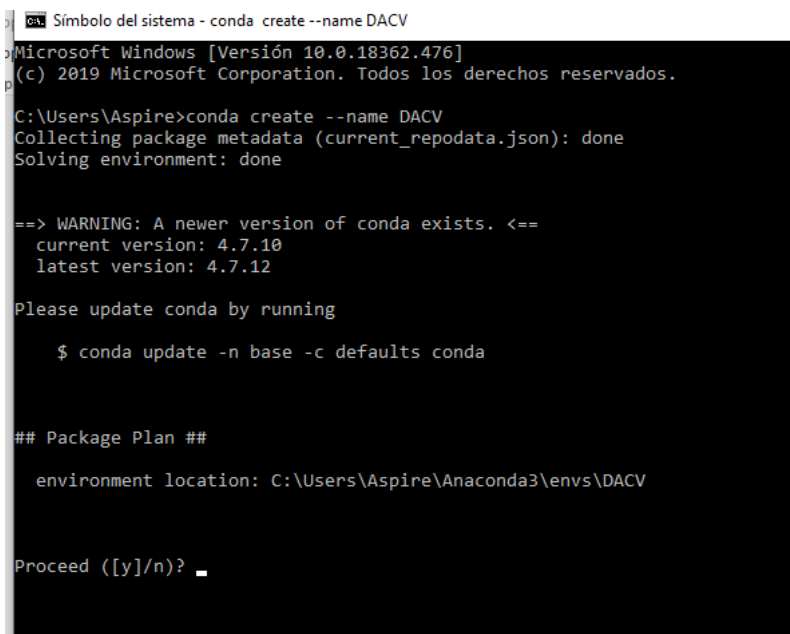
`conda create --name NOMBRE`

para el ejemplo usaremos el nombre DACV, pero puede ser cualquier otro nombre que se desee.



```
C:\Users\Aspire>conda create --name DACV
```

Después de poner el comando que aparece en la imagen, se procede a confirmar con ENTER.



```
C:\Users\Aspire>conda create --name DACV
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.7.10
  latest version: 4.7.12

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\Aspire\Anaconda3\envs\DACV

Proceed ([y]/n)?
```

La confirmación del proceso es presionar la tecla y



```
C:\> Símbolo del sistema - conda create --name DACV
Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>conda create --name DACV
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.7.10
  latest version: 4.7.12

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\Aspire\Anaconda3\envs\DACV

Proceed ([y]/n)? y_
```

Después de confirmar, tecleando y, se presiona en ENTER.

```
C:\> Símbolo del sistema
latest version: 4.7.12

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\Aspire\Anaconda3\envs\DACV

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate DACV
#
# To deactivate an active environment, use
#
#     $ conda deactivate

C:\Users\Aspire>
```

El ambiente se ha creado satisfactoriamente, lo podemos comprobar de la siguiente manera:



El comando que permite verificar si el ambiente se creó es:
activate NOMBRE_DEL_AMBIENTE

```
C:\> Símbolo del sistema

Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV_
```

Cuando se presione ENTER deberá salir lo siguiente:

```
C:\> Símbolo del sistema

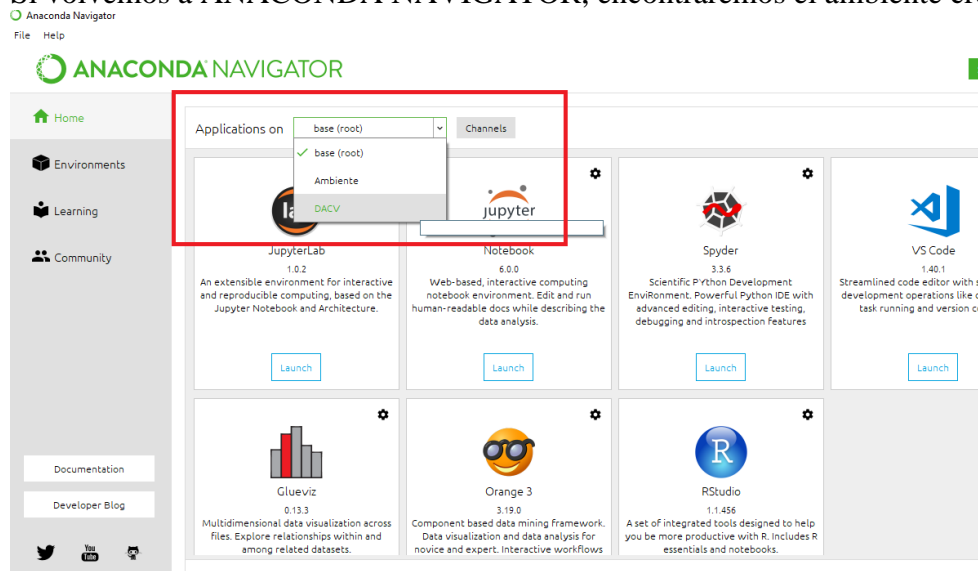
Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV

(DACV) C:\Users\Aspire>
```

Eso confirma que el ambiente fue instalado correctamente.

Si volvemos a ANACONDA NAVIGATOR, encontraremos el ambiente creado.



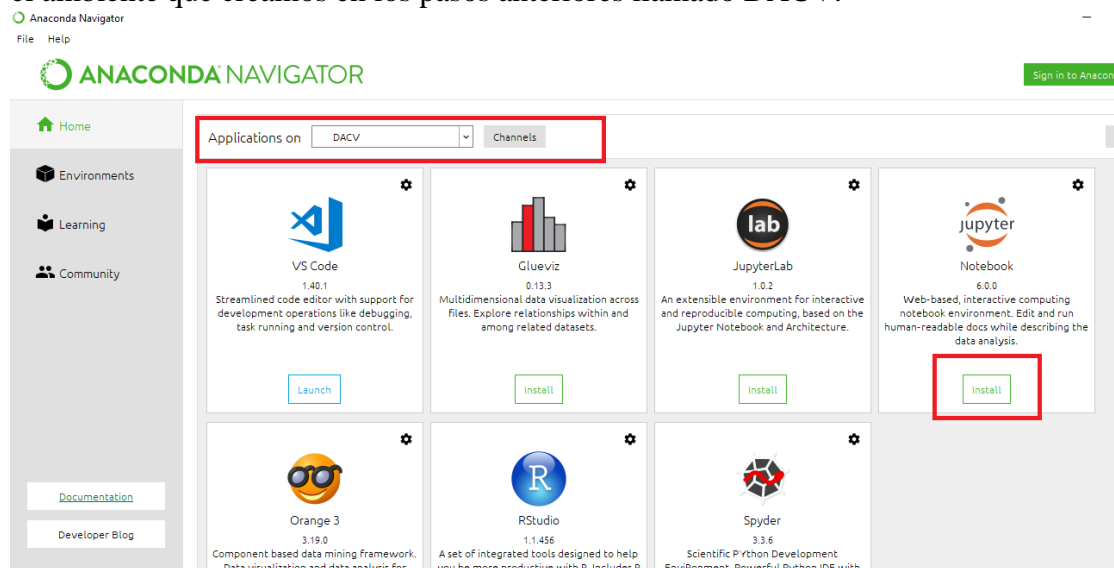
Clic en el ambiente que acabamos de crear, en este caso DACV.



ACTIVACION DE JUPYTER NOTEBOOK

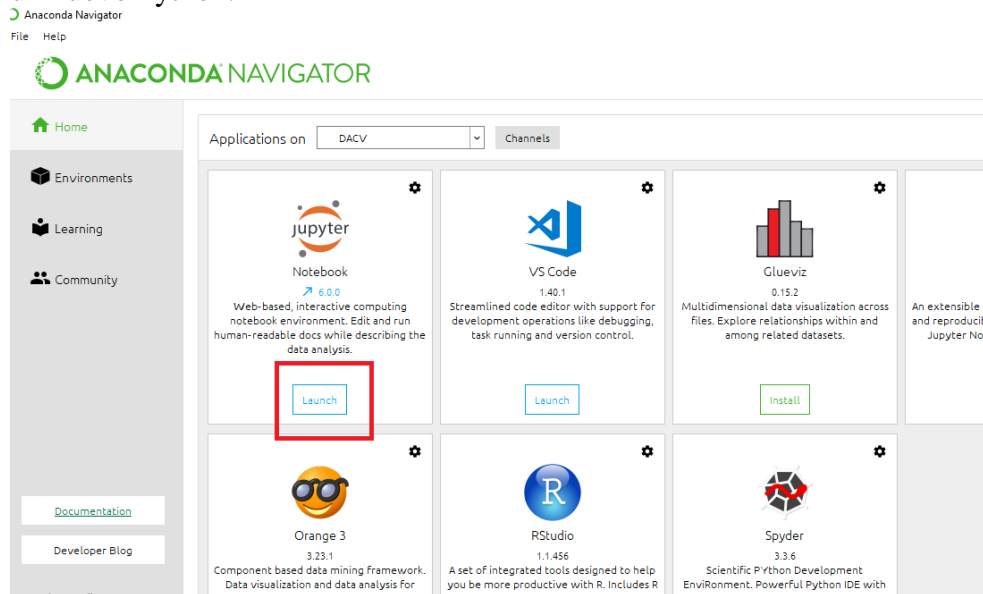
Jupyter es un entorno computacional interactivo basado en la web que nos permite crear documentos jupyter, un documento notebook jupyter es un documento JSON que usa la extensión (.ipynb).

Para la activación se necesita estar en el ambiente conveniente a tu proyecto, en este caso en el ambiente que creamos en los pasos anteriores llamado DACV.



Cuando estemos allí: se da clic en Install, cuando termine de instalar podemos correrlo.

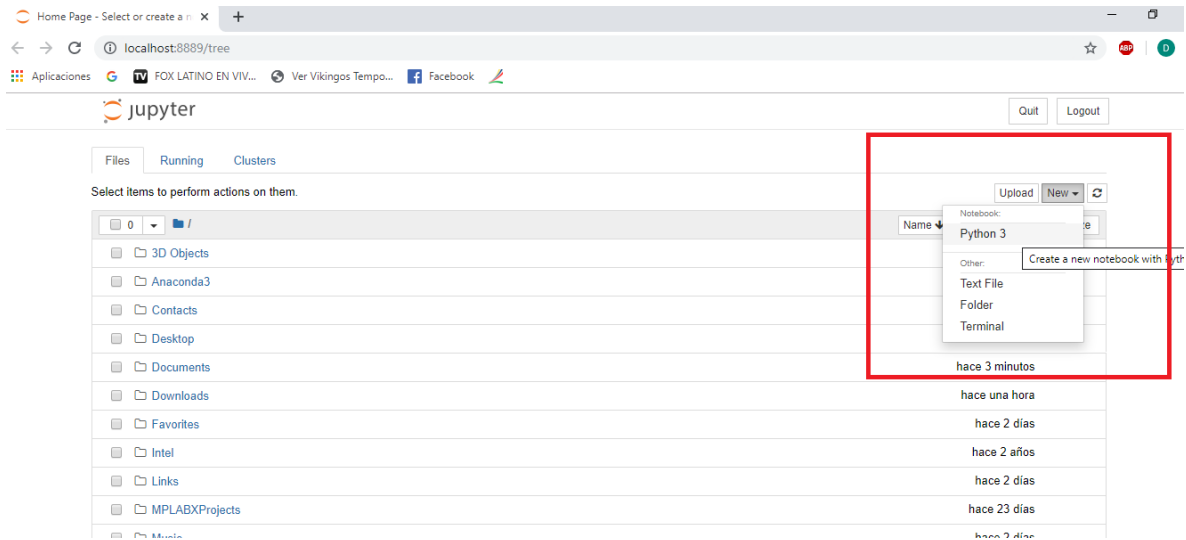
Para correrlo se da clic en Launch, lo que hace es abrir un explorador donde podemos abrir un nuevo Python.



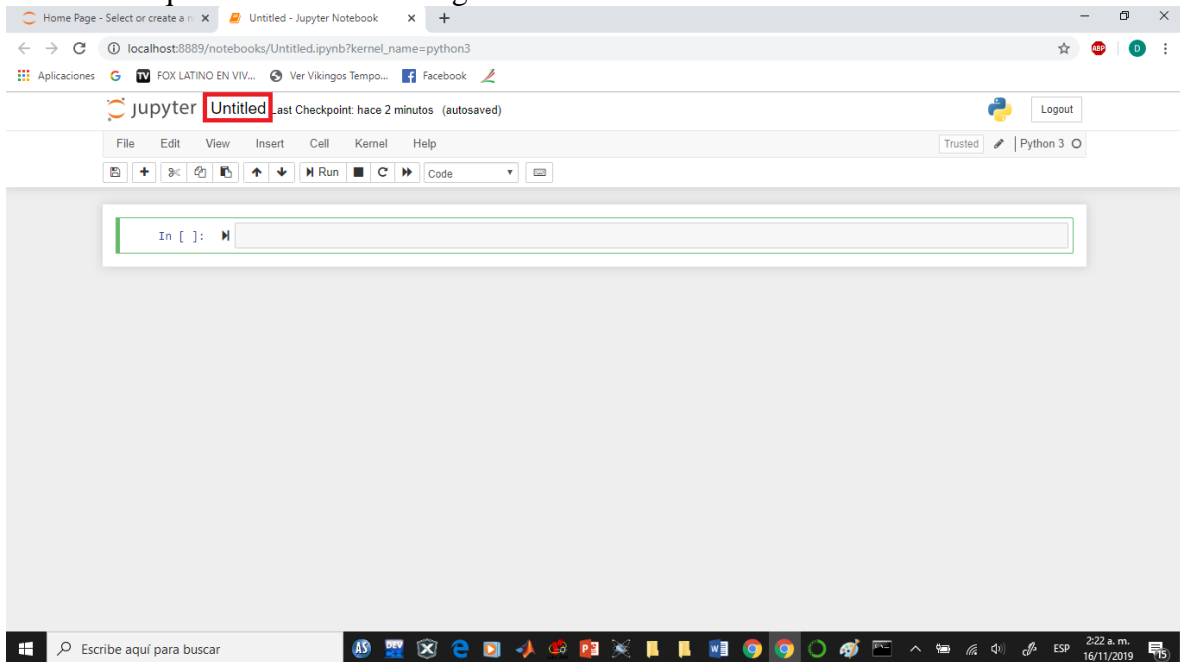
Universidad de Nariño- Facultad de Ingenierías
Departamento de electrónica
Electiva I: Procesamiento de imágenes.
Profesor: Wilson Achicanoy.
David Aldemar Cordoba Vargas



La ventana emergente permite crear un editor de texto, haciendo lo siguiente, clic en New , clic en Python 3



Al finalizar obtendremos el editor de texto. Puede cambiarle el nombre dando doble clic en el recuadro que se señala en la imagen.



NOTA: El archivo se crea en la carpeta predeterminada para documentos del equipo donde se instale la plataforma anaconda, en este caso la dirección a la carpeta de almacenamiento es: C:\Users\Aspire



INSTALACION DE PAQUETERIA NECESARIA PARA CORRER EL CODIGO.

En la terminal del sistema, se digita los siguiente:

activate NOMBRE_DEL_AMBIENTE

```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV_
```

Cuando se presione ENTER deberá salir lo siguiente:

```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV
(DACV) C:\Users\Aspire>_
```

Cuando el nombre de ambiente creado se representa en paréntesis, significa que estamos dentro del ambiente que se creó correctamente.

LIBRERIAS NECESARIAS PARA EL CODIGO.

Se deben instalar las siguientes librerías para que el código pueda correr sin problemas, se usa la terminal del sistema con el comando **pip install NOMBRE_LIBRERIA**.

numpy
imageio
matplotlib
keras
tensorflow
pandas

NOTA: en algunas paqueterías requiere de confirmación, que significa, teclear la tecla y, vale aclarar que no es para todas las paqueterías, solo para algunas.



Ejemplo: instalar paquetería numpy en el ambiente creado en los pasos previos.

```
C:\> Símbolo del sistema

Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV

(DACV) C:\Users\Aspire>pip install numpy_
```

Para confirma se debe presionar ENTER.

```
C:\> Símbolo del sistema

Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Aspire>activate DACV

(DACV) C:\Users\Aspire>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/34/40/c6eae19892551ff91bdb15f884fef2d42d6f58da55ab18fa540851b48a32/numpy-1.17.4-cp37-cp37m-win_amd64.whl (12.7MB)
    | 12.7MB 656kB/s
Installing collected packages: numpy
Successfully installed numpy-1.17.4

(DACV) C:\Users\Aspire>
```

Eso significa que la paquetería se instaló correctamente y su instalación finalizó.

CORRECCION DE ERROR.

Cuando después de iniciar a correr el programa inesperadamente en una de las iteraciones aparece un error de este tipo:

la solución es copiar y pegar las siguientes líneas de código en la cabecera del programa:

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
tf.reset_default_graph()
```

Universidad de Nariño- Facultad de Ingenierías
Departamento de electrónica
Electiva I: Procesamiento de imágenes.
Profesor: Wilson Achicanoy.
David Aldemar Cordoba Vargas
EXTRA:



Cuando una librería no se encuentra instalada, se corrige de la siguiente manera:

Problemas con las librerías o paqueterías.

```
--  
ModuleNotFoundError                                Traceback (most recent call las  
t)  
<ipython-input-2-e738239fbd96> in <module>  
      3  
      4 import os  
----> 5 from imageio import imread, imwrite  
      6 import numpy as np  
      7 import matplotlib.pyplot as plt  
  
ModuleNotFoundError: No module named 'imageio'
```

Solución de librerías, instalarlas.

```
ca. Símbolo del sistema  
Microsoft Windows [Versión 10.0.18362.476]  
(c) 2019 Microsoft Corporation. Todos los derechos reservados.  
  
C:\Users\Aspire>activate DACV  
  
(DACV) C:\Users\Aspire>pip install imageio
```

ENTER PARA CONFIRMAR

```
ca. Símbolo del sistema  
Microsoft Windows [Versión 10.0.18362.476]  
(c) 2019 Microsoft Corporation. Todos los derechos reservados.  
  
:\Users\Aspire>activate DACV  
  
DACV) C:\Users\Aspire>pip install imageio  
collecting imageio  
Using cached https://files.pythonhosted.org/packages/1a/de/f7f985018f462ceeffada7f6e609919fbcc934acd9301929cba14bc2c24  
/imageio-2.6.1-py3-none-any.whl  
Requirement already satisfied: numpy in c:\users\aspire\anaconda3\envs\dacv\lib\site-packages (from imageio) (1.17.4)  
collecting pillow  
Downloading https://files.pythonhosted.org/packages/70/f4/9dd0b7b0fea09cf4e7a2822031f2157f40d41f0252a89558bdb583e24ef1  
Pillow-6.2.1-cp37-cp37m-win_amd64.whl (2.0MB)  
2.0MB 344kB/s  
Installing collected packages: pillow, imageio  
Successfully installed imageio-2.6.1 pillow-6.2.1  
  
DACV) C:\Users\Aspire>
```

FIN.



ARCHIVOS:

En la carpeta de nombre `ELECTIVA_IMAGENES` se encuentra otra del nombre `TUTORIAL_GENERACION_DE_ROSTROS_ARTIFICIALES_USANDO_GANS`, conformada por un dataset de 3754 imágenes de rostros con tamaño 128x128x3, un archivo python llamado 'utilidades.py' donde se encuentran las funciones para que el código de generación de rostros artificiales sea más ordenado, una carpeta llamada ejemplos donde se guardaran las muestras cuando el modelo acabe su entrenamiento, y un block de notas donde se encuentra el código fuente.

PROCEDIMIENTO:

1. Copie y pegue la carpeta:
`TUTORIAL_GENERACION_DE_ROSTROS_ARTIFICIALES_USANDO_GANS`
En la dirección que tiene por defecto jupyter notebook en el ambiente creado, por lo general esta se encuentra en `C:\Users\Aspire`.
2. Ingrese a través de Jupyter Notebook a la carpeta que se copió en el paso anterior y abra el block de notas llamado 'generar_rostros_artificiales', copie todo el código que se encuentra en este, posterior a esto cree un nuevo archivo de python3 en jupyter Notebook (puede usar de guía en el paso anterior llamada ACTIVACION DE JUPYTER NOTEBOOK) asegurándose que el archivo quede guardado en la misma carpeta que el archivo .txt, colocándole el mismo nombre 'generar_rostros_artificiales', y pegue allí lo copiado desde el block de notas.
3. Proceda a correr el código y empiece a generar rostros artificiales, por cada iteración se genera una imagen que muestra el proceso del entrenamiento en términos del generador, cuando concluya el programa se generan 100 ejemplos usando el generador entrenado, estas se guardan en la carpeta ejemplos, ese es el motivo de que esta, esté vacía al principio, además, se genera un archivo .h5 que representa al modelo entrenado('generador.h5').

NOTA: Por motivos de tiempo y puesto que es una demostración el número de iteraciones es reducido de 5000 a 10, con el fin de mostrar la convergencia del algoritmo, esto implica que, los resultados obtenidos después de la décima iteración no se asemejarán a un rostro humano. Si se desea entrenar el modelo la variable a incrementar y que representa el número de iteraciones se encuentra en la inicialización de parámetros del código fuente y corresponde a `N_ITS`.



Para usar el dataset en el proyecto se usa la siguiente función de código, la línea de código marcada sirve para normalizar [0 - 1] las imágenes.



```
from imageio import imread, imwrite
import numpy as np

def cargar_datos():
    print('Creando set de entrenamiento...',end="",flush=True)
    filelist = os.listdir(dataset)

    n_imgs = len(filelist)
    x_train = np.zeros((n_imgs,128,128,3))

    for i, fname in enumerate(filelist):
        if fname != '.DS_Store':
            imagen = imread(os.path.join(dataset,fname))
            x_train[i,:] = (imagen - 127.5)/127.5
    print('¡Listo!')

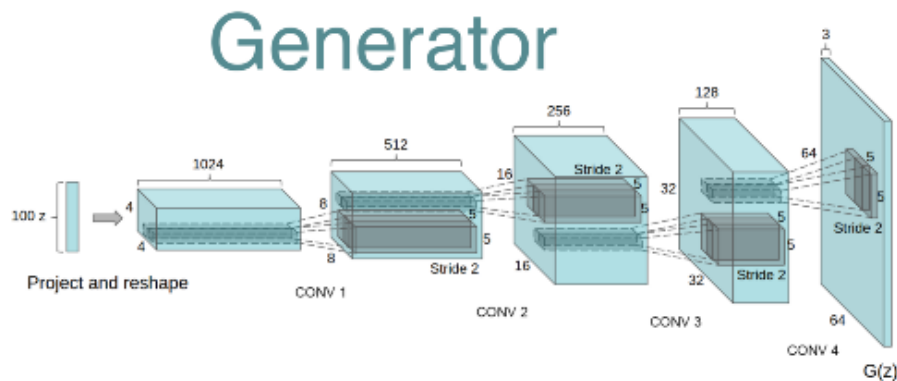
    return x_train
```

GENERADOR:

El generador tomará un vector con cien números aleatorios y será entrenado para generar imágenes de 128x128. Progresivamente serán más parecidas a un rostro real.



La arquitectura para el generador es la DCGAN



Las líneas de código que representan al generador es la siguiente:

```
# Generator
def crear_generador():
    modelo = Sequential()
    modelo.add(Dense(1024*4*4, use_bias=False, input_shape=(TAM_ENTRADA,)))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    modelo.add(Reshape((4,4,1024)))
    #4x4x1024

    modelo.add(Conv2DTranspose(512,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #8x8x512

    modelo.add(Conv2DTranspose(256,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #16x16x256

    modelo.add(Conv2DTranspose(128,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #32x32x128

    modelo.add(Conv2DTranspose(64,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #64x64x64

    modelo.add(Conv2DTranspose(3, (5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(Activation('tanh'))
    #128x128x3

    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

generador = crear_generador()
```



EXPLICACION:

Para el caso del Generador iniciaremos con un vector de 100 elementos, que será conectada a una red neuronal con 1024x1024 elementos, que después será redimensionado a un volumen de 4x4x1024, en las capas restantes se usará la convolución inversa (Conv2DTranspose) hasta progresivamente llegar al volumen deseado de 128x128x3. En todas las capas exceptuando la de salida se usará la función de activación leakyrelu (No elimina por completo los valores negativos), de igual forma en todas las capas exceptuando la de salida usaremos BatchNormalization (Garantiza que la salida de cada capa tenga una desviación estándar = 1 y un valor medio = 0), esto ayuda a la convergencia del algoritmo de optimización en el entrenamiento. Finalmente en la capa de salida usaremos la función de activación tangente hiperbólica 'tanh' para obtener imágenes generadas con pixeles en el rango de [0-1] que es el mismo rango usado en las imágenes de entrenamiento reales.

```
# Generador
def crear_generador():
    modelo = Sequential()
    modelo.add(Dense(1024*4*4, use_bias=False, input_shape=(TAM_ENTRADA,)))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    modelo.add(Reshape((4,4,1024)))
    #4x4x1024

    modelo.add(Conv2DTranspose(512,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #8x8x512

    modelo.add(Conv2DTranspose(256,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #16x16x256

    modelo.add(Conv2DTranspose(128,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #32x32x128

    modelo.add(Conv2DTranspose(64,(5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #64x64x64

    modelo.add(Conv2DTranspose(3, (5,5),strides=(2,2),padding='same', use_bias=False))
    modelo.add(Activation('tanh'))
    #128x128x3

    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

generador = crear_generador()
```



DISCRIMINADOR:

Tendrá como entrada una entra real o falsa y a la salida entregará un numero entre cero y uno indicando la categoría a la que pertenece la imagen de entrada.



La arquitectura para discriminador es la DCGAN





```
def crear_discriminador():
    modelo = Sequential()
    modelo.add(Conv2D(64, (5,5), strides=(2,2), padding='same', input_shape=(128,128,3),
        use_bias=False))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #64x64x64

    modelo.add(Conv2D(128, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #32x32x128

    modelo.add(Conv2D(256, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #16x16x256

    modelo.add(Conv2D(512, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #8x8x512

    modelo.add(Conv2D(1024, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #4x4x1024

    modelo.add(Flatten())
    modelo.add(Dense(1, activation='sigmoid', use_bias=False))

    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

discriminador = crear_discriminador()
```

EXPLICACION:

Para el caso del discriminador se usará una red prácticamente opuesta al generador, pues esta recibe imágenes de 128x128x3(imagen del dataset o imagen generada) y la salida será un número que corresponde a la categoría de la imagen de entrada (Real o Falsa) se obtendrá un valor entre [1 -0] con una función de activación sigmoideal, las capas convolucionales (Con2D) se encargarán de progresivamente reducir la imagen, ancho y alto y de incrementar el número de características extraídas de la imagen, tal como lo hace una red convolucional convencional. En todas las capas exceptuando la de salida se usará la función de activación leakyrelu (No elimina por completo los valores negativos), de igual forma en todas las capas exceptuando la de entrada usaremos BatchNormalization (Garantiza que la salida de cada capa tenga una desviación estándar = 1 y un valor medio = 0), esto ayuda a la convergencia del algoritmo de optimización en el entrenamiento.



```
def crear_discriminador():
    modelo = Sequential()
    modelo.add(Conv2D(64, (5,5), strides=(2,2), padding='same', input_shape=(128,128,3),
        use_bias=False))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #64x64x64

    modelo.add(Conv2D(128, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #32x32x128

    modelo.add(Conv2D(256, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #16x16x256

    modelo.add(Conv2D(512, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #8x8x512

    modelo.add(Conv2D(1024, (5,5), strides=(2,2), padding='same', use_bias=False))
    modelo.add(BatchNormalization(momentum=0.3))
    modelo.add(LeakyReLU(alpha=LEAKY_SLOPE))
    #4x4x1024

    modelo.add(Flatten())
    modelo.add(Dense(1, activation='sigmoid', use_bias=False))

    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

discriminador = crear_discriminador()
```

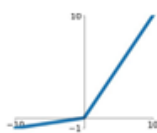
OPTIMIZADOR, METRICA Y ALPHA DE LA FUNCION DE ACTIVACION LEAKY RELU
 Se usará el optimizador Adam que es una variante del gradiente descendente pero que necesita menos iteraciones para lograr la convergencia. El error a usar será la entropía cruzada.

```
OPTIMIZADOR = Adam(lr=0.0002, beta_1=0.5)
ERROR = 'binary_crossentropy' LEAKY_SLOPE = 0.2
```

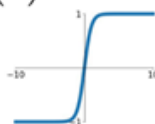
FUNCIONES DE ACTIVACIÓN

A continuación, se muestran las funciones de activación que se usan para este ejemplo práctico.

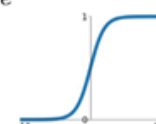
Leaky ReLU
 $\max(0.1x, x)$



tanh
 $\tanh(x)$



Sigmoid
 $\sigma(x) = \frac{1}{1+e^{-x}}$





CREACION DE LA GAN



```
# GAN
def crear_GAN(generator, discriminator):
    modelo = Sequential()
    modelo.add(generator)
    discriminator.trainable = False
    modelo.add(discriminator)
    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

gan = crear_GAN(generator, discriminator)
```

EXPLICACION:

Se usa la instrucción Sequential para crear el contenedor donde estará el modelo de discriminador y el modelo del generador. Los coeficientes de entrenamiento del discriminador empiezan congelados.

```
# GAN
def crear_GAN(generator, discriminator):
    modelo = Sequential()
    modelo.add(generator)
    discriminator.trainable = False
    modelo.add(discriminator)
    modelo.compile(optimizer=OPTIMIZADOR, loss=ERROR)

    return modelo

gan = crear_GAN(generator, discriminator)
```




PARAMETROS DE ENTRENAMIENTO

```
TAM_ENTRADA = 100  
TAM_LOTE = 128  
N_ITS = 5000
```

ENTRENAMIENTO DE LA RED GENERATIVA ADVERSARIA

En el caso de las GANs el proceso de entrenamiento (.train_on_batch) es diferente al proceso que siguen las redes neuronales convencionales(.modelo.fit), pues en la primera se tendrá el control total de los coeficientes en cada iteración.

1. Entrenar discriminador.
2. Congelar sus coeficientes.
3. Entrenar generador.
4. Descongelar Coeficientes discriminador.
5. Repetir 1 a 4.

```
for i in range(1,N_ITS+1):  
    print("Epoch " + str(i))  
  
    # Crear un "batch" de imágenes falsas y otro con imágenes reales  
    ruido = np.random.normal(0,1,[TAM_LOTE,TAM_ENTRADA])  
    batch_falsas = generador.predict(ruido)  
  
    idx = np.random.randint(low=0, high=x_train.shape[0],size=TAM_LOTE)  
    batch_reales = x_train[idx]  
  
    # Entrenar discriminador con imagener falsas y reales, y en cada  
    # caso calcular el error  
    discriminador.trainable = True  
  
    dError_reales = discriminador.train_on_batch(batch_reales,  
        np.ones(TAM_LOTE)*0.9)  
    dError_falsas = discriminador.train_on_batch(batch_falsas,  
        np.zeros(TAM_LOTE)*0.1)  
  
    discriminador.trainable = False  
  
    # Entrenar GAN: se generará ruido aleatorio y se presentará a la GAN  
    # como si fuesen imagenes reales  
    ruido = np.random.normal(0,1,[TAM_LOTE,TAM_ENTRADA])  
    gError = gan.train_on_batch(ruido, np.ones(TAM_LOTE))  
  
    # Graficar ejemplo de imágenes generadas, cada 100 iteraciones  
    if i==1 or i%100 == 0:  
        graficar_imagenes_generadas(i,generador)  
  
generador.save('generador.h5')  
generar_imagenes(generador,100)
```



EXPLICACION

Se crean dos lotes de 128 imágenes cada uno, el primero con imágenes reales provenientes del dataset y el segundo con imágenes falsas obtenidas con el generador, para cada uno de estos lotes generamos las categorías correspondientes, uno para cada imagen real y cero para imagen falsa. Se descongela los coeficientes de entrenamiento del discriminador y con los lotes generados se entrena al discriminador usando (train_on_batch), congelamos de nuevo los coeficientes del discriminador, y entrenamos al GAN (gan), pues al ser congelados los coeficientes del discriminador solo se entrenará el generador. Finalmente podemos con el generador guardar ejemplos en algunas iteraciones del entrenamiento, podemos igualmente guardar en disco duro el modelo obtenido y generar algunos ejemplos de lotes de imágenes.

```
for i in range(1,N_ITS+1):
    print("Epoch " + str(i))

    # Crear un "batch" de imágenes falsas y otro con imágenes reales
    ruido = np.random.normal(0,1,[TAM_LOTE,TAM_ENTRADA])
    batch_falsas = generador.predict(ruido)

    idx = np.random.randint(low=0, high=x_train.shape[0],size=TAM_LOTE)
    batch_reales = x_train[idx]

    # Entrenar discriminador con imagener falsas y reales, y en cada
    # caso calcular el error
    discriminador.trainable = True

    dError reales = discriminador.train_on_batch(batch_reales,
        np.ones(TAM_LOTE)*0.9)
    dError falsas = discriminador.train on batch(batch falsas,
        np.zeros(TAM_LOTE)*0.1)

    discriminador.trainable = False

    # Entrenar GAN: se generará ruido aleatorio y se presentará a la GAN
    # como si fuesen imagenes reales
    ruido = np.random.normal(0,1,[TAM_LOTE,TAM_ENTRADA])
    gError = gan.train_on_batch(ruido, np.ones(TAM_LOTE))

    # Graficar ejemplo de imágenes generadas, cada 100 iteraciones
    if i==1 or i%100 == 0:
        graficar_imagenes_generadas(i,generador)

    generador.save('generador.h5')
    generar_imagenes(generador,100)
```

Universidad de Nariño- Facultad de Ingenierías
Departamento de electrónica
Electiva I: Procesamiento de imágenes.
Profesor: Wilson Achicanoy.
David Aldemar Cordoba Vargas



REFERENCIA

Canal de YouTube dedicado a la inteligencia artificial llamado: **codificandobits**, el creador Miguel Sotaquirá.

Link del canal: <https://www.youtube.com/channel/UCFVF0MpD1INdU5VL3Pz67Yw>