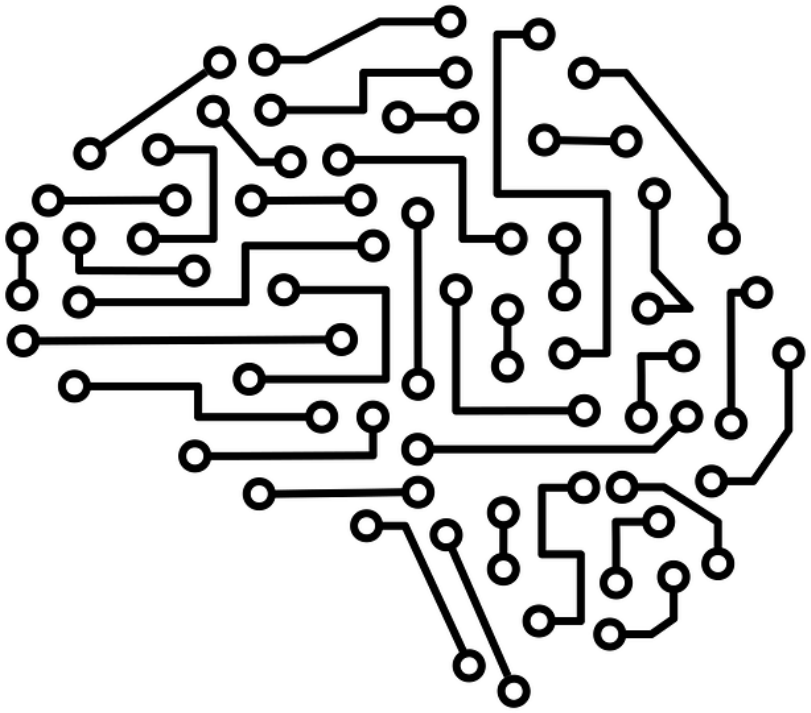
Deep Learning para la identificación de sujetos con mascarilla.



Juan Francisco Martínez Resola

Objetivos:

El objetivo principal es utilizar los medios que la inteligencia artificial nos ofrece con el fin de aplicarla al reconocimiento de imagen, en este caso para el reconocimiento de un sujeto e identificar si este lleva o no la mascarilla puesta, es decir para que decida si en una imagen hay personas con mascarilla o sin mascarilla.

Además otro objetivo será la realización del código base con el que poder entrenar de forma relativamente sencilla casi cualquier modelo.

Para ello sentaremos la premisa de utilizar en todo momento software libre, viendo así como herramientas con el código liberado tienen un gran potencial.

Introducción.

Antes pasaremos a explicar los componentes que vamos a utilizar.

1) Que es la *inteligencia artificial*, y como funciona en la clasificación de imágenes:

   En resumen, podemos decir que una máquina dotada de inteligencia artificial es aquella máquina inteligente capaz de maximizar sus probabilidades de éxito en una tarea no acotada de una forma precisa.

   En nuestro caso, para ser más concretos nos ayudaremos de la inteligencia computacional, una rama de la inteligencia artificial, en ella es en la que se aplican técnicas como las redes neuronales, computación evolutiva etc.

   Combina las técnicas de adaptación, evolución y lógica difusa para crear métodos de computación que son en cierto modo inteligentes, con el fin de ser utilizados como cajas a las que introducimos un parámetro (en nuestro caso una imagen) y obtenemos a su salida otro (en nuestro caso el porcentaje de que en esa foto la persona lleve una mascarilla).

2) Qué son las redes neuronales artificiales.

   Una red neuronal es un entramado de neuronas que en conjunto forman un proceso matemático para la construcción de un modelo inteligente.

   Una neurona la podemos definir como una función, a la cual llegan unos parámetros de entrada y mediante unas operaciones que tienen lugar en su interior tiene como salida un parámetro, el cual puede ir directo a la neurona de salida de la red neuronal, o a la entrada de la siguiente capa de neuronas.

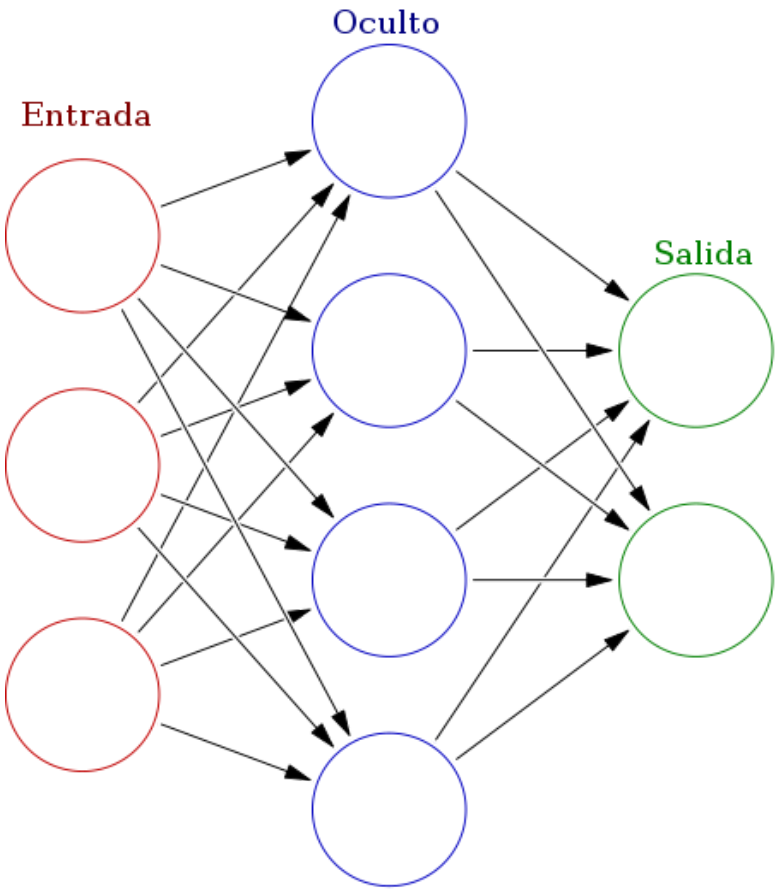
Cada entrada a una neurona está conectada a la salida de otras, los parámetros de entrada que reciben son multiplicados por el valor que tiene el peso de esta estos parámetros, pueden incrementar o decrecer, de igual modo a la salida posee una función limitadora, que delimita el poder de dicha neurona sobre las siguientes.

Pero estos parámetros no se ajustan manualmente, ya que de ser asi seria una programación muy compleja y tediosa, si no que son entrenados mediante el aprendizaje automático, por el cual asigna mayor o menor peso a cada uno, se hace de modo que se intente minimizar la función de pérdida, de este modo los valores de los pesos de las neuronas se van actualizando, mediante la técnica de propagación hacia atrás.

Es decir los parámetros de peso de cada neurona se ajustan mediante el proceso de entrenamiento, mediante el cual se fija, y posteriormente, se utiliza este valor en esta red para ser utilizado.

Estos procesos son los llamado lógica difusa, que no es más que una lógica no binaria, es decir no existen verdaderos o falsos, si no que pertenecen a diferentes áreas de probabilidad, la clásica analogía, es la de la sensación térmica, no siempre es frío o calor y entre ellas existe un amplio rango, del mismo modo que en la respuesta de un modelo inteligente.

El comportamiento de las redes neuronales artificiales, es muy similar al de los humanos, ya que aprendemos por patrones y tenemos una etapa de aprendizaje en la que generamos un modelo para posteriormente hacer uso del mismo.



Tenemos una capa de entrada, una de salida, y entre ambas nos encontramos una cantidad variable de capas ocultas, las cuales pueden organizarse de muchas formas diferentes, con saltos de capas, con capas de reducción de una sola neurona, etc.

3) ¿Qué es el deep learning?

Lo podemos definir como diferentes técnicas matemáticas empleadas en el aprendizaje automático, por las cuales mediante el análisis de los datos dados por el usuario (imagenes, audio, texto etc) generamos un modelo, existen diferentes metodologías como pueden ser cascadas de capas de neuronas, de forma jerárquica en la cual se heredan las características, a múltiples niveles etc.

Es una subcategoría del Machine learning, que en resumen se trata de sistemas de aprendizaje mediante una gran cantidad de datos y diferencia que marca el Deep Learning es su completa autonomía, solo es necesario proporcionarle los datos y trabaja por sí sola, mientras que en el resto necesitan un estudio previo y el planteamiento de un árbol de decisiones.

Haciendo una analogía con el ser humano, se trataría de un profesor (entrenador del modelo) dar a una persona (modelo) una gran cantidad de imagenes de cada tipo, el cual

debe estar especificado, que queremos diferenciar, por ejemplo una carpeta con imágenes de cuadrados y otra con imágenes de triángulos, y posteriormente realizar un test para saber su acierto.

Mediante un proceso de aprendizaje (algoritmos de deep learning) pasado un tiempo, esta persona será capaz de diferenciar entre imágenes diferentes a las estudiadas, haciendo uso de patrones mediante los cuales podrá diferenciar si se trata de una u otra, y para comprobar que realmente han aprendido a diferenciar, se someten a tests con imágenes diferentes, pero del mismo tipo, para evaluar su porcentaje de acierto, así el mismo sistema tiene un feedback.

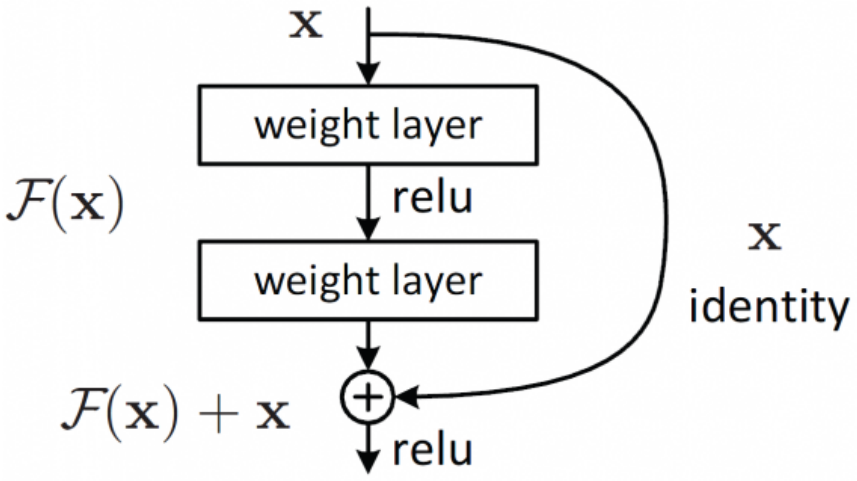
Es decir en ningún caso aprende por relaciones lógicas estudiadas, como puede ser el estudio del área, reconocimiento de curvas y formas mediante sus parámetros de dimensiones, sino por patrones.

Como vemos es muy similar al aprendizaje humano, aprendizaje por patrones y test.

Son procesos muy largos, en los que es requerido un buen equipo con una alta potencia computacional para ser capaz de realizar todas las operaciones.

4) En qué consiste el modelo ResNet

Su nombre proviene de “Residual Network”, se trata de una red de neuronas artificiales, lo que tiene de especial es que no se tratan de estructuras lineales convencionales en la cual las salidas y entradas solo provienen de la capa anterior, si no que da saltos entre capas, esto se hace para mitigar el efecto del cambio en los gradientes o pesos, en definitiva trata de simplificar la red neuronal, y va incorporando las capas a medida que va aprendiendo.



Construcción y uso del modelo.

Una vez sabiendo los componentes que vamos a utilizar, pasamos analizar los pasos que tomaremos para crear nuestro sistema.

1) Selección de software que utilizaremos.
2) Preparación de archivos
3) Entrenamiento bajo deep learning del modelo.
4) Uso del modelo para clasificación.

Selección del software.

-¿Qué lenguaje utilizaremos?

Para este trabajo hemos empleado como lenguaje principal Python, en su versión 3.7.6 de 64 bits.

-¿Por qué?

Es el principal lenguaje utilizado en el ámbito de inteligencia artificial, por su facilidad y sencilla lectura, y ya que es el más utilizado, es más fácil encontrar librerías e información al respecto.

-Librerías utilizadas

Como dependencia principal, hemos utilizado *imagenAI*, la cual es una librería creada para desarrolladores, investigadores y estudiantes con el fin de ser utilizada en sistemas autónomos de deep learning y visión por computador, con un principal objetivo, y es que su uso sea muy sencillo, esta librería no trabaja por sí sola, si no que es dependiente de las siguientes:

Tensor Flow: es la principal biblioteca de código abierto para el desarrollo de modelos de aprendizaje automático, es desarrollada por Google.

Keras: Basada en Python, de código libre, esta se encargará de entrenar al modelo mediante imágenes.

OpenCV: Open computer vision es la principal librería de software libre de visión artificial, esta se encarga de leer y procesar las imágenes, inicialmente desarrollada por Intel en 1999, entre sus aplicaciones cuenta con 2D y 3D, reconocimiento facial, de objetos, segmentación, tracking etc.

Preparación de archivos

Antes debemos preparar el entorno con los archivos necesarios para ello crearemos las siguientes carpetas:

/imágenes/ train/ opción A
opción B
.
.

/imágenes/ test/ opción A
opción B
.
.

En la carpeta train tendremos tantas carpetas como casos de diferenciación que necesitemos, esta carpeta será utilizada para la creación del modelo, según las documentación oficial, es necesario el uso de un mínimo de 500 imágenes, y recomiendan un máximo de 1000.

En la carpeta test, tendremos tantas carpetas como casos de diferenciación que necesitemos, esta carpeta será utilizada para la validación y comprobación del modelo, para este caso se recomienda entre 100 y 200 imagenes, las cuales deben ser imagenes diferentes a las utilizadas en la carpeta train.

Entrenamiento del modelo bajo deep learning.

Crearemos un script para la creación del modelo:

```
from imageai.Classification.Custom import ClassificationModelTrainer

entrenador = ModelTraining()

entrenador.setModelTypeAsResNet()

entrenador.setDataDirectory("ruta_imagenes")

entrenador.trainModel(num_objects=2, num_experiments=100, enhance_data=True,
batch_size=32, show_network_summary=True)
```

Los parámetros que debemos especificar son los siguientes:

-num_objects → Cantidad de diferentes modelos a diferenciar, 2 en nuestro caso.

-num_experiments → Número de interacciones de entrenamiento que queremos que ejecute nuestro algoritmo, a medida que se generan no mejoran en gran cantidad, o puede

incluso que la mejoría sea nula ya que la esta tiene un límite, pero esto depende de la calidad de las imagenes y del tipo de experimento.

-enhance_data → si configuramos a true este dato, a partir de nuestras imagenes (muestras), se generarán nuevas muestras mediante la modificación de las mismas, es decir invirtiendo, recortando, cambiando colores etc, es muy recomendable cuando las muestras de las que disponemos son limitadas.

-batch_size → Numero de imagenes a analizar de forma paralela, esto tiene efecto en la velocidad del entrenamiento.

-show_network_summary → Al estar en true podremos ver mensajes por pantalla de cómo se desarrolla la creación de nuestro modelo.

Ya está listo para ejecutar el programa, como hemos dicho, aunque la teoría y base de funcionamiento es compleja y es necesario conocerla, la escritura es muy simple y directa.

Mientras se está ejecutando el programa, podremos ir viendo por pantalla que tal va el proceso, si sube o baja su precisión, podremos ver las siguiente información:

Número del experimento, tiempo estimado y precisión ganada y perdida.

Analizando la salida de datos podemos observar lo siguiente:

Interacción	Precisión
1	62.29%
10	81.97%
20	91.26%
30	94.43%
40	94.21%
50	90.10%
60	99.12%
70	89.07%
80	99.60%
90	89.06%
100	89.58%

Como podemos observar entre las primeras interacciones conseguimos una mejoría sustancial, sin en cambio en las últimas una mejoría ínfima o casi nula, y en algunos casos incluso perdemos precisión, esto es debido al propio funcionamiento del modelo ResNet, ya que va implementando nuevas capas, las cuales hace que tenga un mayor potencial de ser más preciso, pero a la vez lo hace más complejo y por ello pierde precisión hasta que consigue encajar los parámetros correctos.

Una vez terminado el entrenamiento nos generara varios archivos, los más importantes son:

imagenes/json/model_class.json

Archivo json: es un formato de texto sencillo muy usado en JavaScript, en él encontraremos la relación entre varios elementos, en nuestro caso encontraremos el siguiente texto:

```
{
    "0" : "conmasc",
    "1" : "sinmasc"
}
```

Esto significa que, nuestro modelo tendrá como “salida” un 0 en el caso de detectar una foto de una persona con mascarilla y un 1 si la detecta sin ella, tiene el mismo nombre que las carpetas creadas.

imagenes/models/model_ex_-<numero de iteraccion>_acc-<precisión del modelo>.h5

Se trata de los archivos correspondientes a cada modelo generado en cada interacción, tiene un tamaño menor a 100mb y solo con él seremos capaces de utilizar de utilizar e implementar nuestro modelo en diferentes aplicaciones.

Uso del modelo

Una vez entrenado, pasaremos a utilizar el modelo en un pequeño script de prueba, al cual le pasaremos una imagen y nos mostrará mediante CLI (command line) mediante texto que ha encontrado en dicha imagen, para ello utilizaremos el siguiente código:

```
#Importamos la libreria
from imageai.Classification.Custom import CustomImageClassification

#Configuramos el predictor indicando que es un modelo custom y no de los
que ya posee la propia librería
prediccion = CustomImageClassification()
```

```
#Metodología que emplea nuestro modelo
prediccion.setModelTypeAsResNet()

#Directorio completo o relativo en el que se encuentra nuestro modelo
con extensión h5
prediccion.setModelPath("path.h5")

#Directorio completo o relativo donde se encuentra nuestro archivo con
extensión JSON
prediccion.setJsonPath( "path.json"))

#Número de modelos diferentes
prediccion.loadModel(num_objects=2)

#Cargamos y dejamos que nuestro modelo estudie la imagen
predictions, probabilities =
prediction.classifyImage("path_imagen.formato_imagen", "4.jpg"),
result_count=2)

#Aclarar que puede hacer varias predicciones, es decir si en la misma
imagen aparece una persona con y otra sin mascarilla, debería detectar
un (casi) 100% de cada uno.

#Recorre ambos arrays y los imprime por pantalla.
#la función zip une dos array convirtiéndolo en una tupla
for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , " : " , eachProbability)
```

Testeo

-Con mascarilla, simple.

Descargamos una foto de internet de una persona con mascarilla, en concreto la siguiente:

Es una foto clara, en la que se aprecia la mascarilla en gran medida, ejecutamos el programa y obtenemos la siguiente salida:

```
conmasc  :  95.4566240310669
sinmasc  :  4.543376341462135
```

En más de un 95% de probabilidad se trata de una persona con mascarilla, lo cual está en lo cierto, si el fondo fuese plano, probablemente este porcentaje subiría, vamos a probar el caso contrario, una persona sin mascarilla, y además con un fondo plano.

-Sin mascarilla, simple:

Una imagen muy clara, sin fondo, podemos esperar una respuesta con una probabilidad alta, de no llevarla, y como vemos obtenemos este resultado:

```
sinmasc  :  99.09499287605286
conmasc  :  0.9050123393535614
```

Más de un 99% de que no la lleva, un dato muy alto.

Ahora vamos a ver qué pasa si la imagen que utilizamos lleva algún elemento que pueda confundir nuestro modelo como por ejemplo, una persona sin mascarilla y con barba frondosa.

-Barba frondosa

Utilizamos la siguiente imagen:

Con la que obtenemos el siguiente resultado:

```
sinmasc  :  50.966328382492065
conmasc  :  49.033668637275696
```

Como vemos en este caso la percepción del modelo cambia, ya que la máquina no es inteligente, solo es capaz de seguir patrones, y en este caso el patrón de la barba y la mascarilla se acerca, ya que cubre la misma parte de la cara.

Para ello vamos a realizar una prueba más en este sentido, vamos a hacer el test con una foto de una persona con barba pero sonriendo, con el fin de que el área de imagen que ocupa la boca sea mayor.

-Barba y sonrisa.

Y obtenemos:

```
sinmasc  :  99.85760450363159
conmasc  :  0.14239867450669408
```

Podemos concluir entonces, que el problema es el que nombramos, patrones similares.

Ahora vamos a tratar de confundirla, vamos a utilizar una fotografía de una persona con mascarilla, pero esta lleva un estampado de boca y nariz, simulando que no la lleva.

-Mascarilla con estampado de boca y nariz.

```
conmasc  :  62.02830672264099
sinmasc  :  37.97169029712677
```

Aunque la engañamos un poco, es capaz de reconocer en mayor medida que si lleva mascarilla, esto probablemente se deba a las tiras elásticas que se aprecian en la imagen.

Descarga:

Importante:

1) Usar Python en su versión 3.7 de 64 bits.
2) Descomprimir e instalar los paquetes mediante pip, instalar requierements.txt