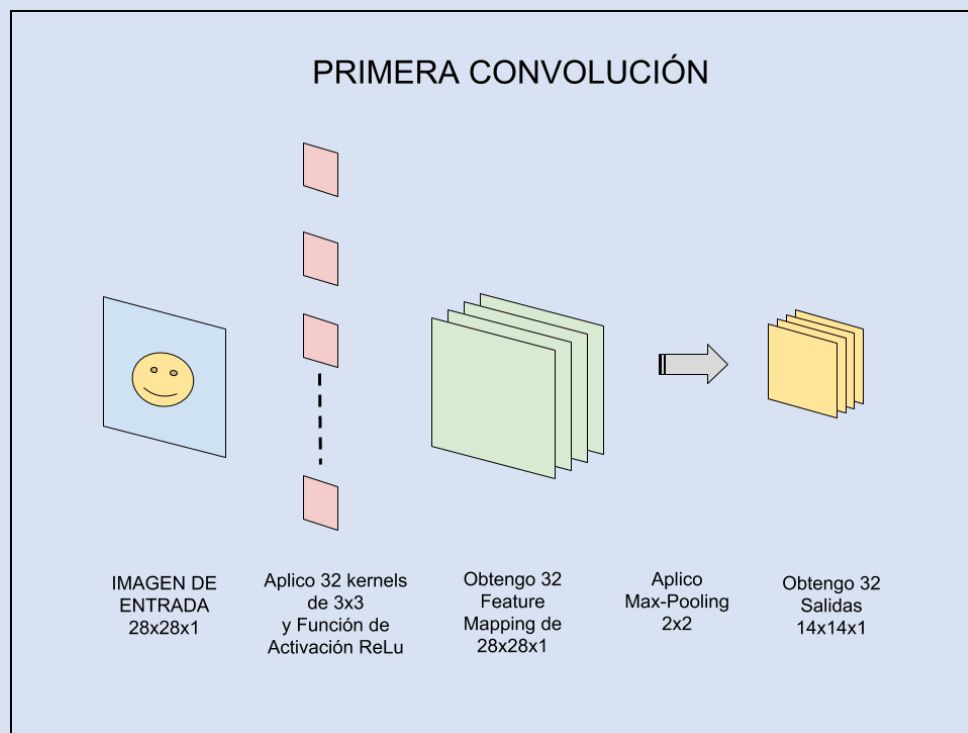




# Prácticas voluntarias:

## *Aplicaciones de Deep Learning con redes convolucionales*



*Montserrat Sacie Alcázar*

*Asignatura de Ingeniería del Conocimiento*

*Facultad de Informática, UCM, 2020*



## *PRÁCTICA 1:*

*Clasificación de fotografías  
de objetos con la red  
AlexNet*



1. *¿Qué es AlexNet?*
2. *Objetivo de la práctica 1*
3. *Manual de instalación*
4. *Ejecución del programa para clasificar objetos*
5. *Conclusiones*

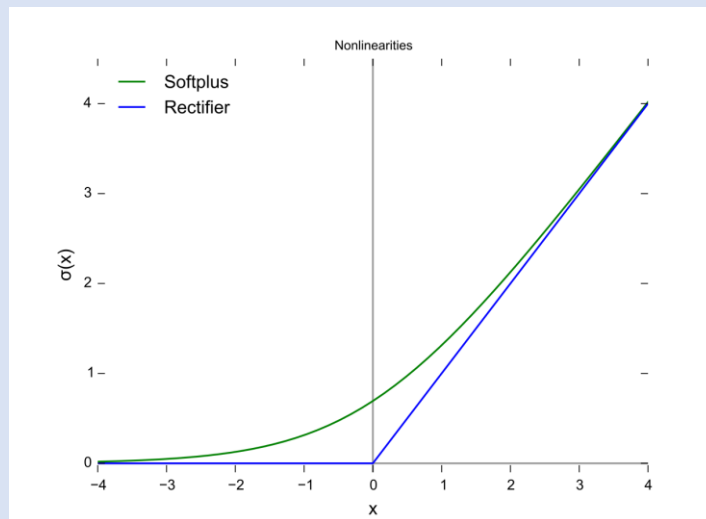


## 1. ¿Qué es AlexNet?

Dentro del **Deep Learning** nos encontramos con un tipo de redes neuronales llamadas Convolucionales. Las **redes convolucionales** son redes neuronales artificiales donde las neuronas corresponden a campos receptivos, simulando la corteza visual primaria del cerebro biológico. Estas redes son una variación del perceptrón multicapa comúnmente usado en todo el ámbito del *Machine Learning* y no solo en *Deep Learning*. Una de las principales diferencias de las redes convolucionales frente al perceptrón multicapa es que las primeras utilizan matrices bidimensionales lo que las hace especialmente útiles para la **clasificación y segmentación de imágenes**, entre otras aplicaciones

**AlexNet** es una red neuronal convolucional (CNN) diseñada por Alex Krizhevsky que participó en el concurso “Image Large Scale Recognition Challenge” en 2012. Esta red neuronal se clasifica en el top-5 de redes de mayor rendimiento consiguiendo un error de 15.3%.

AlexNet contiene 8 capas de neuronas. Las 5 primeras son capas convolucionales y la función de activación implementada es la función rectificador no saturada o “Non-saturating Relu” (ver imagen 1)



Gráfica de la función activadora utilizada en la red AlexNet

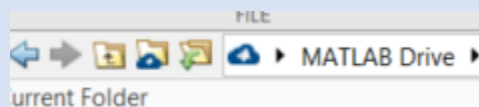


## 2. *Objetivo de la práctica 1*

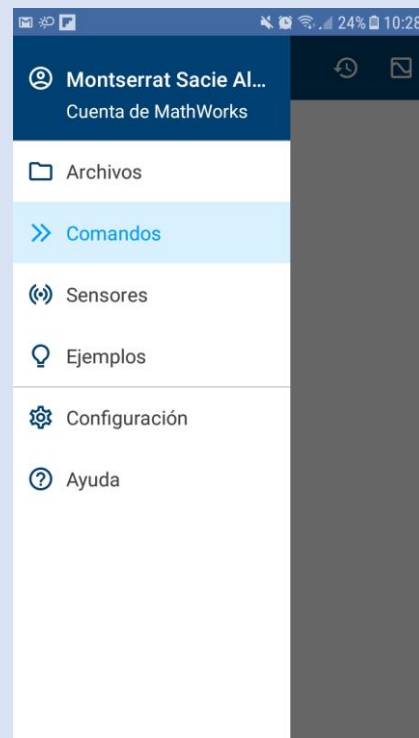
En esta primera práctica se pretende fotografiar objetos con la cámara de nuestro smartphone y **clasificar los los objetos de las imágenes obtenidas** haciendo uso de la red AlexNet.

## 3. *Manual de Instalación*

- 1) Instalar Matlab Drive Connector en nuestro Matlab en el PC



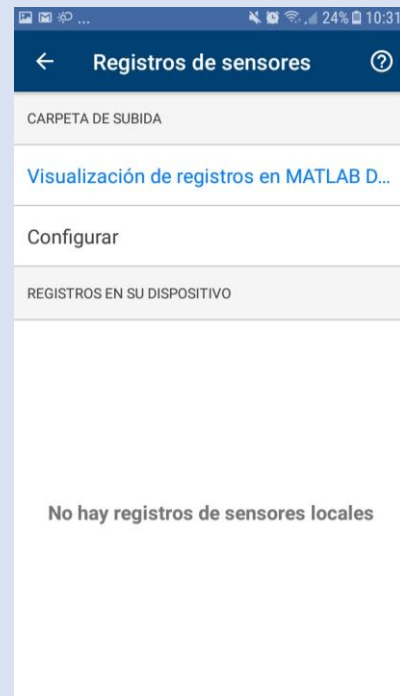
- 2) Instalar la app de Matlab Mobile
  - a. Introducir nuestra cuenta de Matlab



- b. Configuración de “Sensores”
      - i. Transmitir a -> Matlab (en tiempo Real)



- ii. En Registro de sensores -> entramos en Configurar y activamos la cámara






3) En nuestro Matlab (online o en versión de escritorio) instalar las siguientes *Toolboxes*:

*a. Image Processing*



*b. Computer Vision*







**Computer Vision Toolbox** R2020a by MathWorks


Design and test computer vision, 3D vision, and video processing systems

Computer Vision Toolbox™ provides algorithms, functions, and apps for designing and testing detection and tracking, as

-  **Video Labeler** - Label video for computer vision applications
-  **Image Labeler** - Label images for computer vision applications

MathWorks Toolbox


### c. Deep Learning



**Deep Learning Toolbox**

Create, train, and simulate shallow and deep learning neural networks

### d. AlexNet (incluida dentro de la Toolbox Transfer Learning)



**Transfer Learning** version 2019.6.2 by Kevin Chng

Transfer Learning of Pre-trained Neural Network or Imported ONNX Classification Model in GUI

model Pre-trained Neural Network available :1) alexnet2) googlenet(ImageNet)3) googlenet(Places365)4) resnet18 inceptionresnetv211) squeezeNet12) densenet20113)

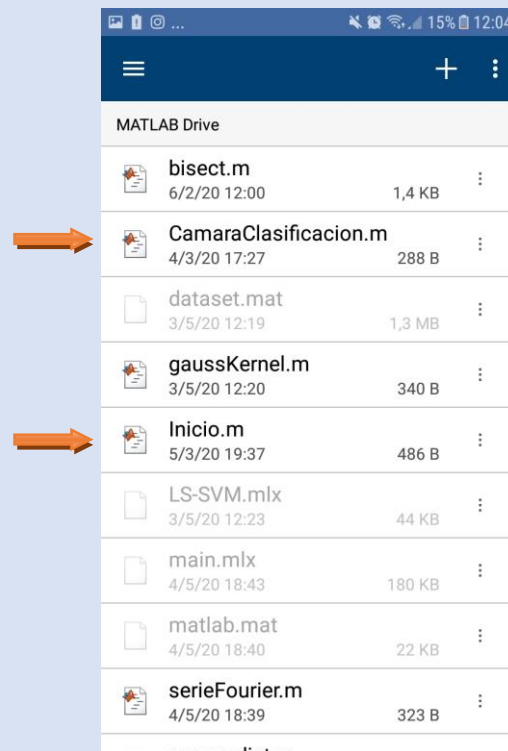
App





## 4. Ejecución del programa para clasificar objetos

Nuestro Malab en el PC y Matlab Móvil se comunican a través de Matlab Drive. Incluimos ahí los ficheros `Inicio.m` y `CamaraClasificacion.m`



Ya podemos ejecutar el programa desde la app móvil de Matlab, realizando los siguientes pasos:

- 1) El comando `>> Inicio` lo introducimos una vez, al inicio de la sesión
- 2) Cada vez que queramos hacer una foto con el móvil y ejecutar la red AlexNet para clasificar el objeto fotografiado introducimos el comando

```
Etiqueta = CamaraClasificacion(camara,RedAlex)
```

**Etiqueta** será la variable donde se guarde la categoría asignada por AlexNet al objeto fotografiado.

La red AlexNet puede clasificar un objeto en cualquiera de las **1000 categorías** con las que ha sido entrenada (Las hemos consultado en <https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a>).

Entre ellas nos encontramos con las etiquetas de los objetos que hemos fotografiado en los ejemplos de ejecución para comprobar el funcionamiento :



- **“Pekinese”** (ejemplo del enunciado, se ha tomado una foto a la imagen del perro facilitada junto al enunciado)
- **“ipod”**
- **“computer keyboard”** = teclado de ordenador
- **“dishwasher”** = trapo de cocina
- **“Pot”** = maceta
- **“cup”** = taza
- **“Coffeepot”** = cafetera

## *Ejemplos de Ejecución*

### Resultados

#### **Ejemplos en los que ha acertado al clasificarlos:**

- Pekinese
- Teclado de ordenador
- Maceta
- Taza
- Cafetera
- Ipod

#### **Ejemplos en los que ha fallado:**

- Trapo de cocina (lo ha etiquetado como “sobre” o “envelope”)
- Televisor (lo ha etiquetado como “vertical” o “upright”, no está mal del todo, pero no ha reconocido el objeto que es)
- Calculadora (lo ha etiquetado como teléfono o “cellular telephone”).



## Capturas de pantalla


```
>> Inicio

>> etiqueta =
    CamaraClasificacion(camara,RedAlex)

etiqueta =

    categorical

    computer keyboard
```




```
>>
```

```
>> etiqueta =
    CamaraClasificacion(camara,RedAlex)

etiqueta =

    categorical

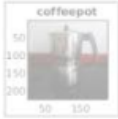
    Pekinese
```




```
>>
```



```
>> etiqueta =  
    CamaraClasificacion(camara,RedAlex)  
etiqueta =  
  
    categorical  
  
    coffeepot
```



```
>> etiqueta =  
    CamaraClasificacion(camara,RedAlex)  
etiqueta =  
  
    categorical  
  
    pot
```





```
>> etiqueta =  
    CamaraClasificacion(camara,RedAlex)  
etiqueta =  
  
    categorical  
  
    iPod  
  
    iPod  
    50 100 150 200  
    50 150  
  
>> etiqueta =  
    CamaraClasificacion(camara,RedAlex)  
etiqueta =  
  
    categorical  
  
    envelope  
  
    envelope  
    50 100 150 200  
    50 150
```

Nota: \*(Las imágenes se ven más turbias porque se hicieron las capturas de pantalla al final y por tanto a continuación de estos ejemplos de ejecución había más y solo el resultado del último comando ejecutado en la ventana de comandos se ve nítido)



## 5. Conclusiones

Tras la ejecución de la red AlexNet desde Matlab, hemos visto cómo esta es capaz de clasificar objetos que podemos fotografiar nosotros desde la cámara de nuestro móvil.

Por otro lado, debemos considerar que cuando nosotros hacemos una fotografía desde nuestro dispositivo, puede haber factores que afecten a la efectividad de la red en la clasificación; como por ejemplo si aparecen varios objetos en la foto o si la luz de la habitación no es suficiente y la foto sale turbia.

También hay que tener en cuenta que la red cometerá errores cuando reciba una imagen de un objeto cuya “clase” no ha sido incluida en ninguno de los ejemplos del *Training set*. Como nosotros no hemos entrenado la red puede pasar, por ejemplo, que justo la raza de nuestro perro no sea detectada por la red porque “no la conoce” (no pertenece a esas 1000 categorías con las que ha sido entrenada). Sin embargo, aun conociendo su margen de error, es sorprendente su *performance* y puede ser interesante recorrer el código que se ha utilizado para su implementación.



# *PRÁCTICA 2:*

## *Algoritmo DeepDream*



1. *El algoritmo DeepDream*
2. *Objetivo de la práctica 2*
3. *Ejecución del programa*
4. *Conclusiones*





## 1. El algoritmo DeepDream

**Deep Dream** es un algoritmo de procesamiento de imágenes creado por Google. Fue creado para el *imageNet large scale visual recognition challenge* (ILSVRC). Este reto iba dirigido a diferentes equipos de investigación que fueran capaces de crear un sistema de reconocimiento de objetos a través del aprendizaje de los patrones de una imagen.

El algoritmo Deep Dream se puede aplicar con la red convolucional AlexNet en Matlab, permitiendo sintetizar una imagen y visualizar la información contenida en cada capa de la red, así como las características aprendidas sobre la imagen capturada. Esto nos permite crear efectos sobre la imagen y ver las imágenes alteradas. Dichas imágenes son útiles para entender y diagnosticar el comportamiento de la red.



*Obra de arte original, la "Gioconda"*



*Imagen sintetizada por una CNN mediante el algoritmo DeepDream.*

*Imágenes obtenidas de Google Images*

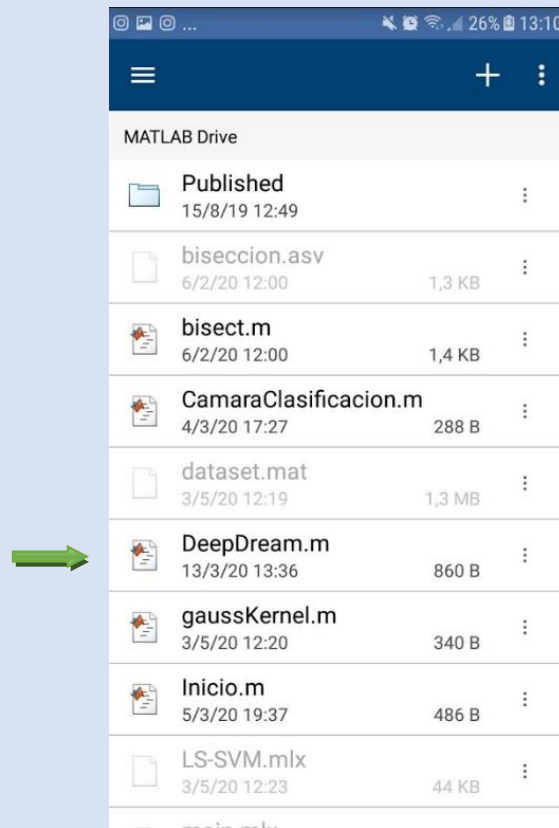
## ***2. Objetivo de la práctica 2***

El objetivo de esta práctica es utilizar el móvil para capturar una imagen y aplicar el algoritmo DeepDream mediante la Red Convolucional AlexNet. Así veremos la información contenida en las capas de la red proyectada sobre la imagen capturada.

El manual de instalación para poder ejecutar esta práctica y la siguiente es el mismo que el de la práctica 1, por lo que si se ha instalado previamente podemos pasar directamente a la ejecución.

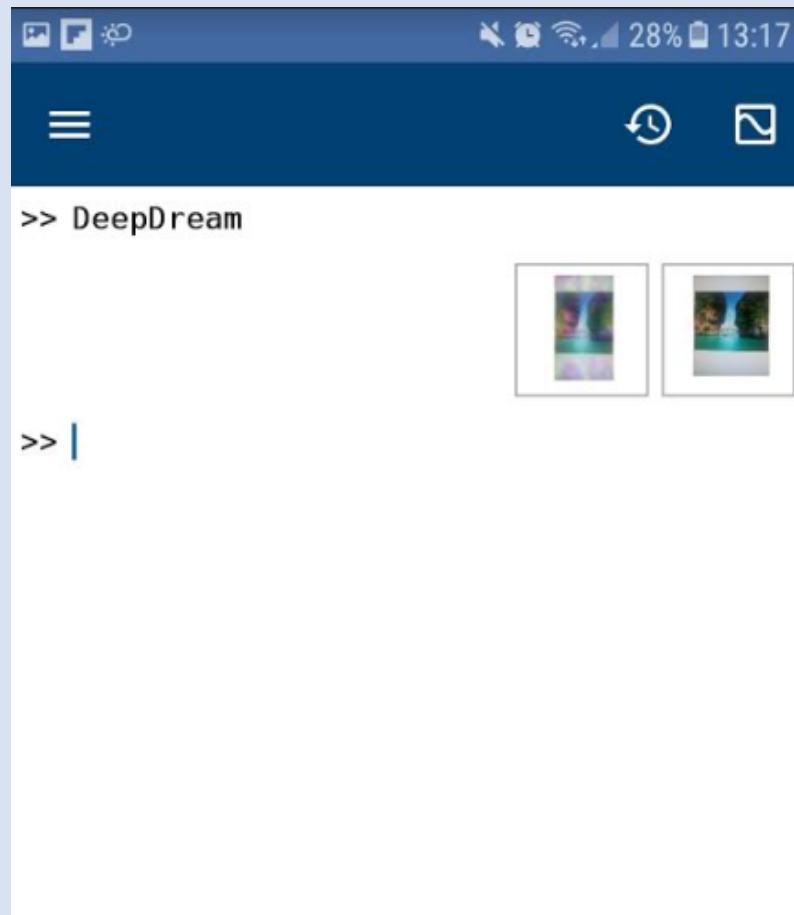
## ***3. Ejecución del programa***

Lo primero que debemos hacer es copiar el fichero `DeepDream.m` en la carpeta de Matlab Drive.



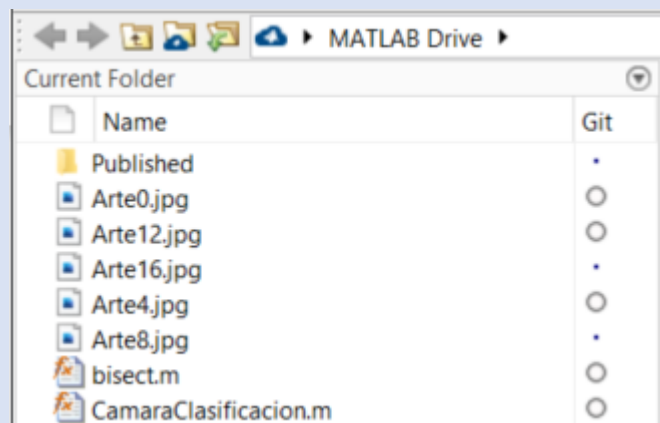
A continuación en la ventana de comandos introducimos el comando `DeepDream`.

Esto activa la cámara de nuestro móvil permitiéndonos tomar una fotografía sobre la que se realizarán los “efectos especiales” configurados. Tomamos como ejemplo la imagen facilitada junto al enunciado



Tras la ejecución del algoritmo, se guardan en la carpeta en la que estamos las imágenes generadas: Arte0.jpg, Arte4.jpg, Arte8.jpg, Arte12.jpg y Arte16.jpg

Podemos acceder a estas fotos desde el Matlab de escritorio:

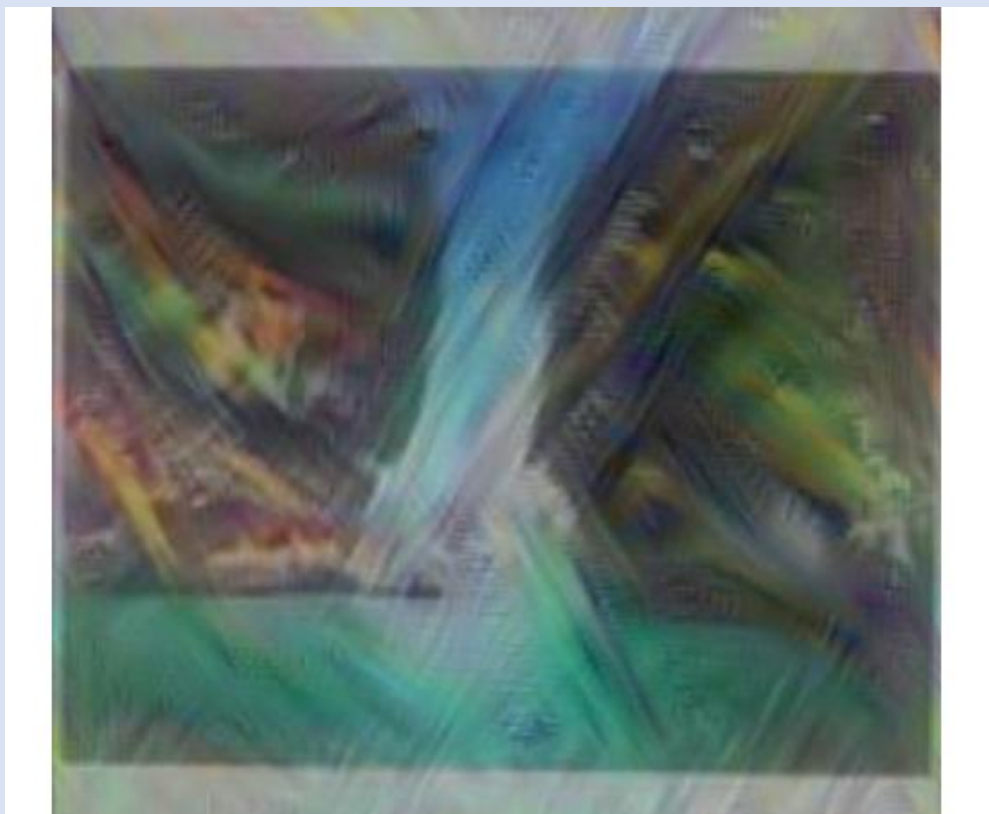




La imagen original es:

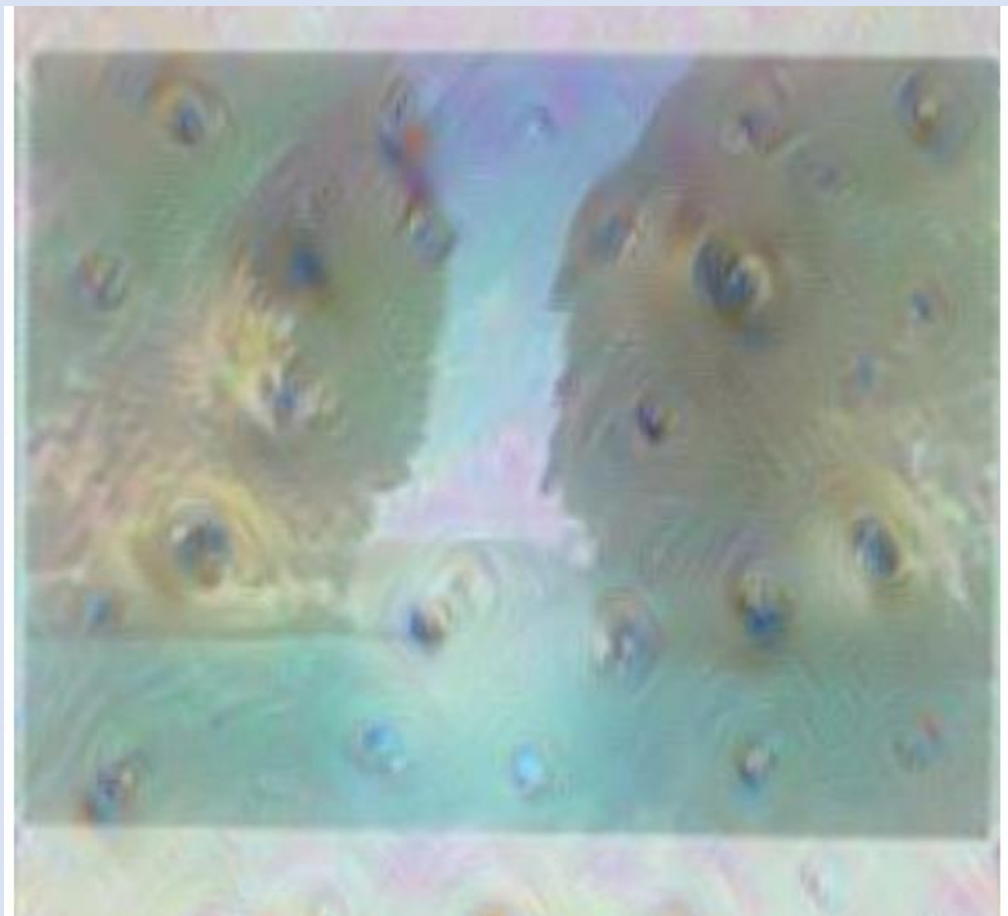


La calidad de las imágenes generadas es menor ya que se ha tomado una foto con la cámara del móvil a la pantalla del ordenador donde teníamos la imagen original abierta.

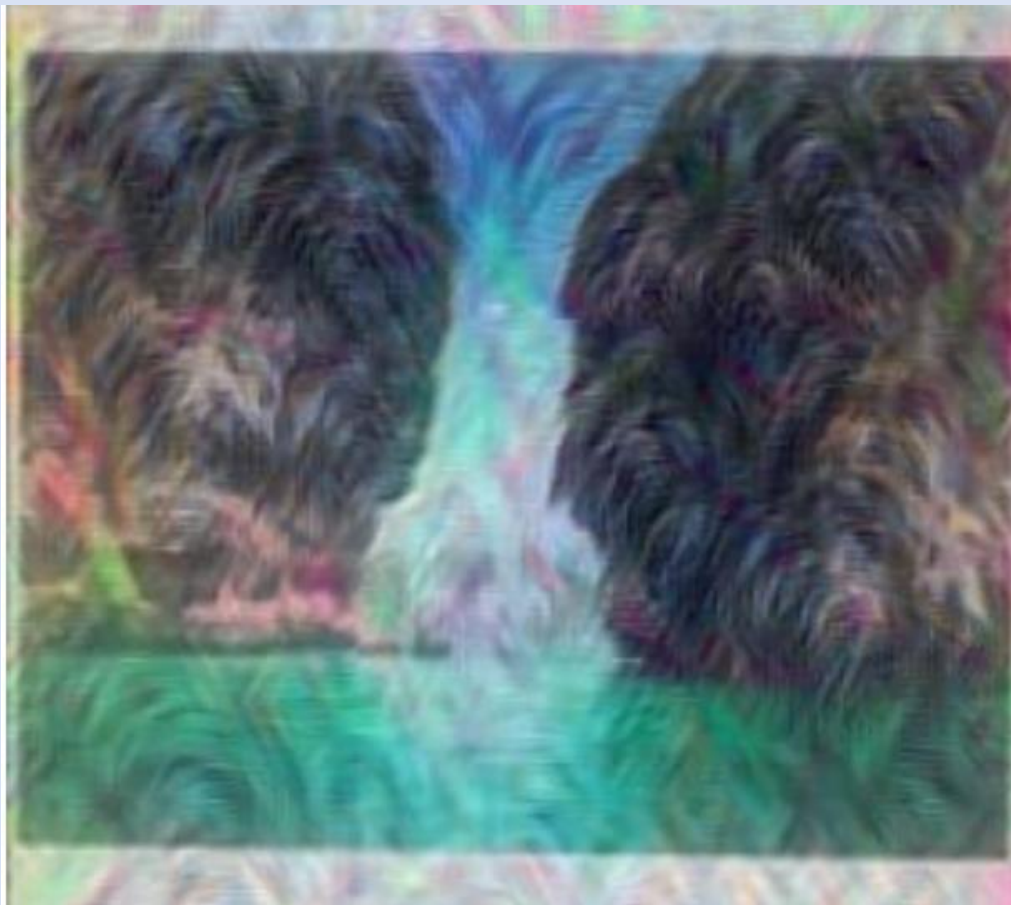


Arte0.jpg





Arte4.jpg



Arte8.jpg





Arte12.jpg



Arte16.jpg

#### 4. Conclusiones

El algoritmo *DeepDream* nos ofrece un abanico de posibilidades para generar efectos muy llamativos sobre imágenes. El arte se está digitalizando como vemos hoy día y cambiando la configuración del algoritmo empleado o de la información que queremos ver sobre cada imagen, podemos obtener los efectos deseados y modificar una fotografía de forma rápida y sencilla desde nuestro móvil.

Esta aplicación de la red AlexNet es también muy interesante como hemos podido ver aquí en una sencilla aplicación y merece la pena explorarla para aplicarla como deseemos.



## *PRÁCTICA 3:*

*Detección de Acciones de  
Movimiento con redes  
Long Short-Term  
Memory*



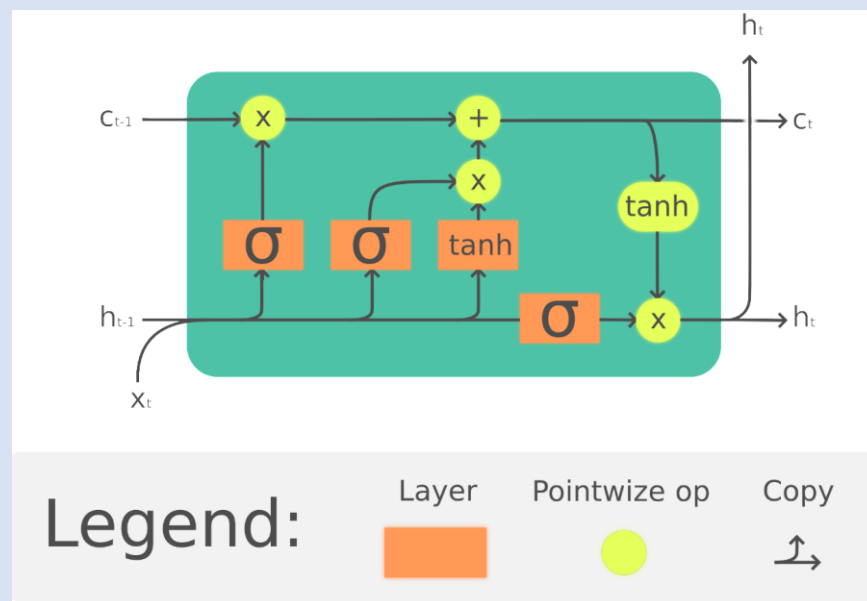
1. *¿Qué son las redes LSTM?*
2. *Objetivo de la práctica 3*
3. *Ejecución del programa*
4. *Conclusiones*



## 1. ¿Qué son las redes LSTM?

**Long short-term memory o LSTM** es una red neuronal recurrente (RNN) utilizada en *Deep Learning* que contiene conexiones entre neuronas “hacia atrás”. Estas redes no procesan puntos de datos (como un ejemplo de entrenamiento dado a una red neuronal *feedforward*) sino que *procesa secuencias de datos completas (como un video)*.

Las **neuronas** artificiales que forman una LSTM son capaces de procesar datos de forma secuencial y mantener su estado oculto a través del tiempo.



*Esquema del procesamiento realizado por una Neurona LSTM*



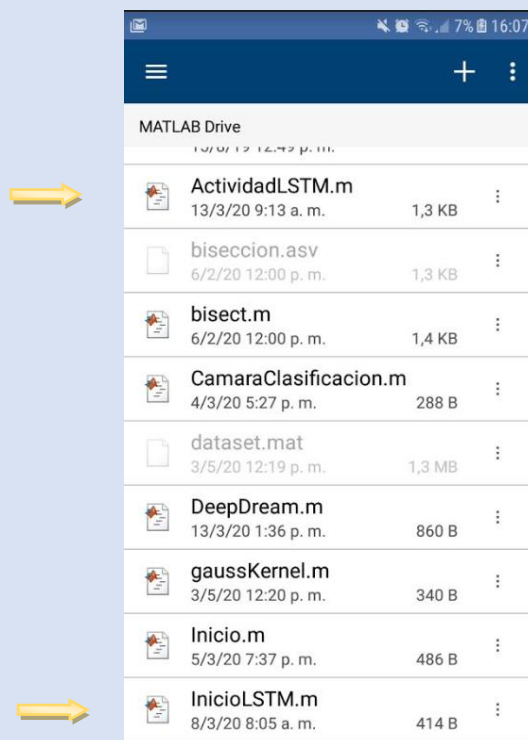
## 2. *Objetivo de la práctica 3*

En esta práctica se va a hacer uso de una red LSTM en la aplicación móvil de Matlab para clasificar acciones o secuencia de movimientos, siendo estos los datos percibidos por los sensores integrados que los envían a la red. La red está pre-entrenada para clasificar las acciones:

- **Dancing**
- **Running**
- **Sitting**
- **Standing**
- **Walking**

## 3. *Ejecución del programa*

En primer lugar, incluimos en la carpeta de Matlab Drive los ficheros InicioLSTM.m y ActividadLSTM.m y el fichero RedLSTM.m que contiene el modelo de la red pre-entrenada.



MATLAB Drive			
	<b>ActividadLSTM.m</b>	13/3/20 9:13 a. m.	1,3 KB
	biseccion.asv	6/2/20 12:00 p. m.	1,3 KB
	bisect.m	6/2/20 12:00 p. m.	1,4 KB
	CamaraClasificacion.m	4/3/20 5:27 p. m.	288 B
	dataset.mat	3/5/20 12:19 p. m.	1,3 MB
	DeepDream.m	13/3/20 1:36 p. m.	860 B
	gaussKernel.m	3/5/20 12:20 p. m.	340 B
	Inicio.m	5/3/20 7:37 p. m.	486 B
	<b>InicioLSTM.m</b>	8/3/20 8:05 a. m.	414 B



MATLAB Drive		
3/5/20 12:20 p. m.		
340 B		
Inicio.m	5/3/20 7:37 p. m.	486 B
InicioLSTM.m	8/3/20 8:05 a. m.	414 B
LS-SVM.mlx	3/5/20 12:23 p. m.	44 KB
main.mlx	4/5/20 6:43 p. m.	180 KB
matlab.mat	4/5/20 6:40 p. m.	22 KB
RedLSTM.mat	19/9/19 7:18 p. m.	108 KB
serieFourier.m	4/5/20 6:39 p. m.	323 B
svr_predict.m	3/5/20 12:19 p. m.	486 B
svr_trainer.m	3/5/20 12:19 p. m.	2,8 KB

En la app del móvil activamos los sensores de Aceleración y Posición (sensores disponibles en el dispositivo móvil utilizado para desarrollar esta práctica).

Tasa de muestreo	10,0 Hz
Más	
SENSORES	
Aceleración	<input checked="" type="checkbox"/>
X m/s <sup>2</sup>	0,842
Y m/s <sup>2</sup>	4,524
Z m/s <sup>2</sup>	8,951
Campo magnético	<input type="checkbox"/>
X μT	---
Y μT	---
Z μT	---
EMPEZAR	
Orientación	



Una vez activados los sensores, ejecutamos los siguientes comandos:

```
>> InicioLSTM

>> ActividadLSTM

Index in position 2 exceeds array
bounds.

Error in ActividadLSTM (line 22)
yAngVel = vangular(:,2);

>> ActividadLSTM

...
h 157

Walking Walking Walking
Columns 158 through 160
Walking Walking Walking
Columns 161 through 163
Walking Walking Walking
Columns 164 through 166
Walking Walking Walking
Columns 167 through 169
Walking Walking Walking
>>
```

Como vemos, nos aparece en el primer intento de ejecutar el fichero `ActividadLSTM.m` un error. Esto se debe a que en el código se intenta acceder a los datos obtenidos por el **sensor de Velocidad Angular y el sensor de orientación (Roll)** para representar su evolución frente al tiempo en un plot. Como **no se ha podido activar estos sensores** ya que no están disponibles en el móvil, se estaba intentando acceder a una matriz de datos nula.

Para evitar este error, comentamos las líneas de código que acceden a estos datos y realizan el plot.

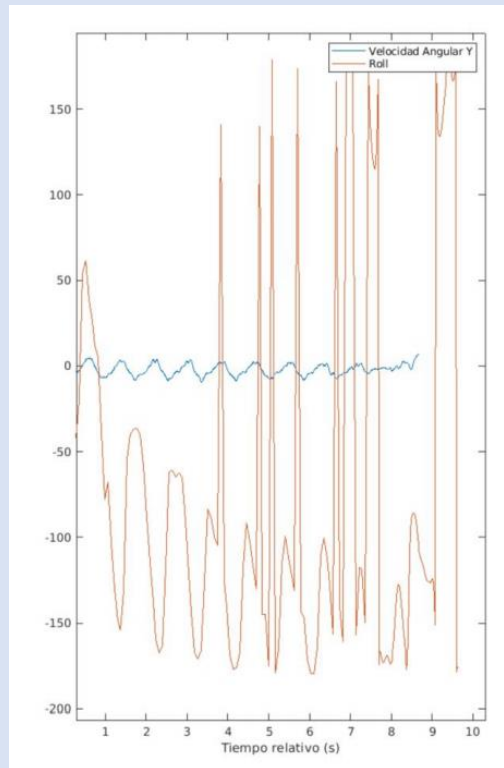




Código  
comentado  
adicional

```
MATLAB Drive > ActividadLSTM.m
11
12 %% Desactivar la captura
13 m.Logging = 0;
14
15 %% Recuperar los datos almacenados
16 [aceleracion, taceleracion] = accellog(
17 %%[vangular, tvangular] = angvellog(m);
18 %%[orientacion, torientacion] = orientl
19 %%[magenetico, tmagnetico] = magfieldlo
20 [posisicion, tposicion] = poslog(m);
21
22 %%yAngVel = vangular(:,2);
23 %%roll = orientacion(:, 3);
24 %%legend('Velocidad Angular Y', 'Roll')
25 %%plot(tvangular, yAngVel, torientacion
26 %%legend('Velocidad Angular Y', 'Roll')
27 %%xlabel('Tiempo relativo (s)');
28
29
30 %% Clasificaci3n de la acci3n mediante
31 [M,N] = size(aceleracion);
32
33 if M == 0
34     disp('No se han capturado datos: repe
35 else
36     X = aceleracion';
37     Actividades = classify(RedLSTM,X)
38 end
39
40 discardlogs(m);
41
42
```

Si estos sensores estuvieran disponibles, obtendríamos una gráfica similar a esta:



Durante la ejecución de `ActividadLSTM.m` (10 segundos de espera que vienen dados por la instrucción `pause(2)` incluida en el código 5 veces) se ha andado por la habitación, sentado y bailado. Durante esos segundos de espera los sensores han estado recopilando datos.

El resultado obtenido es este:



Columns 212 through 214		
Sitting	Sitting	Sitting
Columns 215 through 217		
Sitting	Sitting	Sitting
Columns 218 through 219		
Sitting	Sitting	
Columns 220 through 221		
Standing	Running	
Columns 222 through 224		
Running	Running	Running
Columns 225 through 227		
Dancing	Dancing	Dancing
Columns 228 through 230		
Dancing	Dancing	Dancing
Columns 231 through 233		
Dancing	Dancing	Dancing
Columns 234 through 236		
>>		



```
Columns 2039 through 2041
    Sitting    Sitting    Sitting
Columns 2042 through 2044
    Sitting    Sitting    Sitting
Columns 2045 through 2047
    Sitting    Sitting    Sitting
Columns 2048 through 2050
    Sitting    Sitting    Sitting
Columns 2051 through 2053
    Sitting    Sitting    Sitting
Columns 2054 through 2056
    Sitting    Sitting    Sitting
Columns 2057 through 2059
    Sitting    Sitting    Sitting
La salida devuelta supera la longitud
mxima. Se muestran resultados
parciales.
>>
```

## 4. Conclusiones

La funcionalidad de clasificar el movimiento a partir de unos sensores que miden variables como la velocidad o la localizaci3n GPS, probada en esta prctica, resulta muy interesante y nos da pie a pensar en las posibles aplicaciones que pueden tener estas redes LSTM en la vida real.

En este caso, hemos comprobado que la red clasifica correctamente el tipo de movimiento en base a la aceleraci3n y la posici3n nicamente. Sin embargo, esta red puede tener errores derivados de la falta de ms variables; por ejemplo: una persona que est parada, pero en pie ser clasificada como "sentada".



La red podría ser entrenada con más variables a considerar y clasificar el movimiento en más estados aumentando así su nivel de complejidad y ampliando las posibilidades de actuación.