laSalle
UNIVERSITAT RAMON LLULL

Escola Universitària d'Enginyeria Tècnica de Telecomunicació La Salle

Trabajo Final de Máster

Máster Universitario en Ciencia de Datos

Utilización de redes neuronales para la detección de asteroides

Alumno

José Miguel Val Serra

Profesor Ponente

Xavier Vilasís Cardona

ACTA DEL EXÁMEN
DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en el día de la fecha, el alumno

D. **José Miguel Val Serra**

expuso su Trabajo Final de Máster, el cual trató sobre el tema siguiente:

Utilización de redes neuronales para la detección de asteroides

Finalizada la exposición y atendidas por parte del alumno las objeciones formuladas por los Srs. miembros del tribunal, éste valoró el mencionado Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

UNIVERSIDAD RAMON LLULL

TFM:

MD10 UTILIZACION DE REDES NEURONALES

PARA LA DETECCION DE ASTEROIDES

Máster universitario en Ciencia de los Datos / Data Science (MUDS)

Tutores:

- Xavier Vilasís, Vicens Gaitán

Estudiante:

- Josep Miquel Val Serra

Tabla de contenidos

RESUMEN

El proyecto de fin de Máster consiste en el análisis de una serie de quintetos de fotos del firmamento, tomadas desde varios de los telescopios de los observatorios de las Canarias, a diferentes intervalos de tiempo, con el fin de detectar asteroides. Para lograr dicha detección se creará un modelo basado en redes neuronales artificiales y más concretamente en redes neuronales convolucionales ya que son las que mejor están adaptadas para el análisis de fotos. Estas redes se utilizarán de forma supervisada a partir de los datos obtenidos de la visualización manual de dichos quintetos anotando el tipo de cuerpo encontrado y las coordenadas de su posición en un fichero anexo para cada una de las fotografías.

Antes del entreno de dichas redes, se procederá, como se suele hacer habitualmente en estos casos, al análisis y a la mejora de los datos de etiquetaje y de las fotografías, de manera a dejarlos preparados para que el entreno de la red pueda tener los mejores resultados posibles. En este caso la preparación será más profunda y se hará en dos etapas. En una primera etapa se mirará de mejorar el fichero de anotaciones de manera programática generando valores más coherentes y fotografías modificadas, es decir “limpias” de objetos no susceptibles de ser asteroides. En una segunda etapa se adaptarán los 2 grupos de fotografías (originales y modificadas) al formato de entrada de la red neuronal escogida y se procederá al entreno de la red.

Se considerará dentro de dichas redes convolucionales, las que se denomina estructuras “state-of-the-art” y que ofrecen para este tipo de aplicación los mejores resultados en precisión y en tiempo de ejecución. Se estudiará diferentes estructuras y se mirará de implementar varias de estas soluciones en Python con los “frameworks” Keras y Tensor Flow. El código se editará y ejecutará en Jupyter Notebook en el marco del servicio “Collaboratory” de la plataforma de Google.

1.INTRODUCCION

Desde hace mucho tiempo, los observatorios astronómicos y muchos particulares hacen un seguimiento especial de los asteroides y de los cometas. Este seguimiento es debido al hecho que dichos cuerpos podrían cruzarse en la trayectoria de la Tierra y su impacto con ella podría producir desde daños muy importantes en las zonas circundantes al impacto hasta la destrucción total de la vida en la Tierra. Existe una base de datos de asteroides y de cometas que se amplía cada año con nuevos avistamientos y que sirve para controlar su trayectoria y la probabilidad de impacto con la Tierra.

Los asteroides y los cometas se diferencian por su composición, su tamaño, su trayectoria y su velocidad.

- Un asteroide es un cuerpo rocoso, carbonáceo o metálico que puede alcanzar un tamaño de 1000km. Este cuerpo celeste es más pequeño que un planeta y mayor que un meteoroide (trozo de asteroide de tamaño inferior a 50m). Existe un gran cinturón de asteroides entre Marte y Júpiter, pero hay un número considerable que proviene de fuera del sistema solar y que en algunos casos tiene una trayectoria cercana a la de la Tierra. En el pasado, hay constancia de diferentes impactos de asteroides en la superficie terrestre pero sus consecuencias fueron limitadas debido a su tamaño reducido (al entrar en la atmosfera terrestre suelen vaporizarse parcial o totalmente) [1]

- Un cometa es un cuerpo formado por hielo, polvo y rocas y deja una estela detrás suyo al acercarse al sol al vaporizarse el hielo que lo compone. Su velocidad es muy superior a la de un asteroide y tiene diferentes tipos de trayectoria (hiperbólica, parabólica). Pero el número de cometas que entran en el sistema solar es muy reducido y aún lo es más el número de los que pasan relativamente cerca de la Tierra. [2]

De ahora en adelante, sólo nos ocuparemos de los asteroides ya que, aunque siendo muy reducida, la probabilidad de impacto es bastante más elevada que la de un cometa.

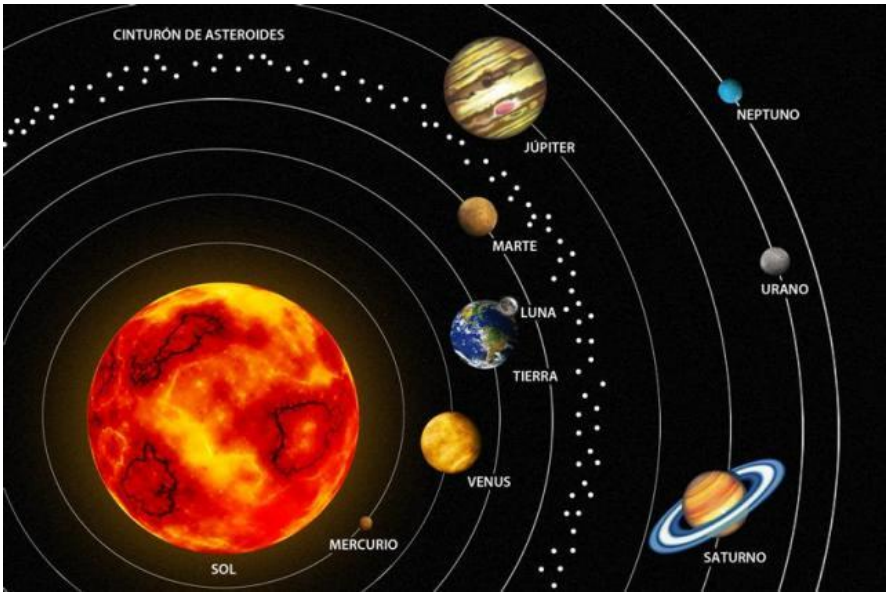


Figura 1- Cinturón de asteroides principal en el sistema solar

Siguiendo con el interés suscitado por los asteroides, la Nasa patrocinó un concurso [3] en el 2014 donde se pedía a los participantes, a partir de cuartetos de fotos del firmamento facilitadas por la propia Nasa, algoritmos de detección de dichos cuerpos. Se propusieron diferentes algoritmos de Machine Learning [4] (Random Forest [5], Naive Bayes [6]) y de procesamiento de imágenes donde se obtuvieron resultados, en varios casos, bastante buenos.

En lo que concierne este proyecto, se trata de un reto parecido al de la Nasa, donde se intentará explotar otros algoritmos de Machine Learning para predecir la posición de los asteroides. Dichos algoritmos, que han tenido un nuevo impulso desde hace unos 2 o 3 años, forman parte de lo que se denomina Deep Learning [7] y que está formado por algoritmos de redes neuronales de diferentes tipos. En este caso particular, se buscará utilizar estructuras de redes neuronales más eficientes en precisión y en tiempo de cálculo. Dichas estructuras se utilizarán de manera supervisada ya que las fotos estarán etiquetadas y el fichero correspondiente contendrá para cada una de las fotos las coordenadas de la posición del o de los asteroides encontrados.

2.DISEÑO

Antes de detallar las diferentes partes del diseño, vamos a resumir los pasos que se van a seguir.

En primer lugar, se va a analizar el conjunto de datos a partir de los cuales se va a trabajar y se completará dicho conjunto de manera adecuada, ya que, con los datos disponibles hasta ahora, no es posible la aplicación de los algoritmos supervisados previstos de Deep Learning [7]. Seguidamente se describirán los tipos de algoritmos que se va a utilizar y la manera como se implementará el mejor adaptado a esta finalidad. Finalmente se hará las pruebas sobre los conjuntos de datos y se presentará los resultados, los costes y las conclusiones.

2.1 Conjunto de datos

En este apartado separaremos la obtención de los datos y su análisis.

2.1.1 Obtención de los datos

Como se ha mencionado anteriormente, los observatorios astronómicos de Canarias proveen una serie de quintetos de fotos del firmamento nocturno que utilizaremos en este proyecto.

Los 2 principales observatorios de donde probablemente se han tomado las fotografías son el Observatorio del Teide en Tenerife y el de Roque de los Muchachos en la Palma. Estos 2 centros son plataformas de observación polivalentes abiertas a todas las comunidades científicas nacionales e internacionales. Cada una de ellas tiene a su disposición varios telescopios con diferente óptica y cámara adaptadas para una función específica y se dividen en 2 categorías principales, los instrumentos de observación nocturna y los de observación solar.

En el caso que nos concierne, es decir, telescopios nocturnos, suelen estar conectados a un sistema informático de control y de recogida de datos en el que se puede programar la orientación del telescopio y el intervalo de toma de fotografías de la cámara asociada.

Por ejemplo, en el gran Telescopio de Canarias en Roque de los Muchachos, figura 2, tenemos diferentes grupos de instrumentos conectados a dicho telescopio. Entre ellos hay un sistema llamado OSIRIS formado por una cámara CCD y un espectrógrafo de resolución baja e intermedia operando en el rango visible. Así mismo, hay otro sistema conectado llamado CANARICAM compuesto por una cámara y un espectrógrafo en el infrarrojo térmico. [9]

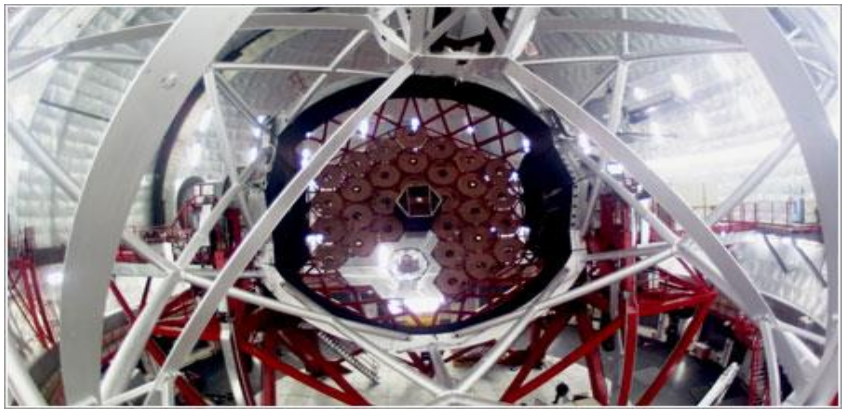


Figura 2 – Gran telescopio de La Palma

Las fotografías que tenemos que analizar son fotografías obtenidas de cámaras CCD que tienen un rango de trabajo dentro del espectro visible.

Las cámaras CCD funcionan convirtiendo fotones en electrones que luego se almacenan en cada píxel. Cada píxel puede contener un número máximo fijo de electrones, típicamente desde 35000 hasta 500000, dependiendo del modelo específico de CCD. Al integrar o exponer una imagen, los fotones golpean píxeles individuales y se convierten en electrones y se almacenan en cada píxel. La cantidad de electrones almacenados en cada píxel es proporcional a la cantidad de fotones que impactaron ese píxel. Después de que se ha completado una exposición, los electrones de cada píxel son desplazados fuera del CCD y se convierten en un número, que indica qué tan oscuro o claro debe ser cada píxel en particular, y se almacena en el archivo de una imagen. [9]

En un mundo ideal, cada fotón que choca con un píxel se convertiría exactamente en un electrón. Luego, el número de electrones se contabilizaría con precisión y se convertiría en un número que le indicaría al fotógrafo exactamente cuánta luz incidía en cada píxel. Pero, desafortunadamente, el proceso de conversión de valores de luz a píxel en una imagen de una cámara CCD se rige por algunas leyes físicas fundamentales y otros factores que introducen "ruido" en esa imagen. El ruido es una variación no deseada en los valores de píxeles que hacen que la imagen sea una representación distorsionada de la representación exacta de la escena original. [9]

El ruido en las imágenes CCD puede manifestarse de múltiples maneras, incluida la "granulosidad" en áreas de fondo más oscuras, líneas horizontales o verticales débiles que se vuelven visibles en áreas de baja señal de la imagen, gradientes manchados entre regiones más oscuras y más claras en una nebulosa, un gradiente de oscuro a la luz desde una esquina o lado de una imagen a la otra, y especialmente como imágenes de bajo contraste. Al final el resultado es una relación señal / ruido reducida. Si consideramos de media que 90000 electrones impactan un píxel y que 30 electrones de media corresponden al ruido, obtendríamos una relación señal sobre ruido(SNR) de 3000, aproximadamente 70db. Cuanto más alto sea el valor SNR más perfectas tendremos las imágenes. Y los fabricantes tienden a ofrecer modelos con un SNR cada vez más alto. [9]

Existe diferentes tipos de ruido como el “dark current”, la no uniformidad de los píxeles, “Shot noise”, “CCD Read Noise”,” CCD Camera Noise”,” Electronic interference”. [9]

El “Dark Current” es una corriente que circula en la cámara CCD estando o no expuesta a la luz y es causada por electrones que se generan térmicamente en los píxeles. [9]

La no uniformidad de los píxeles corresponde al hecho que los píxeles son ligeramente diferentes (dispersión entre 1% y 2%) en la generación de electrones. [9]

El “Shot Noise” es causado por la llegada aleatoria de fotones. Este es un rasgo fundamental de la luz. Como cada fotón es un evento independiente, la llegada de cualquier fotón no puede predecirse con precisión. [9]

El “CCD Read Noise” está formado por varias fuentes de ruido en los circuitos que forman la lectura de la cámara cuando está en funcionamiento. Los fabricantes de CCD suelen combinar todas las fuentes de ruido en el chip y expresar este ruido como una cantidad de electrones RMS. [9]

Los ruidos “CCD Camera Noise” y “Electronic interference” están relacionados con la aparición, en esos 2 casos diferentes, de microvoltajes que se traducen en ruido, respectivamente, cuando la cámara está apagada en parte de la circuitería analógica y cuando está encendida en la circuitería de conversión analógica->digital. [9]

Aunque pueda resultar preocupante ver todas estas fuentes de ruido y hacer creer que la imagen que vamos a obtener es muy imperfecta, la realidad es que, en la mayoría de las cámaras utilizadas actualmente, todos estos ruidos son minimizados por los fabricantes y al final podemos aproximar el ruido restante en las imágenes por modelos matemáticos conocidos.

2.1.2 Análisis de los datos

Estas fotografías tienen diferentes referencias en el nombre del fichero que parecen indicar que han sido tomadas desde diferentes instrumentos entre los cuales podría estar el telescopio citado anteriormente y han sido todas guardadas en un formato específico usado en diferentes campos entre los cuales está la astronomía.

Este formato es denominado **FITS** o **Flexible Image Transport System.** El más joven y quizás más usado tipo de datos en FITS es una imagen con una cabecera o bloque de datos. El término "imagen" es aplicado con cierta libertad, puesto que el formato soporta matrices de dimensiones arbitrarias -- una imagen con datos normales normalmente está en 2-D o 3-D (con la tercera dimensión se representa el plano de color). Los datos pueden encontrarse en otros formatos como los enteros o de punto flotante, siendo especificados en las cabeceras.

Las cabeceras de una imagen FITS pueden contener información acerca de uno o más sistemas de coordenadas que cubren la imagen por sí misma. Las imágenes contienen un sistema implícito de coordenadas cartesiano que describe la localización de cada pixel en la imagen, pero los usos científicos requieren que se trabaje con un sistema de coordenadas mundial. Como FITS se ha ido generalizando desde sus inicios, las especificaciones del sistema de coordenadas mundial han venido siendo más y más sofisticadas: desde muy temprano las imágenes en FITS permitieron un factor escalar simple para representar el tamaño de los píxeles; pero versiones recientes del estándar permiten múltiples sistemas de coordenadas no lineales, representando distorsiones arbitrarias de la imagen. [10]

FITS es a menudo utilizado para almacenar también datos que no son imágenes, como espectros electromagnéticos, listas de fotones, cubos de datos y muchos más. Un fichero FITS podría contener varias extensiones, y cada una de ellas podría contener datos de un objeto. Por ejemplo, es posible almacenar imágenes de rayos X y también imágenes pertenecientes al infrarrojo en el mismo archivo FITS. [10]

La mayor ventaja de FITS para datos científicos es que la información de las cabeceras es legible en ASCII, de modo que un usuario puede examinar las cabeceras para investigar un archivo de procedencia desconocida. Cada archivo FITS consiste en una o más cabeceras que contienen secuencias de 80 cadenas de caracteres fijos que llevan pares de valores, interpolados entre los bloques de datos. Los pares de valores proveen información (metadatos) como son el tamaño, origen, formato binario de los datos, comentarios, historia de los datos y cualquier otra información que el creador desee: mientras varias palabras están restringidas para FITS, el estándar permite el uso arbitrario de todas las palabras. [10]

En la figura siguiente se puede ver la estructura de un fichero FITS:

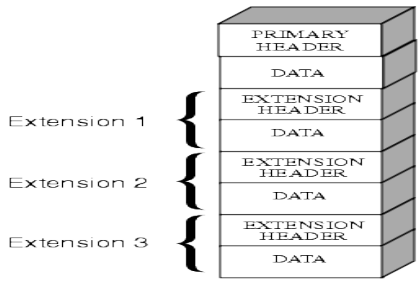
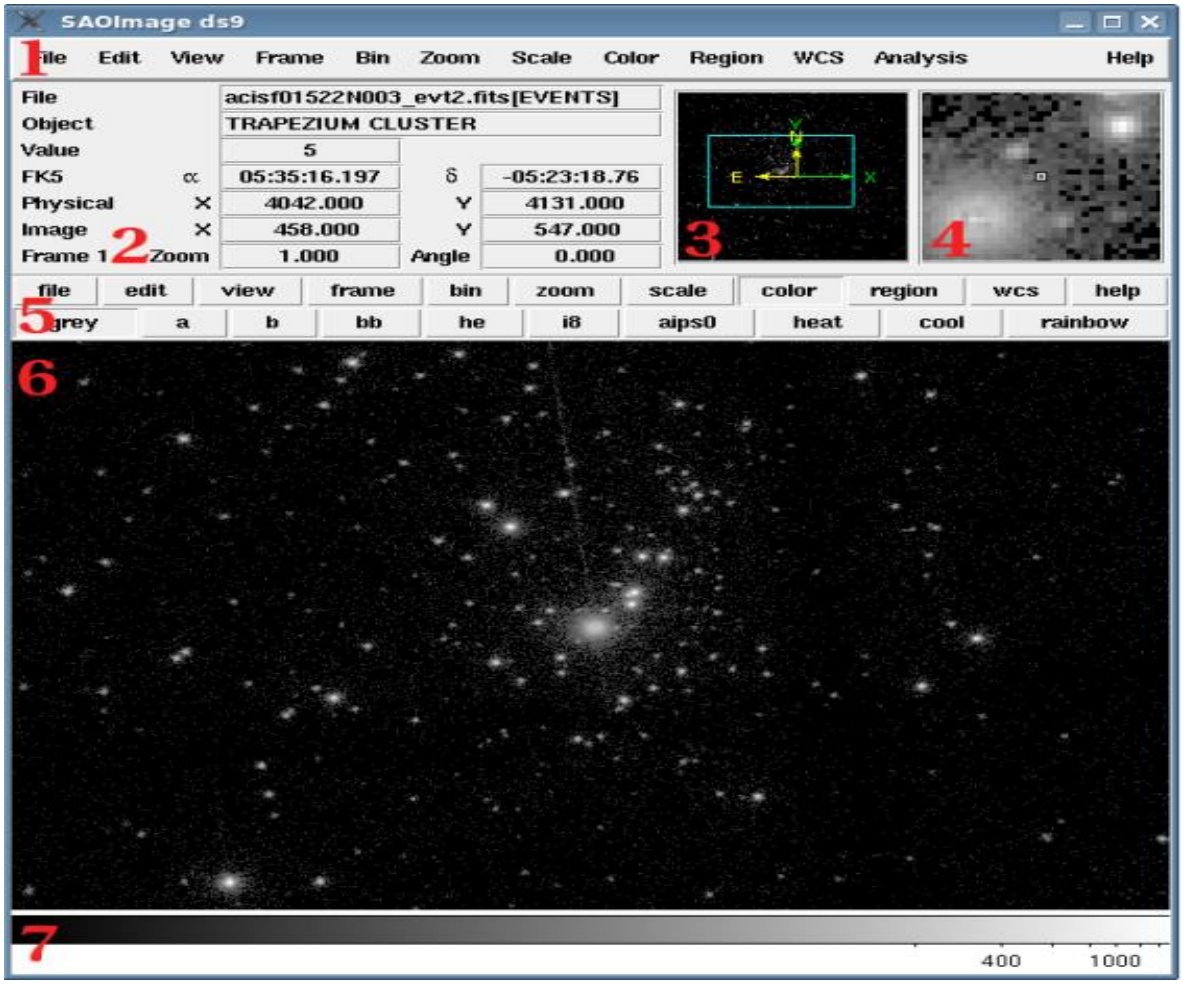
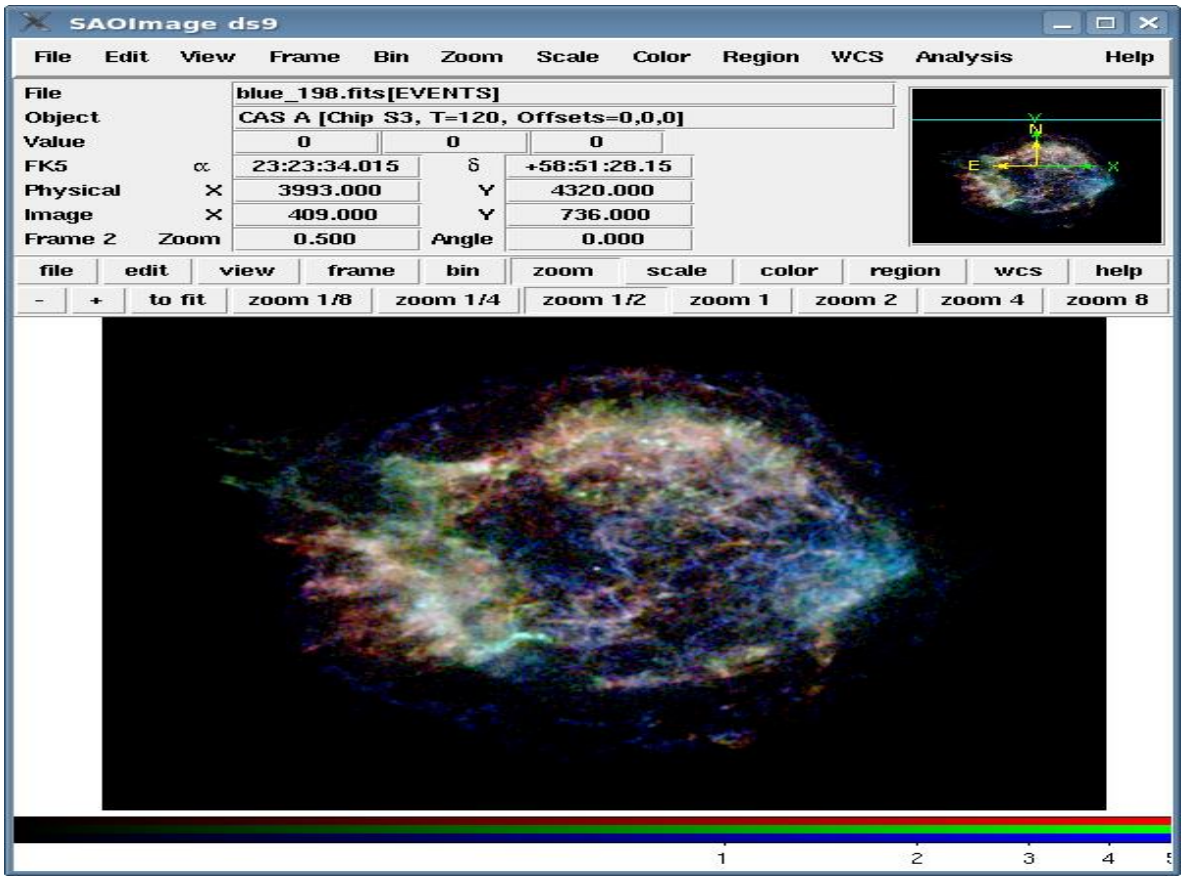


Figura 3 – Estructura general de un fichero FITS

Existen diferentes herramientas para poder ver las imágenes y la cabecera que contienen. Adjuntamos varios ejemplos de figuras (una en blanco y negro y otro en color) donde se ve fotos del firmamento en el visor "SAO Image DS9" (software de visualización):





Figuras 4 y 5 – Visualización de fotos astronómicas en formato FITS

Se adjunta también un ejemplo de cabecera que puede contener un fichero FITS (lista de datos en formato clave- valor):

```
SIMPLE  =                    T /
BITPIX  =                   16 /
NAXIS   =                    4 /
NAXIS1  =                 2048 /
NAXIS2  =                 1024 /
NAXIS3  =                    1 /
NAXIS4  =                    1 /
EXTEND  =                    T / TABLES FOLLOWING MAIN IMAGE
BLOCKED =                    T / FILE MAY BE BLOCKED
OBJECT  = '3C405    '          / SOURCE NAME
TELESCOP= '         '          /
INSTRUME= '         '          /
OBSERVER= 'PERL     '          /
DATE-OBS= '27/10/82'           /OBSERVATION START DATE DD/MM/YY
DATE-MAP= '14/07/83'           /DATE OF LAST PROCESSING DD/MM/YY
BSCALE  =    7.04625720812E-05 /REAL = FITS_VALUE * BSCALE + BZERO
BZERO   =    2.18688869476E+00 /
BUNIT   = 'JY/BEAM '           /UNITS OF FLUX
EPOCH   =      1.950000000E+03 /EPOCH OF RA DEC
DATAMAX =      4.495524406E+00 /MAX PIXEL VALUE
DATAMIN =     -1.217470840E-01 /MIN PIXEL VALUE
CTYPE1  = 'RA---SIN'           /
CRVAL1  =    2.99435165226E+02 /
CDELT1  =     -4.166666986E-05 /
CRPIX1  =      1.024000000E+03 /
CROTA1  =      0.000000000E+00 /
CTYPE2  = 'DEC--SIN'           /
CRVAL2  =    4.05961940065E+01 /
CDELT2  =      4.166666986E-05 /
CRPIX2  =      5.130000000E+02 /
CROTA2  =      0.000000000E+00 /
CTYPE3  = 'FREQ     '          /
CRVAL3  =    4.86635000000E+09 /
CDELT3  =      1.250000000E+07 /
CRPIX3  =      1.000000000E+00 /
```

Figura 6 –Ejemplo de los metadatos de un fichero FITS

En muchos ficheros FITS de proyectos de la Nasa acostumbra a haber más de una cabecera. En realidad, existe una cabecera primaria donde se indica los datos genéricos asociados a cada una de las imágenes (formato de las matrices asociadas a la imagen, zona del firmamento donde se ha hecho la foto, fecha de la foto) y unas cabeceras secundarias donde se indican datos de objetos específicos de la imagen (tipo de objeto, posición en el firmamento).

En nuestro proyecto, los metadatos contenidos en la cabecera no son especialmente útiles a nivel de conocimientos complementarios sobre los asteroides ya que a parte de la fecha de la foto no tenemos ningún otro dato relevante del asteroide. Simplemente podemos verificar que los quintetos de fotos se han tomado en la misma parte del firmamento en diferentes momentos del tiempo próximos entre ellos.

Las fotos de los observatorios de las Canarias vienen en una escala de grises y en un tamaño variable de matrices (613x613,683x683,683x700, …, 1012x1012, …, 1024x1024, 2056x2048, …,2593x2760). Además, el formato de los datos en las matrices (intensidad de la escala de grises) es también variable. En algunas fotos tenemos enteros de 16bits (uint16) y en otras, números flotantes de 32bits (float32).

Las fotos vienen, como ya se ha dicho anteriormente, en formato de tensores **lin** x **col** x **can** donde **lin** es el número de líneas, **col** el número de columnas y **can** el número de canales. En este caso particular, el número **can** será igual a 1 ya que se trata de fotos en la escala de grises, por lo que nuestro tensor se reducirá a una matriz **lin** x **col.**

En las figuras siguientes, se adjunta una parte de la matriz correspondiente a la misma zona concreta en varias fotos de un mismo quinteto que no contienen ningún cuerpo celeste:

```
array([[1396, 1400, 1390, 1387, 1388],
       [1380, 1397, 1379, 1397, 1385],
       [1391, 1397, 1386, 1391, 1376],
       [1395, 1403, 1379, 1385, 1387],
       [1394, 1396, 1374, 1394, 1390]], dtype=uint16)
```

Figura 7 – Intensidad de luz en la foto nº0 de un quinteto

```
array([[1390, 1386, 1379, 1378, 1378],
       [1380, 1378, 1380, 1386, 1384],
       [1373, 1379, 1382, 1378, 1374],
       [1380, 1383, 1387, 1385, 1384],
       [1387, 1388, 1390, 1382, 1380]], dtype=uint16)
```

Figura 8 – Intensidad de luz en la foto nº1 de un quinteto

```
array([[1394, 1391, 1389, 1387, 1392],
       [1392, 1394, 1395, 1395, 1401],
       [1397, 1398, 1391, 1384, 1391],
       [1404, 1391, 1390, 1395, 1395],
       [1402, 1398, 1397, 1394, 1389]], dtype=uint16)
```

Figura 9 – Intensidad de luz en la foto nº2 de un quinteto

```
array([[1461, 1445, 1450, 1456, 1461],
       [1457, 1446, 1451, 1446, 1454],
       [1459, 1449, 1454, 1444, 1449],
       [1456, 1450, 1461, 1463, 1455],
       [1453, 1447, 1455, 1455, 1448]], dtype=uint16)
```

Figura 10 – Intensidad de luz en la foto nº3 de un quinteto

```
array([[1460, 1458, 1449, 1454, 1458],
       [1459, 1454, 1453, 1449, 1449],
       [1458, 1455, 1451, 1449, 1445],
       [1450, 1456, 1453, 1449, 1456],
       [1447, 1453, 1463, 1448, 1452]], dtype=uint16)
```

Figura 11 – Intensidad de luz en la foto nº4 de un quinteto

Como se puede ver, las fotos contienen un ruido de fondo de la misma posición del cielo que es variable según las fotos. Cada una de ellas tiene una mediana y una desviación estándar diferente. (En este caso el valor de la mediana tiene más sentido que el de la media debido a que los "outliers", correspondientes a estrellas muy brillantes que tienen valores de intensidad superiores a 60000 y a los defectos del detector con valores cercanos o inferiores a 0, modifican sustancialmente su valor)

En el caso particular de las figuras anteriores, obtenemos los valores siguientes:

Mediana->1390 Desviación estándar->14.12 Foto nº0

Mediana->1369 Desviación estándar->12.04 Foto nº1

Mediana->1379 Desviación estándar->13.76 Foto nº2

Mediana->1440 Desviación estándar->14.61 Foto nº3

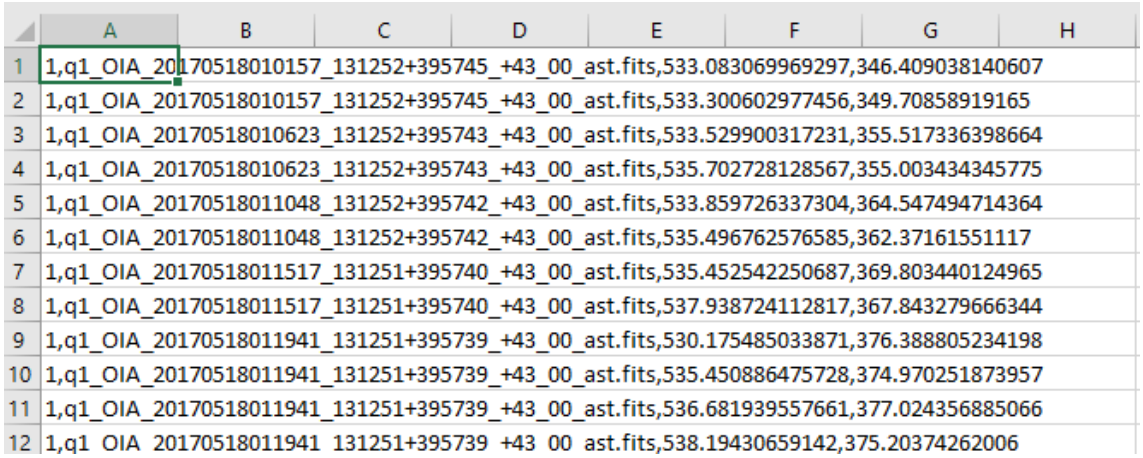
Mediana->1441 Desviación estándar->13.22 Foto nº4

Así mismo, existe un fichero de datos en formato **csv** proveído también por los observatorios de las Canarias en el que diferentes operarios utilizando los visores citados anteriormente han buscado los posibles asteroides vistos en un quinteto de fotos y han indicado la posición de su centro de gravedad o de sus centros de gravedad, en el caso de que haya más de 1 asteroide.

Sin tener ninguna idea a priori de donde estarían los asteroides, es de suponer que el operario divide la imagen en cuadrículas relativamente pequeñas ya que se supone que el desplazamiento de una asteroide es inferior a los 18 píxeles entre una foto y la foto sucesiva, y teniendo abierto cinco visores diferentes conteniendo cada uno de ellos una imagen diferente del quinteto, intenta ver un desplazamiento entre las diferentes imágenes. Una vez examinada la cuadrícula pasa a cuadrícula adyacente hasta recorrer toda la imagen. Esta manera de proceder puede ser muy laboriosa y más aún en el caso de asteroides pequeños que se desplazan menos de 2 o 3 píxeles. Existe la posibilidad también, que, gracias a ciertos datos adicionales obtenidos de astrónomos o físicos, los operarios puedan acotar la hipotética posición del asteroide a ciertas cuadrículas de la imagen.

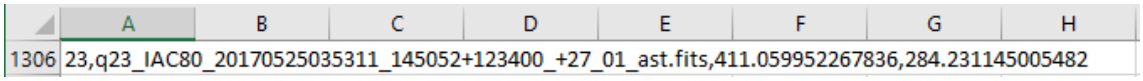
Pero dicho fichero **csv** plantea un gran problema para la utilización de algoritmos supervisados como los que queremos utilizar, ya que los operarios que anotaron dichos centros de gravedad no parecen haberlos anotado de manera correcta.

En las dos figuras siguientes se puede ver los diferentes problemas del fichero **csv**:



	A	B	C	D	E	F	G	H
1	1,q1_OIA_20170518010157_131252+395745_+43_00_ast.fits,533.083069969297,346.409038140607							
2	1,q1_OIA_20170518010157_131252+395745_+43_00_ast.fits,533.300602977456,349.70858919165							
3	1,q1_OIA_20170518010623_131252+395743_+43_00_ast.fits,533.529900317231,355.517336398664							
4	1,q1_OIA_20170518010623_131252+395743_+43_00_ast.fits,535.702728128567,355.003434345775							
5	1,q1_OIA_20170518011048_131252+395742_+43_00_ast.fits,533.859726337304,364.547494714364							
6	1,q1_OIA_20170518011048_131252+395742_+43_00_ast.fits,535.496762576585,362.37161551117							
7	1,q1_OIA_20170518011517_131251+395740_+43_00_ast.fits,535.452542250687,369.803440124965							
8	1,q1_OIA_20170518011517_131251+395740_+43_00_ast.fits,537.938724112817,367.843279666344							
9	1,q1_OIA_20170518011941_131251+395739_+43_00_ast.fits,530.175485033871,376.388805234198							
10	1,q1_OIA_20170518011941_131251+395739_+43_00_ast.fits,535.450886475728,374.970251873957							
11	1,q1_OIA_20170518011941_131251+395739_+43_00_ast.fits,536.681939557661,377.024356885066							
12	1,q1_OIA_20170518011941_131251+395739_+43_00_ast.fits,538.19430659142,375.20374262006							

Figura 12 – Centro de gravedad del asteroide del quinteto nº1



	A	B	C	D	E	F	G	H
1306	23,q23_IAC80_20170525035311_145052+123400_+27_01_ast.fits,411.059952267836,284.231145005482							

Figura 13 – Centro de gravedad del asteroide del quinteto nº23

En el caso de la figura 12, tenemos 12 anotaciones para 5 fotos.

La foto q1_OIA_2017051801**0157**_131252+395745_+43_00_ast. fits tiene 2 anotaciones.

La foto q1_OIA_2017051801**0623**_131252+395743_+43_00_ast. fits tiene 2 anotaciones.

La foto q1_OIA_2017051801**1048**_131252+395742_+43_00_ast. fits tiene 2 anotaciones.

La foto q1_OIA_2017051801**1517**_131252+395740_+43_00_ast. fits tiene 2 anotaciones.

La foto q1_OIA_2017051801**1941**_131252+395739_+43_00_ast. fits tiene 4 anotaciones.

Tenemos más de una anotación por foto y además hay una diferencia que puede alcanzar los 8 píxeles en las anotaciones de una misma foto. En el caso de unas cuantas fotos (no en todas), en el nombre del fichero **fits** que contiene el tensor con la fotografía, se indica el año (resaltado en rojo), el mes (resaltado en verde), el día (resaltado en azul), la hora-minuto-segundo (resaltado en negrita) en que fue tomada y finalmente (resaltado en amarillo) el grupo de trabajo del observatorio astronómico. Las coordenadas, por ejemplo, de la primera anotación de la primera foto es de 533.08 píxeles en el eje x y de 346.41 píxeles en el eje y.

En el caso de la figura 13, tenemos sólo una anotación y corresponde a la primera foto del quinteto (q1_IAC80_2017052**5035311**_145052+123400_+27_01_ast. fits). Este dato está incompleto, ya que para que haya un movimiento se necesita la detección de los objetos en 2 fotos diferentes y en el caso del movimiento uniforme necesitamos como mínimo 3 fotografías. Aunque, por lo que se verá más adelante podría haber casos en que el asteroide estuviera tocando visualmente una estrella y no se detectaría propiamente dicho por lo que podría haber quintetos sólo 2 fotos con asteroide, aun siendo el movimiento uniforme.

Estos 2 son unos simples ejemplos de la imprecisión que hay en el etiquetaje de las fotos y que implican 2 posibles caminos por tomar:

1. Mirar una por una las 5 fotografías de cada uno de los 192 quintetos y ver para cada una de ellas la posición en los tensores correspondientes de cada asteroide detectado y corregir manualmente su posición en el fichero **csv**.
2. Hacer un programa que lea los ficheros **fits** y detecte, a partir de diferentes funciones aplicadas a las 5 fotografías que componen el quinteto, los posibles asteroides que hubiera en cada una de ellas.

Se ha elegido la segunda opción ya que aporta globalmente más ventajas que desventajas.

Ventajas:

- El programa puede detectar uno o varios asteroides en un tiempo máximo que puede variar entre 5s y 90s dependiendo del tamaño del tensor y de la zona del espacio estudiada.
- El programa puede marcar la superficie ocupada y esto puede ser de ayuda posteriormente en el uso de los algoritmos de Deep Learning [7] supervisados que se podrían utilizar, ya que puede marcar automáticamente lo que se suele llamar "ground truth" [11]
- El programa permite ajustes para la búsqueda de asteroides, según los parámetros que consideremos para decidir si el movimiento de un objeto puede ser considerado como aceptable para un asteroide.

Desventajas:

- Aunque el tiempo para verificar cada uno de los asteroides y corregirlos manualmente puede ser largo y tedioso, siempre será más corto que hacer un programa de etiquetaje. Y visto los diferentes casos que se pueden dar, aun hacen más complicado el programa y con ello el tiempo dedicado a esa tarea.

2.2 Programa de etiquetaje de asteroides

En la elaboración del programa se ha pasado por diferentes etapas.

- En un primer momento al tener sólo 2 quintetos nº22 y nº23, con unos tamaños relativamente pequeños y unos movimientos suficientemente amplios, los asteroides no se solapaban y simplemente buscando las medianas y las desviaciones estándar, para eliminar el ruido de fondo y posteriormente ejecutando una función XOR y eliminando los restos de estrellas que quedaban por imprecisión de las fotografías, se obtenía los asteroides existentes.
- Posteriormente, al tener los otros quintetos hasta los 192 finales, se pudo constatar que el desplazamiento de los asteroides podía ser muy pequeño (inferior a 1 píxel entre foto a foto), que el tamaño de los astroides podía ser bastante grande y que el nivel de intensidad de los píxeles de un asteroide era, para unos cuantos casos, apenas superior al umbral del ruido, por lo que la utilización de la función XOR no permitía obtener los asteroides existentes. Por lo que se tuvo que buscar una solución bastante más elaborada que se detallará seguidamente.

2.2.1 Casos posibles de asteroides

El primer paso en la busca de los asteroides es la eliminación del ruido de fondo de cada una de las fotos que contiene el quinteto de fotos hechas de la misma zona del espacio y obtenidas a intervalos de tiempo determinados.

Cuando se elimina el ruido, dos objetos dependiendo de lo suficientemente cerca que estén, se confunden en un solo objeto ya que entre los dos existe toda una serie de píxeles que están por encima del umbral de ruido y no permiten al programa poderlos separar.

El caso que se viene de citar es uno de los casos posibles, pero ahora citaremos otros casos que pueden aparecer:

1- Un asteroide aparece en cada una de las 5 fotos de manera nítida y está separado de los objetos fijos (estrellas). Este sería el caso más sencillo.
2- Un asteroide aparece sólo en alguna de las 5 fotos (tiene que aparecer como mínimo en 2 ellas) e intercepta total o parcialmente otros objetos fijos(estrellas) en el resto de ellas
3- Un asteroide aparece en dos de las fotos cerca de los bordes de la matriz de manera nítida y las siguientes posiciones estarían fuera del marco de la fotografía
4- El nivel de intensidad del píxel con el valor más alto de todo el "blob" está justo por encima del umbral del ruido y genera por lo tanto "blobs" muy pequeños inferiores a 3 píxeles.
5- Un asteroide aparece sólo en una de las fotos. En el resto de las fotografías podría interceptar otros objetos fijos (estrellas)
6- Un asteroide sólo aparece en algunas de las fotos y por problemas de ruido no aparece en las demás

El programa puede detectar asteroides en los casos nº1, nº2, nº3, nº4 y sólo es capaz de hacer alguna detección en el caso nº5 y no lo detecta en el caso nº6.

En las figuras siguientes se muestra un ejemplo de los casos 1 y 2 correspondientes a los quintetos nº1 y nº23 de los cuales ya se ha hablado en el apartado anterior.

Quinteto nº1:

```
1  myphotos[0][344:365,530:540]

array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0, 1192, 1174, 1213, 1213, 1197,    0,    0,    0,    0],
       [   0, 1188, 1229, 1380, 1462, 1347,    0,    0,    0,    0],
       [   0, 1183, 1251, 1533, 1685, 1468, 1240,    0,    0,    0],
       [   0, 1192, 1323, 1623, 1715, 1419, 1216,    0,    0,    0],
       [   0, 1248, 1391, 1632, 1680, 1417, 1210,    0,    0,    0],
       [   0, 1226, 1314, 1515, 1610, 1404, 1188,    0,    0,    0],
       [   0,    0, 1214, 1346, 1490, 1385,    0,    0,    0,    0],
       [   0,    0,    0, 1174, 1260, 1237,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 14 - Asteroide del quinteto nº1 en la foto 0 (Centro de Gravedad: x=533.933, y=347.998)

```
1 myphotos[1][344:365,530:540]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0, 1208,    0,    0,    0,    0],
       [   0,    0, 1206, 1271, 1360, 1358, 1233,    0,    0,    0],
       [   0,    0, 1235, 1355, 1522, 1508, 1335, 1216,    0,    0],
       [   0, 1221, 1290, 1454, 1649, 1617, 1369, 1213,    0,    0],
       [   0, 1235, 1349, 1553, 1712, 1597, 1356, 1212,    0,    0],
       [   0, 1247, 1420, 1636, 1681, 1456, 1260,    0,    0,    0],
       [   0, 1235, 1402, 1605, 1562, 1322, 1205,    0,    0,    0],
       [   0, 1205, 1283, 1374, 1346, 1225,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 15 - Asteroide del quinteto nº1 en la foto 1 (Centro de Gravedad: x=534.014, y=355.01)

```
1 myphotos[2][358:378,530:540]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 1304, 1391, 1406, 1322,    0,    0,    0],
       [   0,    0, 1283, 1423, 1615, 1608, 1397, 1257,    0,    0],
       [   0,    0, 1337, 1554, 1731, 1632, 1403, 1262,    0,    0],
       [   0, 1258, 1363, 1591, 1756, 1644, 1420, 1275,    0,    0],
       [   0,    0, 1330, 1531, 1722, 1653, 1424, 1288,    0,    0],
       [   0,    0, 1282, 1422, 1588, 1578, 1397, 1260,    0,    0],
       [   0,    0,    0, 1330, 1430, 1427, 1328,    0,    0,    0],
       [   0,    0,    0,    0, 1283, 1272,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 16 - Asteroide del quinteto nº1 en la foto 2 (Centro de Gravedad: x=534.016, y=361.997)

```
1 myphotos[3][358:378,530:540]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0, 1310, 1382, 1398, 1356,    0,    0],
       [   0,    0,    0, 1293, 1417, 1548, 1560, 1445, 1330,    0],
       [   0,    0,    0, 1352, 1526, 1708, 1644, 1421, 1315,    0],
       [   0,    0,    0, 1376, 1610, 1844, 1691, 1398, 1294,    0],
       [   0,    0,    0, 1365, 1650, 1933, 1708, 1392, 1292,    0],
       [   0,    0,    0, 1338, 1577, 1829, 1618, 1332, 1298,    0],
       [   0,    0,    0, 1297, 1405, 1568, 1472, 1295,    0,    0],
       [   0,    0,    0,    0,    0, 1357, 1349,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 1294,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 17 - Asteroide del quinteto nº1 en la foto 3 (Centro de Gravedad: x=535.017, y=369.996)

```
1 myphotos[4][365:385,530:540]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0, 1359, 1348,    0,    0,    0],
       [   0,    0,    0, 1347, 1427, 1499, 1464, 1353,    0,    0],
       [   0,    0,    0, 1406, 1558, 1698, 1616, 1403,    0,    0],
       [   0,    0, 1331, 1437, 1635, 1863, 1779, 1472, 1337,    0],
       [   0,    0,    0, 1393, 1601, 1936, 1888, 1526, 1343,    0],
       [   0,    0,    0, 1340, 1519, 1866, 1884, 1565, 1347,    0],
       [   0,    0,    0,    0, 1432, 1668, 1708, 1518, 1349,    0],
       [   0,    0,    0,    0, 1353, 1468, 1473, 1381,    0,    0],
       [   0,    0,    0,    0,    0, 1343, 1329,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 18 - Asteroide del quinteto nº1 en la foto 4 (Centro de Gravedad: x=535.082, y=376.001)

Quinteto nº23:

```
1 myphotos[0][277:297,409:419]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [1363, 1403, 1421, 1370,    0,    0,    0,    0,    0,    0],
       [1376, 1453, 1485, 1415,    0,    0,    0,    0,    0,    0],
       [   0, 1365, 1387,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 19 – Asteroide del quinteto nº23 en la foto 0(Centro de Gravedad x=282.988, y=410.987)

```
1 myphotos[1][277:297,409:419]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 1354,    0,    0,    0],
       [   0,    0,    0,    0,    0, 1353, 1389, 1412, 1385,    0],
       [   0,    0,    0,    0,    0,    0, 1392, 1430, 1421, 1373],
       [   0,    0,    0,    0,    0,    0,    0, 1362, 1375, 1352],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 20 – Asteroide del quinteto nº23 en la foto 1(Centro de Gravedad x=284.982, y=416.01)

```
1 myphotos[2][277:297,419:429]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0, 1381, 1386, 1378, 1378],
       [   0,    0,    0,    0,    0, 1372, 1394, 1410, 1412, 1414],
       [   0,    0,    0,    0, 1374, 1396, 1416, 1441, 1452, 1460],
       [   0,    0,    0,    0, 1376, 1401, 1436, 1477, 1526, 1552],
       [   0,    0,    0,    0, 1380, 1401, 1450, 1537, 1658, 1763],
       [   0,    0,    0,    0, 1384, 1405, 1457, 1563, 1771, 2018],
       [   0, 1387, 1401, 1393, 1393, 1409, 1456, 1542, 1738, 2034],
       [1372, 1426, 1461, 1447, 1414, 1398, 1418, 1475, 1589, 1777],
       [   0, 1401, 1447, 1441, 1402, 1377, 1387, 1418, 1471, 1551],
       [   0,    0, 1380, 1387, 1375,    0, 1382, 1386, 1414, 1453],
       [   0,    0,    0,    0,    0,    0,    0, 1372, 1388, 1398],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 21 – Asteroide del quinteto nº23 en la foto 2

```
1 myphotos[3][277:297,425:435]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1375, 1371, 1366,    0,    0,    0,    0,    0],
       [1375, 1384, 1397, 1403, 1394, 1381,    0,    0,    0,    0],
       [1401, 1418, 1438, 1450, 1435, 1414, 1385, 1366,    0,    0],
       [1425, 1470, 1518, 1543, 1528, 1480, 1432, 1399, 1375,    0],
       [1447, 1529, 1648, 1751, 1755, 1644, 1515, 1429, 1386,    0],
       [1450, 1553, 1757, 1999, 2076, 1900, 1645, 1482, 1407, 1375],
       [1432, 1525, 1727, 2021, 2183, 2032, 1728, 1517, 1428, 1392],
       [1419, 1488, 1603, 1789, 1932, 1868, 1669, 1510, 1436, 1398],
       [1419, 1480, 1537, 1595, 1636, 1618, 1545, 1475, 1428, 1395],
       [1422, 1482, 1518, 1515, 1498, 1480, 1465, 1438, 1406, 1390],
       [1392, 1433, 1452, 1441, 1431, 1423, 1420, 1411, 1390, 1378],
       [   0, 1369, 1381, 1382, 1389, 1390, 1386, 1385, 1374,    0],
       [   0,    0,    0,    0,    0, 1370,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1366,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1393, 1404, 1381],
       [   0,    0,    0,    0,    0,    0, 1369, 1413, 1435, 1407],
       [   0,    0,    0,    0,    0,    0,    0, 1390, 1409, 1392],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 22 – Asteroide del quinteto nº23 en la foto 3

```
1 myphotos[4][277:297,425:435]
```

```
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0, 1421, 1425, 1430, 1428, 1417,    0,    0,    0,    0],
       [1427, 1461, 1475, 1482, 1475, 1453, 1425, 1414,    0,    0],
       [1449, 1508, 1562, 1584, 1566, 1524, 1469, 1434,    0,    0],
       [1474, 1558, 1696, 1811, 1808, 1692, 1553, 1472, 1426,    0],
       [1487, 1593, 1810, 2069, 2147, 1959, 1688, 1514, 1446, 1411],
       [1469, 1569, 1771, 2067, 2219, 2063, 1763, 1548, 1469, 1426],
       [1444, 1510, 1627, 1810, 1938, 1877, 1701, 1549, 1473, 1431],
       [1423, 1463, 1511, 1576, 1647, 1658, 1602, 1521, 1463, 1430],
       [   0, 1428, 1447, 1474, 1512, 1547, 1550, 1512, 1469, 1432],
       [   0,    0, 1419, 1435, 1452, 1504, 1546, 1534, 1487, 1440],
       [   0,    0,    0, 1415, 1420, 1470, 1516, 1517, 1489, 1440],
       [   0,    0,    0,    0,    0, 1426, 1444, 1457, 1451, 1423],
       [   0,    0,    0,    0,    0,    0,    0, 1417, 1422,    0],
       [   0,    0,    0,    0,    0,    0, 1415, 1438, 1442, 1418],
       [   0,    0,    0,    0,    0,    0, 1423, 1462, 1475, 1448],
       [   0,    0,    0,    0,    0,    0,    0, 1430, 1444, 1426],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 23 – Asteroide del quinteto nº23 en la foto 4

Como se puede ver en el quinteto nº1, figuras 14,15,16,17 y 18 el asteroide no está cerca de ningún otro cuerpo celeste y aparece visible en todas las fotos del quinteto.

Por el contrario, en el quinteto nº23, figuras 19,20,21,22 y 23 el asteroide sólo es visible en las fotos 0 y 1 (figuras 19 y 20). En la foto 2 (figura 21), el asteroide (marcado por una línea roja se solapa con una estrella marcada con una línea azul). Posteriormente, en las fotos 3 y 4 (figuras 22 y 23) el asteroide queda enmascarado por la misma estrella marcada por una línea azul. Hay que resaltar que existe otra estrella más pequeña marcada en línea azul que en la foto 3 sale separada de la estrella principal y en la foto 4 sale solapada con esta última. No sólo existe solapamientos entre asteroides y estrellas, pero también entre estrellas.

2.2.2 Estructura del programa

El programa se compone de varios bloques:

1.Lectura de los ficheros ".fits"

El programa lee todos los ficheros ". fits" que encuentra en el directorio del programa y sus subdirectorios y, gracias a la función "fits.getdata" de la librería "astropy" [12], separa el tensor de los metadatos asociados. De los metadatos extrae la hora a la que fue tomada la fotografía para poder calcular más adelante el intervalo de tiempo que pasa entre las diferentes fotografías. Una vez leídos todos los ficheros ". fits" los va asociando por grupos de 5 para poder tener juntas las fotografías de un mismo quinteto. Además, corrige la incompatibilidad de dimensiones de los diferentes tensores (imágenes) de un quinteto. A partir del quinteto 100 se producen numerosas anomalías que se traducen en dimensiones de fotografías diferentes. Por ejemplo, en un mismo quinteto encontramos 4 fotografías de 1013x1013 y 1 fotografía de 1012x1012. El programa recorta las matrices a la dimensión más pequeña.

2.Cálculo del umbral de ruido y supresión del ruido

Para poder calcular el umbral de ruido de cada una de las fotografías, suponemos que el ruido se puede aproximar por una distribución "Gaussiana", de la cual podemos mirar de calcular la media y la desviación estándar. Como en el caso de una distribución "Gaussiana", la media y la mediana tienen el mismo valor, utilizaremos la mediana, ya que, como se ha señalado anteriormente, los defectos del detector de la cámara haciendo aparecer algunos puntos negros (píxeles con intensidades muy bajas y muy alejadas de la media) y algunas estrellas muy brillantes (píxeles con intensidades muy elevadas y muy alejadas de la media) distorsionan su valor. Para poder calcular la mediana y la desviación estándar existe una función llamada "sigma-clipped-stats " [13] de la librería "astropy" en la que se calculará el valor de la mediana y del valor sigma de la desviación estándar en una primera iteración y que a partir de la segunda iteración ira desechando los valores que estén por encima de un valor n*sigma (siendo n un valor fijado por el usuario al inicio, usualmente entre 2 y 3), volviendo a calcular la mediana y la sigma y que parará el bucle cuando, de una iteración a la siguiente, las variaciones de la mediana y de la sigma sean inferiores a un valor determinado suficientemente pequeño. Una vez obtenidas la mediana y la desviación estándar se hará la aproximación que todos los píxeles con intensidad inferiores a la mediana+3*sigma pertenecen al ruido de fondo y serán anulados (este valor significará que discriminaremos el 97.5% del ruido y que la mayoría de los valores encontrados serán realmente cuerpos celestes).

En las figuras siguientes, vemos, por ejemplo, el resultado de la eliminación del ruido para la fotografía nº0 del quinteto nº22 en el intervalo 285-295 para las líneas y 420-430 para las columnas.

```
[34]  1 photosdatagroup[0][0][285:295,420:430]
```

```
array([[1447, 1432, 1433, 1421, 1417, 1407, 1431, 1411, 1393, 1391],
       [1427, 1424, 1437, 1411, 1420, 1430, 1477, 1464, 1402, 1379],
       [1402, 1420, 1398, 1420, 1409, 1456, 1519, 1519, 1434, 1394],
       [1405, 1414, 1402, 1415, 1407, 1437, 1474, 1474, 1432, 1396],
       [1410, 1445, 1459, 1434, 1425, 1419, 1428, 1410, 1404, 1404],
       [1406, 1453, 1491, 1462, 1405, 1403, 1399, 1401, 1382, 1386],
       [1393, 1403, 1425, 1428, 1403, 1419, 1395, 1405, 1385, 1395],
       [1389, 1389, 1386, 1394, 1406, 1387, 1385, 1386, 1390, 1397],
       [1379, 1382, 1385, 1392, 1398, 1382, 1388, 1403, 1382, 1393],
       [1391, 1396, 1390, 1391, 1384, 1390, 1385, 1406, 1394, 1399]],
      dtype=uint16)
```

Figura 24 – Fotografía nº0 antes de la eliminación del ruido

Estos son los valores estadísticos de la fotografía estudiada:

```
mediana  1379.0
sigma  14.121132661778443
umbral 1421.3633979853353
```

```
[32]  1 mysources[0][285:295,420:430]
```

```
array([[1447, 1432, 1433,    0,    0,    0, 1431,    0,    0,    0],
       [1427, 1424, 1437,    0,    0, 1430, 1477, 1464,    0,    0],
       [   0,    0,    0,    0,    0, 1456, 1519, 1519, 1434,    0],
       [   0,    0,    0,    0,    0, 1437, 1474, 1474, 1432,    0],
       [   0, 1445, 1459, 1434, 1425,    0, 1428,    0,    0,    0],
       [   0, 1453, 1491, 1462,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1425, 1428,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 25 – Fotografía nº0 después de haber puesto a 0 todos los píxeles con intensidad<umbral

Se ha estudiado también la posibilidad de utilizar un filtro llamado “mediano” [14] que calcula la mediana de un cuadrado n x n (n impar) cuyo centro es el valor que queremos filtrar. En el trabajo de la Nasa alguna de las soluciones lo utilizan y permite de eliminar pequeños grupos de píxeles que están cerca del umbral. Así mismo permite de nivelar la intensidad de los píxeles que forman un objeto. Pero en nuestro caso no se ha considerado su utilización, ya que existen quintetos cuya solución está formada por asteroides de menos de 5 píxeles con una intensidad cercana al umbral del ruido y su utilización los eliminaría. Así mismo, su utilización provoca que diferentes objetos cercanos tengan más posibilidad de solaparse y que más difícil sea su discriminación.

3.Eliminación de grandes estrellas

Existen casos en alguno de los quintetos en que hay cuerpos celestes con una superficie aproximada de más de 1000 píxeles. Estos cuerpos que acostumbran a ser estrellas podrían dar lugar a una posible mal interpretación de uno de los casos citados al principio del análisis del programa. Nos podemos encontrar varios pares de objetos que parecen tener un movimiento en 2 de las fotos y que interceptan en las otras 3 fotos este cuerpo

de más de 1000 píxeles. En varios de los quintetos, se ha dado este caso obteniéndose de 10 a 20 soluciones posibles pero dudosas. Por este motivo, se han eliminado todas estas "grandes estrellas". El valor 1000 se ha escogido de manera orientativa, pero se ha confirmado en estos 192 quintetos que era una aproximación válida ya que, todos estos asteroides que podrían aparecer como soluciones, han sido eliminados.

4.Eliminación de imperfecciones y separación de cuerpos levemente unidos

Después de la supresión de los píxeles con un valor de intensidad por debajo del umbral de ruido, se generan como ya se ha dicho anteriormente toda una serie de "blobs" (grupos de píxeles de intensidad no nula) que podemos caracterizar completamente gracias a las funciones de la librería "openCV" [15]. La función "findContours" [16] puede encontrar las coordenadas de todos los píxeles que forman el contorno de cada uno de los "blobs". Con la función "pointPolygonTest" [17] podemos determinar las coordenadas de los píxeles que están dentro de cada uno de los contornos encontrados anteriormente. Así mismo podemos encontrar las coordenadas del píxel con el valor de intensidad máximo que tiene cada uno de los "blobs". Desgraciadamente, por la manera que tienen las funciones de "openCV" [15] de caracterizar cada "blob", existen ciertas anomalías. Se da la casualidad de que, en la discriminación del ruido, el nivel de intensidad de los píxeles no es una función monótona entre el valor más alto que acostumbra a estar cerca del centro de gravedad del "blob" y el los píxeles que están en los bordes. Encontramos grupos de píxeles nulos antes de llegar al borde del "blob". En ese caso, las librerías de "openCV" [15] tratan el grupo entero de píxeles como 2 "blobs". El "blob" principal más grande con sus píxeles de intensidad no nula y un "blob" más pequeño que es sólo un contorno y que también contiene píxeles de intensidad no nula y que rodea la zona de píxeles nulos. Como en el proceso de búsqueda de los asteroides, se irán eliminando todos los cuerpos que no correspondan a un asteroide, se puede dar el caso que eliminemos el "blob" principal y nos dejemos el "blob" más pequeño que rodea los píxeles nulos. Por lo que, en este apartado, borraremos, todos los "blobs" cuyo centro de gravedad tenga un valor nulo.

Así mismo, se mirará los "blobs" que tengan más de un píxel con un valor máximo (píxel rodeado por píxeles de valor inferior), es decir máximos locales, y se intentará separar los que tengan esos máximos separados por una distancia mínima. Para ello se mirará de desplazarse entre los 2 píxeles con intensidad máxima y se buscará mínimos de intensidad y se cambiará el valor de intensidad de cada mínimo por el valor nulo de manera a separar en 2 "blobs" diferentes. Esta técnica es más factible para "blobs" de un cierto tamaño (>80 píxeles).

A continuación, como ejemplo, se muestra la fotografía nº1 del quinteto nº31. En rojo está rodeado el píxel de máxima intensidad en este caso de un asteroide y en azul está rodeado el píxel de máxima intensidad de una estrella. Como se ve en la figura 26, los 2 "blobs" se han unido en un solo "blob" ya que los dos objetos son de gran tamaño y están muy próximos. En este quinteto en particular es muy importante poder separar este asteroide de la estrella adyacente en esta precisa fotografía, ya que sólo en una única fotografía (fotografía nº0) están realmente separados y ésta (fotografía nº1) es la fotografía donde hay más posibilidades de separarlos. En el caso de no poder separarlos, el programa no podrá detectar el asteroide, ya que será borrado al estar "pegado" a una estrella con un máximo de intensidad superior al suyo. (10161>7284)

```
1 mysources[1][373:393,320:329]
```

```
array([[  677,   749,   530,   313,   216,   182,   171,   164,   162],
       [ 2415,  3102,  1796,   659,   285,   201,   177,   165,   161],
       [ 4459,  7284,  4506,  1357,   374,   216,   180,   167,   162],
       [ 2771,  5443,  4154,  1466,   392,   213,   177,   168,   163],
       [  886,  1612,  1438,   691,   298,   201,   178,   169,   164],
       [  320,   405,   379,   275,   208,   185,   173,   168,   168],
       [  200,   215,   211,   191,   178,   174,   168,   164,   165],
       [  172,   182,   179,   169,   169,   170,   165,   164,   160],
       [    0,   166,   164,   164,   162,   161,   161,   162,   165],
       [  160,   163,   166,   165,   161,   164,   164,   161,   160],
       [    0,     0,     0,   161,   163,   163,   164,   169,   166],
       [    0,   161,   162,   160,   159,   161,   162,   168,   175],
       [    0,   159,   162,   160,   164,   169,   172,   176,   179],
       [    0,     0,   163,   167,   170,   177,   193,   206,   210],
       [  159,   161,   160,   164,   173,   197,   239,   291,   306],
       [    0,     0,   165,   171,   195,   253,   422,   689,   762],
       [  161,   159,   164,   177,   223,   372,   998,  2592,  3224],
       [  162,   166,   171,   187,   234,   469,  1806,  6836, 10161],
       [  163,   165,   169,   189,   234,   415,  1438,  5528,  8894],
       [  160,     0,   162,   176,   205,   292,   599,  1407,  1966]])
```

Figura 26- Fotografía nº1 del quinteto nº31 antes de la separación

En la figura 27, se ve la misma fotografía en el mismo punto después de que se haya ejecutado el algoritmo y se hayan borrado los píxeles con la mínima intensidad entre los máximos de los 2 cuerpos que estaban unidos.

Se ha seguido marcando los centros de cada uno de los cuerpos, así como los límites de cada uno de ellos. El cuerpo marcado en rojo es el asteroide y el cuerpo marcado en azul es una estrella. En este caso, no importa que sólo haya 2 fotografías en que se ve el asteroide separado de los demás cuerpos (fotografías nº0 y nº1) ya que el programa es capaz de saber deducir si en las demás fotografías el asteroide ha interceptado una estrella.

De la misma manera se buscará para “blobs” más pequeños con varios máximos locales, píxeles del “blob” que estén rodeados por un número mínimo de píxeles de intensidad nula y se les cambiará el valor por el valor 0. Es decir que se va a mirar de romper “blobs” pequeños que están formados de dos o más “sub-blobs”

Este hecho se ve reflejado en las figuras 28 y 29 donde aparece la fotografía nº0 del quinteto º22 que ya hemos introducido en un apartado anterior. En ellas, se ve un asteroide y una estrella ligeramente unidos y que el programa separa antes de la etapa de eliminación de las estrellas. En azul está la estrella y en rojo está el asteroide. En verde claro, está rodeado el píxel que une los 2 “sub-blobs”.

```
1 mynewsources[1][373:393,320:329]
```

```
array([[  677,   749,   530,   313,   216,   182,   171,   164,   162],
       [ 2415,  3102,  1796,   659,   285,   201,   177,   165,   161],
       [ 4459,  7284,  4506,  1357,   374,   216,   180,   167,   162],
       [ 2771,  5443,  4154,  1466,   392,   213,   177,   168,   163],
       [  886,  1612,  1438,   691,   298,   201,   178,   169,   164],
       [  320,   405,   379,   275,   208,   185,   173,   168,   168],
       [  200,   215,   211,   191,   178,   174,   168,   164,   165],
       [  172,   182,   179,   169,   169,   170,   165,   164,     0],
       [    0,   166,   164,   164,   162,   161,   161,   162,     0],
       [    0,   163,   166,   165,   161,   164,   164,   161,     0],
       [    0,     0,     0,     0,     0,     0,     0,     0,     0],
       [    0,     0,     0,     0,     0,     0,     0,     0,   175],
       [    0,     0,     0,     0,     0,     0,     0,     0,   179],
       [    0,     0,     0,     0,     0,     0,   193,   206,   210],
       [    0,     0,     0,     0,     0,   197,   239,   291,   306],
       [    0,     0,     0,     0,   195,   253,   422,   689,   762],
       [    0,     0,     0,   177,   223,   372,   998,  2592,  3224],
       [    0,     0,     0,   187,   234,   469,  1806,  6836, 10161],
       [    0,     0,     0,   189,   234,   415,  1438,  5528,  8894],
       [    0,     0,     0,   176,   205,   292,   599,  1407,  1966]])
```

Figura 27 – Fotografía nº1 del quinteto nº31 después de la separación

```
[32] 1 mysources[0][285:295,420:430]
```

```
array([[1447, 1432, 1433,    0,    0,    0, 1431,    0,    0,    0],
       [1427, 1424, 1437,    0,    0, 1430, 1477, 1464,    0,    0],
       [   0,    0,    0,    0,    0, 1456, 1519, 1519, 1434,    0],
       [   0,    0,    0,    0,    0, 1437, 1474, 1474, 1432,    0],
       [   0, 1445, 1459, 1434, 1425,    0, 1428,    0,    0,    0],
       [   0, 1453, 1491, 1462,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1425, 1428,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 28 – Fotografía nº0 del quinteto nº22 antes de la separación

```
1 mynewsources[0][285:295,420:430]
```

```
array([[1447, 1432, 1433,    0,    0,    0, 1431,    0,    0,    0],
       [1427, 1424, 1437,    0,    0, 1430, 1477, 1464,    0,    0],
       [   0,    0,    0,    0,    0, 1456, 1519, 1519, 1434,    0],
       [   0,    0,    0,    0,    0, 1437, 1474, 1474, 1432,    0],
       [   0, 1445, 1459, 1434,    0,    0, 1428,    0,    0,    0],
       [   0, 1453, 1491, 1462,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1425, 1428,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0]])
```

Figura 29 – Fotografía nº0 del quinteto nº22 después de la separación

5.Eliminación de las estrellas

Para eliminar las estrellas, nos guardaremos los máximos de cada “blob” para cada una de las 5 fotografías del quinteto y compararemos la posición de ese objeto con los objetos de las demás fotografías que están a menos de 3 píxeles de ese máximo. Si las fotografías

fueran perfectas, simplemente tendríamos que buscar los máximos que coinciden exactamente con el máximo inicial. Si encontramos grupos de 3,4 o 5 “blobs” que verifican las condiciones de proximidad, son borrados automáticamente. Puede sorprender que se borren grupos de “blobs” que no sean de 5 (uno en cada foto), ya que se supone que las estrellas deben salir en cada una de las fotos del quinteto. Pero como, ya se ha dicho anteriormente, puede que una o varias estrellas de ese supuesto quinteto se hayan fusionado por proximidad con un asteroide más grande o con una estrella más grande, y el máximo al cual se le compara sea del asteroide más grande o de la estrella más grande.

Para entender, la manera de borrar las estrellas tomaremos el ejemplo de la estrella que hemos separado del asteroide en la fotografía nº1 del quinteto nº31. En las figuras siguientes mostraremos de color amarillo las coordenadas de la posición posible del píxel de máxima intensidad en las diferentes fotografías del quinteto para asegurar que una estrella sea borrada.

Como se puede ver en las figuras siguientes los máximos de intensidad para los diferentes “blobs” que están en una posición parecida en las diferentes fotografías son los siguientes:

Fotografía nº0 quinteto nº31 “blob” situado en las coordenadas (390,328) y de intensidad=11368

Fotografía nº1 quinteto nº31 “blob” situado en las coordenadas (390,328) y de intensidad=10161

Fotografía nº2 quinteto nº31 “blob” situado en las coordenadas (390,328) y de intensidad=10078

Fotografía nº3 quinteto nº31 “blob” situado en las coordenadas (391,328) y de intensidad=13209

Fotografía nº4 quinteto nº31 “blob” situado en las coordenadas (391,328) y de intensidad=9712

```
[51]  1 mynewsources[0][383:397,324:333]

array([[    0,     0,     0,     0,   169,   166,     0,     0,     0],
       [    0,   164,     0,   163,   174,   170,     0,     0,     0],
       [  162,   170,   166,   174,   181,   178,   166,   167,   163],
       [  166,   180,   186,   196,   216,   202,   181,   172,   166],
       [  169,   198,   241,   296,   310,   275,   224,   196,   179],
       [  184,   238,   422,   716,   800,   570,   333,   226,   190],
       [  211,   315,   857,  2647,  3437,  1670,   590,   273,   205],
       [  220,   388,  1407,  6597, 11368,  4852,  1015,   321,   213],
       [  214,   353,  1078,  4944, 10675,  5392,  1051,   334,   209],
       [  201,   272,   516,  1328,  2417,  1644,   574,   283,   215],
       [  183,   219,   276,   391,   517,   446,   301,   222,   195],
       [  173,   184,   203,   236,   274,   245,   216,   192,   172],
       [  164,   173,   181,   185,   203,   207,   192,   177,   171],
       [    0,   162,   171,   171,   185,   180,   173,   173,   166]])
```

Figura 30 – “blob” de la fotografía nº0 del quinteto nº31

En la fotografía de la figura 30, la posición del máximo de los “blobs “similares a éste en las demás fotografías tienen que estar en las coordenadas marcadas en amarillo para que este “blob” sea considerado una estrella, es decir los máximos en las demás fotografías tienen que estar en las coordenadas:

(389,326) o (390,326) o (391,326) o (388,327) o (389,327) o (390,327) o (391,327) o (392,327) o (388,328) o (389,328) o (390,328) o (391,328) o (392,328) o (388,329) o (389,329) o (390,329) o (391,329) o (392,329) o (389,330) o (390,330) o (391,330)

Lo mismo podríamos aplicar a las demás fotografías y encontraríamos coordenadas parecidas.

```
[52]  1 mynewsources[1][383:397,324:333]
```

```
array([[    0,     0,     0,     0,     0,     0,     0,   162,   161],
       [    0,     0,     0,     0,   175,   168,   161,   164,   159],
       [    0,     0,     0,     0,   179,   177,   169,   164,     0],
       [    0,     0,   193,   206,   210,   202,   182,   167,     0],
       [    0,   197,   239,   291,   306,   266,   217,   192,     0],
       [  195,   253,   422,   689,   762,   541,   333,   232,   189],
       [  223,   372,   998,  2592,  3224,  1864,   685,   303,   205],
       [  234,   469,  1806,  6836, 10161,  5599,  1469,   385,   218],
       [  234,   415,  1438,  5528,  8894,  5197,  1441,   386,   223],
       [  205,   292,   599,  1407,  1966,  1331,   586,   288,   205],
       [  177,   216,   291,   406,   466,   386,   279,   225,   193],
       [  169,   180,   202,   227,   250,   236,   207,   187,   172],
       [  165,   168,   176,   185,   199,   194,   180,   172,   169],
       [    0,     0,     0,   176,   180,   172,   170,     0,     0]])
```

Figura 31 - "blob" de la fotografía nº1 del quinteto nº31

```
[53]  1 mynewsources[2][383:397,324:333]
```

```
array([[ 6579,  5569,  1930,   522,   251,   196,   182,   168,   165],
       [ 5684,  5677,  2125,   571,   262,   207,   184,   171,   167],
       [ 1989,  2088,   952,   397,   256,   212,   197,   177,   171],
       [  524,   552,   393,   297,   267,   224,   203,   182,   170],
       [  256,   286,   318,   359,   359,   291,   227,   203,   186],
       [  221,   296,   506,   861,   909,   585,   336,   232,   195],
       [  229,   374,  1048,  3055,  3764,  1927,   670,   295,   208],
       [  244,   427,  1589,  6578, 10078,  5092,  1334,   392,   219],
       [  225,   383,  1208,  4830,  8471,  4727,  1329,   398,   226],
       [  205,   280,   593,  1587,  2583,  1712,   667,   309,   213],
       [  185,   215,   301,   477,   624,   506,   317,   231,   196],
       [  174,   191,   216,   244,   278,   258,   223,   196,   183],
       [  165,   177,   185,   200,   211,   196,   190,   178,   172],
       [  159,   167,   171,   175,   185,   177,   175,   172,   163]])
```

Figura 32 - "blob" de la fotografía nº2 del quinteto nº31

```
  1 mynewsources[3][383:397,324:333]
```

```
array([[  162,     0,   163,   172,   174,   163,   167,   165,     0],
       [  171,   169,   171,   171,   178,   172,   174,   168,   161],
       [  167,   170,   179,   183,   185,   183,   176,   169,   161],
       [  172,   178,   199,   222,   227,   214,   195,   175,   168],
       [  183,   211,   268,   358,   383,   313,   244,   205,   181],
       [  202,   279,   484,   921,  1082,   698,   374,   249,   201],
       [  231,   392,  1045,  3160,  4473,  2205,   701,   325,   222],
       [  263,   513,  1900,  7978, 13209,  5822,  1313,   410,   241],
       [  276,   552,  2236,  9964, 16508,  7093,  1557,   454,   251],
       [  252,   455,  1510,  6249, 10391,  4776,  1178,   398,   244],
       [  222,   318,   700,  1944,  3160,  1782,   643,   306,   233],
       [  199,   236,   330,   530,   682,   511,   330,   241,   207],
       [  178,   201,   222,   271,   303,   269,   227,   209,   192],
       [  167,   177,   182,   201,   219,   212,   189,   182,   177]])
```

Figura 33 - "blob" de la fotografía nº3 del quinteto nº31

```
[55]    1 mynewsources[4][383:397,324:333]

array([[   0,    0,    0,    0,  166,  169,    0,  159,  161],
       [   0,  161,  163,  169,  167,  163,  165,  164,  160],
       [   0,  162,  171,  178,  175,  178,  173,  168,  159],
       [ 171,  172,  190,  208,  214,  201,  191,  180,  165],
       [ 170,  195,  243,  318,  334,  283,  225,  190,  169],
       [ 188,  250,  461,  852,  953,  616,  334,  228,  192],
       [ 210,  344, 1015, 2862, 3739, 2009,  662,  290,  209],
       [ 218,  413, 1593, 6272, 9712, 5037, 1270,  379,  225],
       [ 223,  379, 1266, 5430, 9657, 5377, 1399,  412,  234],
       [ 208,  291,  628, 1952, 3566, 2645, 1081,  432,  253],
       [ 195,  247,  436, 1229, 3209, 3718, 1780,  585,  282],
       [ 192,  228,  369, 1261, 4438, 6340, 3112,  837,  298],
       [ 178,  212,  314,  891, 3501, 6092, 3355,  888,  299],
       [ 174,  194,  245,  462, 1436, 2721, 1703,  569,  256]])
```

Figura 34 – "blob" de la fotografía nº4 del quinteto nº31

Pero, la condición de pertenencia a ese grupo de coordenadas es una condición necesaria para que el "blob" sea considerado una estrella, pero no es una condición suficiente. Se necesita también que no exista ningún movimiento uniforme en la posición de ese máximo. Por este motivo, hay que añadir una limitación en la distancia de los máximos que separan la primera fotografía y la última. En el caso que hayan 5 "blobs" la distancia máxima que puede separar los máximos de los "blobs" debe ser inferior a 4 píxeles. Si hubieran sólo 4 "blobs" la distancia debería ser inferior a 3 píxeles y para 3 "blobs" sólo 2 píxeles. Si la distancia fuera superior se tendría que tratar como un asteroide.

También hay que considerar un problema añadido durante la eliminación de las estrellas (el programa puede borrar un asteroide tomándolo por una estrella debido a la existencia de residuos). Un residuo es un "blob" de uno o dos píxeles (la mayoría de las veces es de 1 píxel) que se genera cuando se elimina el ruido de fondo. Esto se debe al hecho que en el caso de que no hayan 5 "blobs" (caso frecuente cuando se trata un asteroide, ya que los máximos están separados como mínimo de 1 píxel y se sale de la zona de las coordenadas de una estrella) Estos "residuos" de manera casual aparecen en una fotografía en la zona de posibles coordenadas de una estrella en otra de las fotografías y el programa puede considerar que se trata de una estrella y borrar los "blobs" correspondientes. Para evitar este problema, cuando se buscan los "blobs" en las coordenadas posibles de una estrella, se verifica la relación de las amplitudes entre los máximos de los 2 "blobs" así como la relación entre el número de píxeles de cada uno de ellos. Al tratarse los residuos de "blobs" de 1 píxel con un máximo cercano al umbral del ruido el programa considera que no es válido, aunque lo haya encontrado en la zona de coordenadas válida.

6.Eliminación de "blobs" especiales

Una vez, eliminada la mayoría de las estrellas, miraremos de eliminar los "blobs" que tengan características especiales. Por ejemplo, miraremos los "blobs" superiores, por ejemplo, a 9 píxeles cuyo máximo se encuentre en el borde del "blob" o que tenga un centro de gravedad nulo. La razón de este borrado es que los cuerpos celestes tienen una disposición más o menos simétrica alrededor de su centro de gravedad que acostumbra a estar cerca del píxel de intensidad máxima. Todos estos cuerpos totalmente disimétricos no son reales y simplemente han aparecido de manera aleatoria en el momento de la supresión del ruido. También se borran las anomalías que se encuentran en los bordes de la imagen (en la mayoría de los quintetos aparecen "blobs" con máximos situados en las coordenadas de los

bordes de la imagen y la intensidad de dichos máximos está totalmente descorrelada de la intensidad del resto de píxeles del “blob).

Así mismo, se eliminan los “blobs” cuyo número de píxeles sea inferior a 5 y que no estén cerca de otros “blobs” de imágenes contiguas (es decir que estén por encima de una distancia máxima) Esto obedece al hecho que si un “blob” tiene por ejemplo un solo píxel significa que está a una gran distancia de la tierra con relación a los demás “blobs” y que su movimiento teniendo en cuenta las escalas de tiempo que se usan entre cada fotografía, tiene que ser muy lento (por ejemplo si es de 1 píxel no puede moverse más de 2 píxeles entre una foto y la foto sucesiva). Así mismo, si buscamos un asteroide real el tamaño no puede tener variaciones excesivas en las fotos del mismo quinteto (por ejemplo, un “blob” de 1 píxel sólo se puede relacionar con “blobs” de 6 píxeles de tamaño máximo)

7.Eliminación las estrellas no borradas y de los cuerpos que están en una posición determinada

Como ya se ha dicho durante el proceso de la eliminación de las estrellas, los borrados de estrellas sólo conciernen a 3 ,4 o 5 fotos de los quintetos. Si suponemos, por ejemplo, que hemos borrado una estrella grande en las 5 fotos de un quinteto, y que, en 3 de las fotografías debido a su proximidad y a la supresión del ruido, había una estrella más pequeña pegada a la estrella grande, han quedado 2 fotografías con la estrella pequeña que no ha sido borradas. En este apartado se borran las estrellas que han quedado desligadas de sus homólogas en las otras fotos del quinteto. Para ello se busca en las demás fotografías que no hay ningún objeto en una posición cercana a la estrella que se quiere borrar y una vez testeado este punto se busca en esas mismas fotos antes del borrado de las estrellas que haya algún objeto con un máximo local cerca del mismo valor. En ese caso significaría que existía un objeto en una posición muy cercana que fue borrado durante la etapa de eliminación de las estrellas. Si es ése el caso, significa que el objeto que se está estudiando es una estrella y debe ser borrado.

El borrado de estrellas desparejadas es un poco problemático y queda pendiente de afinar en el caso de pequeños asteroides ya que, en algunos casos puntuales, el programa puede borrar los “blobs” de las fotografías que están en los extremos, aunque correspondan a asteroides.

Después del borrado del apartado anterior, se calcula los objetos que se han desplazado menos de 18 píxeles en cada uno de los ejes (x,y) (18 píxeles es el desplazamiento máximo que se supone que puede tener un asteroide entre una foto y la foto sucesiva para los intervalos de tiempo considerados. En la mayoría de los casos 18 píxeles es válido cuando el intervalo entre cada una de las fotos es el mismo. Si no es el mismo hay que sumarle un “offset” proporcional a 18 y cuyo coeficiente de proporcionalidad depende de la relación entre los intervalos de toma de fotografías). Si se encuentra un objeto dentro de ese intervalo de 18 píxeles se verificará, que la relación entre los máximos de intensidad está por debajo de un umbral. A todos los objetos que no verifiquen esa condición se les cambiará la intensidad a 0.

8.Búsqueda final de asteroides

En esta parte final se asocian todos los objetos de la fotografía nº0 con todos los objetos de las fotografías sucesivas que no han sido eliminados previamente, después se pasa a la fotografía nº1 y se asocia con todos los objetos de esa fotografía con todos los objetos de

las fotografías sucesivas hasta llegar a la fotografía nº4. A partir de ahí se forman quintetos, cuartetos, tercetos y dúos de objetos. Se comparan los desplazamientos entre cada una de las fotografías de los objetos de cada una de las agrupaciones y se limita de manera "gruesa" la variación del desplazamiento entre el mínimo y el máximo para obtener el desplazamiento más uniforme posible. En esta primera etapa la separación entre los desplazamientos mínimo y máximo no deben superar el valor de 5 píxeles.

A partir de esta primera criba sólo quedan un cierto número de quintetos, cuartetos, tercetos y dúos que verifican la condición del párrafo anterior sobre los desplazamientos. En este punto, se pueden eliminar dúos, tercetos y cuartetos que sean subgrupos de otros grupos más grandes. Por ejemplo, nos podemos encontrar dúos en que todos sus componentes aparezcan en uno o varios tercetos o tercetos cuyos componentes formen parte de cuartetos y finalmente de cuartetos cuyos compontes formen parte de quintetos. Todos los objetos repetidos de rango inferior serán eliminados. El rango viene determinado por el número de objetos. La agrupación de mayor rango es la de 5 objetos, luego la de 4 objetos y así sucesivamente hasta 2 objetos.

A partir de ahí se hace una segunda criba más fina de la uniformidad de todas las agrupaciones. Se verifica también que la disparidad en el número de píxeles de cada uno de los grupos de objetos. A todos los objetos que no verifican las condiciones impuestas, se les pone el valor de intensidad a 0.

Seguidamente, los quintetos se aceptan como buenos y en los cuartetos, tercetos y dúos se verifica que interceptan alguna de las estrellas que se han borrado anteriormente. Para ello, se chequea que los píxeles interceptados pertenecen a un "blob" más grande borrado durante el período de eliminación de las estrellas para todas las fotografías del quinteto. Es decir, que se verifica que el máximo de ese "blob" está cerca de los máximos de otros "blobs" de las demás fotografías justo antes del borrado de las estrellas. Si esta condición no se cumple, se pone a 0 el valor de intensidad de todos sus píxeles ya que no intercepta ninguna estrella borrada.

Se hacen finalmente otros tipos de verificaciones en lo que concierne la relación de amplitud entre el valor máximo de un objeto en una agrupación y su valor mínimo. Así mismo se verifica que el número de píxeles del objeto más grande tenga una relación coherente con el objeto con el número de píxeles más pequeño. Todas las agrupaciones que tengan objetos con relaciones de amplitud o longitud fuera del límite serán borradas.

9.Resultados

A partir de ahí obtendremos principalmente uno o varios grupos de trayectorias, que se presentarán en fichero **csv** para cada uno de los quintetos. El número de asteroides en algunos casos de quintetos dependerá de cuan uniforme queramos que sean las trayectorias.

Como ya se ha dicho anteriormente, si sólo tenemos una fotografía en el quinteto donde el objeto está separado de los demás cuerpos y no hemos logrado una separación en el apartado nº4, no detectaremos nada.

Así mismo si en la separación del ruido de fondo, en una fotografía del quinteto no aparece ningún objeto, pero en las demás fotografías aparece y parece tener movimiento uniforme, este objeto no podrá ser dado por válido y se considerará como ruido

El programa generará unas máscaras y unas imágenes modificadas para ser usadas posteriormente en los algoritmos de Deep Learning [7]. Se guardarán en los ficheros la mediana y la desviación estándar que servirán igualmente en los algoritmos posteriores.

2.3 Algoritmos de "Deep Learning"

Desde hace un cierto tiempo, se está resolviendo la búsqueda de asteroides por métodos de tratamiento de imágenes de manera completa o por tratamiento de imágenes + algoritmos de Machine Learning [4]. En este trabajo, se va a mirar de resolver el problema de la detección con algoritmos de Deep Learning [7] completamente y/o con una mezcla de tratamiento de imágenes + algoritmos de Deep Learning. [7]

A partir de este momento analizaremos qué tipo de algoritmo supervisado de Deep Learning [7] utilizar en este caso concreto de la localización de asteroides, teniendo en cuenta que ya tenemos lo que se llama "ground truth" [11] (posición constatada del objeto) y podemos así mismo a partir de los datos obtenidos anteriormente obtener la caja o "box" que encuadra la solución o los píxeles que forman parte de la solución.

Para llevar a cabo este análisis, vamos a ver las diferentes estructuras que se utilizan, partiendo de las estructuras más "simples" a las más elaboradas.

Esto se traduce analizando primeramente las estructuras de una red Deep Learning [7] para la clasificación de fotografías, seguidamente para la clasificación y localización, posteriormente la detección de diferentes objetos por clases y finalmente la segmentación de la fotografía por clases de objetos.

La figura 35 nos muestra esta evolución en el reconocimiento de imágenes:

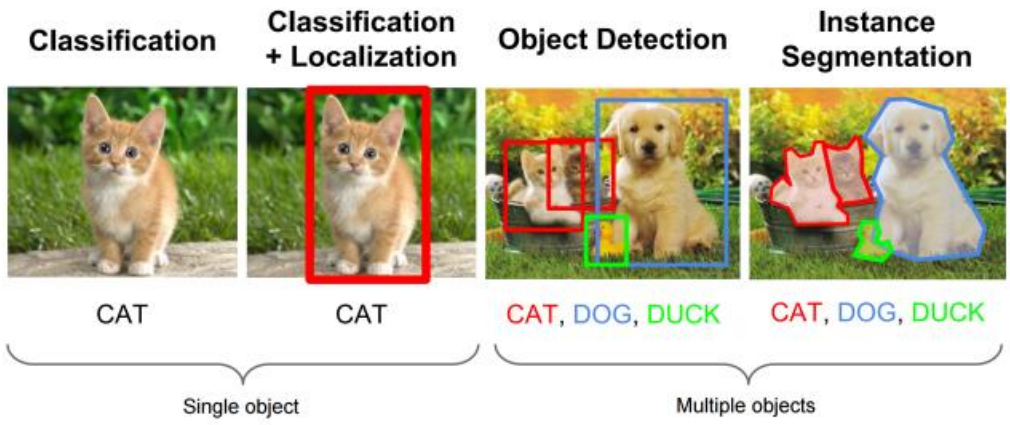


Figura 35 – Evolución del reconocimiento de imágenes

2.3.1 Clasificación de imágenes

El origen de este tipo de algoritmos se basa en las redes neuronales artificiales adaptadas para el tratamiento específico de este tipo de datos. Por ese motivo, vamos a comenzar explicando el funcionamiento básico de las redes neuronales artificiales.

2.3.1.1 Red neuronal artificial (ANN)

Una red neuronal artificial ANN es un modelo computacional inspirado en la forma en que las redes neuronales biológicas en el cerebro humano procesan la información. Dichas redes están basadas en las neuronas llamadas también nodos. Cada nodo recibe datos de otros nodos o de una fuente externa y calcula una salida. Cada entrada tiene un peso asociado (w), que se asigna en función de su importancia relativa con relación a las otras entradas. El nodo aplica una función f (definida a continuación) a la suma ponderada de sus entradas como se muestra en la figura a continuación: [18]

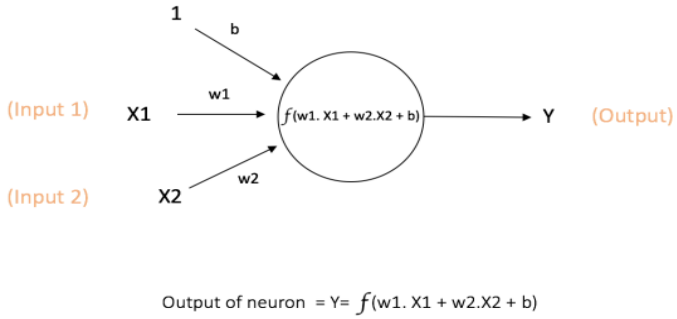


Figura 36- Neurona

La red anterior toma las entradas numéricas X1 y X2 y tiene los pesos w1 y w2 asociados con esas entradas. Además, hay otra entrada 1 con peso b asociado a ella. La salida Y de la neurona se calcula como se muestra en la Figura 36. La función f no es lineal y se denomina Función de activación. El objetivo de la función de activación es introducir la no linealidad en la salida de una neurona. Esto es importante porque la mayoría de los datos del mundo real no son lineales y queremos que las neuronas aprendan estas representaciones no lineales. Existen diferentes funciones de activación, pero las más usadas son: [18]

Sigmoid: toma una entrada de valor real y la escala para que oscile entre 0 y 1.

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

Tanh: toma una entrada de valor real y lo aplasta al rango [-1, 1]

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

ReLU (rectificador lineal): Toma una entrada de valor real y la ajusta a cero

$$f(x) = \max(x, 0)$$

Softmax: es una generalización de Sigmoid para j variables

$$f(z_i) = \frac{e^{z_i}}{\sum_j z^j}$$

Las funciones de activación de las tres primeras funciones se muestran en la figura37.

Las redes neuronales artificiales se componen de toda una serie de capas de neuronas interconectadas. La primera capa es llamada capa de entrada (“input layer”), la última capa es llamada capa de salida (“ouput layer”) y todas las demás capas se llaman capas escondidas (“hidden layers”). Si tenemos más de una capa denominada “hidden layer”, hablamos del concepto de “Deep Learning”.

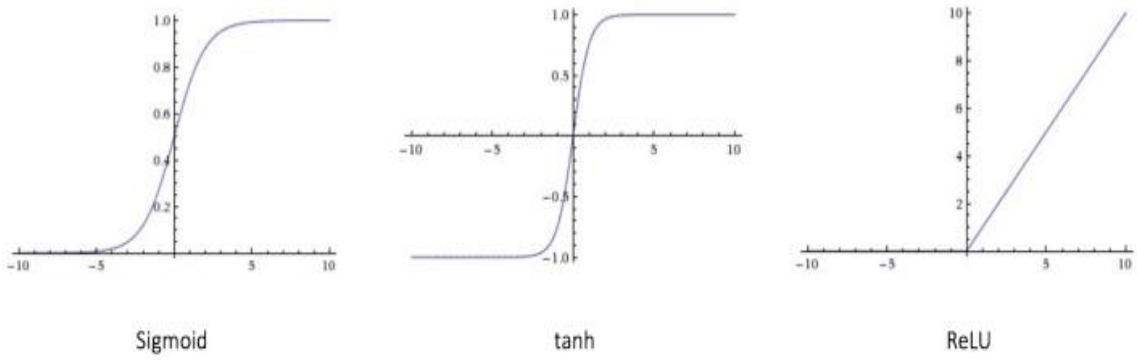


Figura 37 - Curvas de las funciones de activación

Como ya hemos visto anteriormente, cada neurona tiene entradas y salidas ponderadas por pesos (valores w de la figura 36). El objetivo del entrenamiento de la red neuronal es asignar pesos correctos para todos los nodos de la red de manera que para una entrada de datos conocida obtengamos la salida esperada (Dado un vector de entrada, los pesos de las neuronas determinan cuál es el vector de salida) [18]

Inicialmente, todos los pesos de los nodos(neuronas) se asignan aleatoriamente. Para cada entrada en el conjunto de datos de entrenamiento, la ANN se activa y se observa su salida. Este resultado se compara con el resultado deseado que ya conocemos, gracias a la función de coste que explicaremos más adelante (A este funcionamiento se le llama “Forward Propagation”) [18]

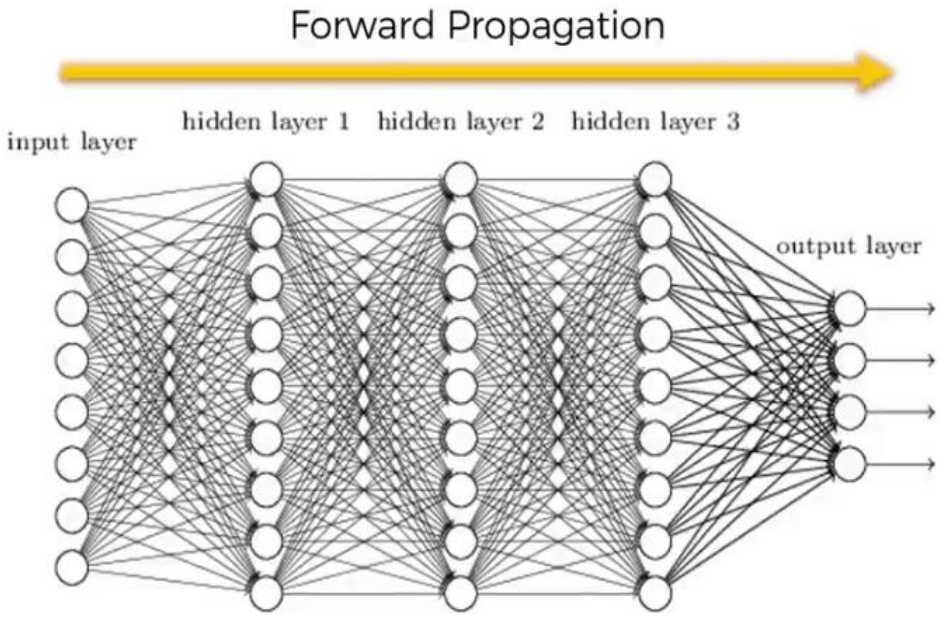


Figura 38 – “Forward Propagation” en una red neuronal

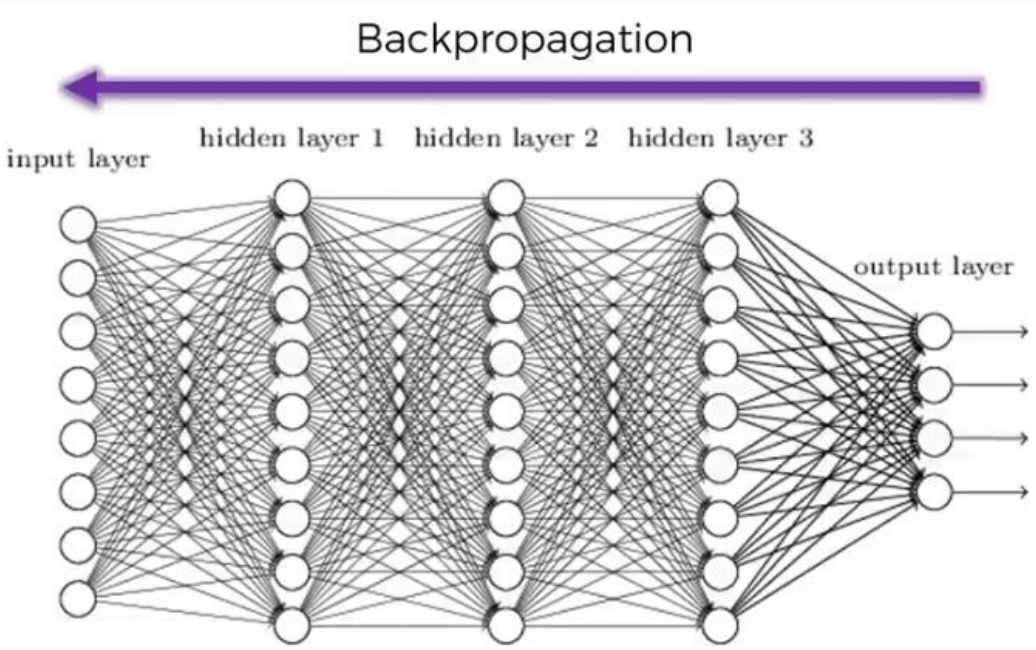


Figura 39 - "Back Propagation" en una red neuronal

En el caso de la función de coste que se citado en el entrenamiento de la red neuronal, existen varias versiones, aunque entre las más usadas están la MSE (Mean Square Error) y la CE (Cross Entropy).

La función de pérdida cuadrática media (MSE) se usa ampliamente en regresión lineal como medida de rendimiento

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

El objetivo de la MSE es la minimización del cuadrado de la diferencia entre el valor de salida real y el valor de salida predicho.

La CE (Cross Entropy) se usa comúnmente en la clasificación binaria (se asume que las etiquetas toman los valores 0 o 1) como una función de pérdida (para la multi-clasificación, se usa la entropía cruzada multi-clase), que se calcula mediante:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log(\hat{y}^{(i)}) + \left(1 - y^{(i)}\right)\log(1 - \hat{y}^{(i)})\right]$$

La CE mide la divergencia entre dos distribuciones de probabilidad, si la entropía cruzada es grande, significa que la diferencia entre dos distribuciones es grande, mientras que, si la entropía cruzada es pequeña, significa que dos distribuciones son similares entre sí.

El error obtenido de la función de coste se "propaga" hacia la capa anterior, y así sucesivamente hasta la capa inicial. Este error se observa y los pesos se "ajustan" en consecuencia. (se calculan las derivadas parciales de la función de coste en función de las variables de cada capa y se calculan los nuevos valores de los pesos para cada neurona en cada capa). Una vez los pesos ajustados se vuelven a entrar nuevos datos y se observa la nueva salida y se compara con la salida esperada a través de la función de coste. El error se propaga de nuevo hacia la entrada y

así sucesivamente. Este proceso se repite hasta que el error de salida está por debajo de un umbral predeterminado. [18]

El cálculo de variación de los pesos de cada neurona en cada capa por el algoritmo de “back propagation” viene dado por:

$$W_{new} = W_{old} + \alpha * \left(\frac{\partial E}{\partial W}\right)_{W_{old}}$$

$$\alpha = Learning\ Parameter\ (between\ 0\ and\ 1)$$

Una vez que el algoritmo anterior termina, tenemos una red neuronal "aprendida" que, consideramos, está lista para trabajar con "nuevas" entradas. Se dice que esta ANN ha aprendido de varios ejemplos (datos etiquetados) y de sus errores (propagación de errores).

Cuantos más datos se utilicen para el aprendizaje, más precisa será la predicción. Estas redes básicas se aplican en muchos problemas, pero existen ciertos tipos de datos en que estas redes básicas son ineficientes. Estos datos que necesitan unas redes neuronales más elaboradas son las imágenes.

Si se utilizaran redes neuronales artificiales básicas, las imágenes que son tensores (matrices multidimensionales) se transformarían en vectores de 1 columna y n líneas. Por ejemplo, una imagen en color de 28 píxeles x 28 píxeles se transformaría en un vector de una columna y de 28 x 28 x 3 líneas.

El problema se puede ver en el ejemplo siguiente donde se muestran 2 números centrados en una foto:

En este caso, la red neuronal obtiene una muy buena predicción, pero si desplazamos el número a un lado de la foto obtenemos el resultado siguiente:

Esto se debe a que nuestra red solo aprendió el patrón de un "8" perfectamente centrado. No tiene absolutamente ninguna idea de lo que es un "8" fuera del centro. Sabe exactamente un patrón y un patrón solamente.

Eso no es muy útil en el mundo real. Los problemas del mundo real nunca son tan limpios y simples. Entonces tenemos que descubrir cómo hacer que nuestra red neuronal funcione en los casos en que el "8" no está perfectamente centrado.

Existe una solución para encontrar un "8" centrado en una imagen. ¿Qué pasa si escaneamos alrededor de la imagen para posibles "8" en secciones más pequeñas, una sección a la vez, hasta que encontremos una?

Este enfoque se llama ventana deslizante. Es la solución de fuerza bruta. Funciona bien en algunos casos limitados, pero es realmente ineficiente. Debe verificar la misma imagen una y otra vez buscando objetos de diferentes tamaños.

Debería haber alguna manera de hacer que la red neuronal sea lo suficientemente inteligente como para saber que un "8" en cualquier parte de la imagen es lo mismo sin todo ese entrenamiento adicional. Esa solución es la llamada red neuronal" convolucional". Pero, para explicar las denominadas redes" convolucionales", explicaremos primero lo que es una operación de convolución. [19]

2.3.1.2 Convolución

La operación de convolución se aplica sobre los valores de intensidad contenidos en los píxeles de una imagen, ya que una imagen es una matriz de píxeles:

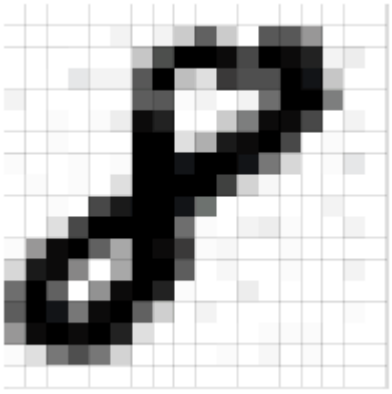


Figura 40 - Píxeles

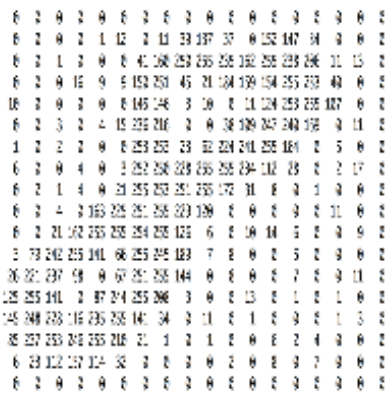


Figura 41 - Valor Intensidad

Por ejemplo, en el caso de las dos figuras justo encima, se trata una fotografía en blanco y negro donde se ha dibujado un 8 en posición oblicua y en la que la figura 40 muestra los píxeles y la figura 41 los valores de la intensidad para cada uno de esos píxeles. [19]

Una imagen de m*n píxeles se representa por un tensor m x n x t, es decir t matrices de 2 dimensiones de m líneas por n columnas. En la mayoría de los casos, t puede tener el valor 1 (imagen en blanco y negro) o 3 (imagen en color formato RGB).

Si analizáramos píxel por píxel veríamos que en el caso de una imagen en blanco y negro tendríamos una matriz 2d donde la posición del píxel en la imagen sería el número de la fila y el número de la columna y el valor de cada punto sería la intensidad del "gris" (255= blanco y 0=negro. En este ejemplo, en particular, la imagen está invertida). En el caso de una imagen en color tendríamos 3 matrices y las características de un píxel serían el mismo número de fila y el mismo número de columna de cada matriz y el color del píxel sería la combinación de los valores de cada una de esas matrices en ese píxel. [19]

El objetivo principal de la Convolución es extraer las características de la imagen de entrada. La Convolución preserva la relación espacial entre píxeles mediante el aprendizaje de

características de la imagen utilizando pequeños cuadrados de datos de entrada. Se llama filtro o “kernel” a la matriz que se va a usar para hacer la operación de convolución con la imagen.

Un ejemplo de la convolución es el siguiente:

Si consideramos una imagen de 5x5 píxeles y un filtro de 3x3 píxeles:

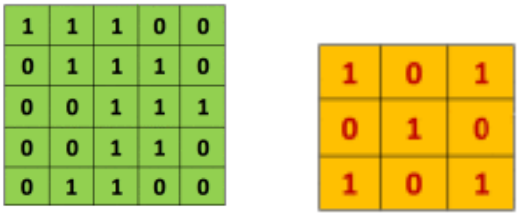
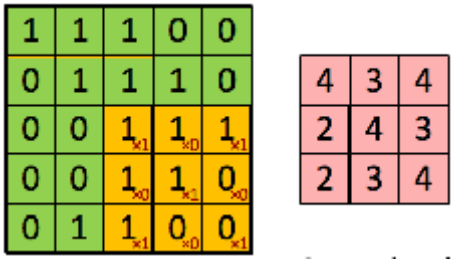


Imagen 5x5 *Filtro 3x3*

Obtendremos el resultado siguiente:



Resultado de la convolución

La convolución consiste en desplazar el filtro por toda la imagen y sumar los 9 valores obtenidos de la multiplicación de los 2 valores superpuestos (imagen + filtro). En la imagen superior se ve la superposición del filtro en el recuadro inferior izquierdo de la imagen. [19]

Seguidamente adjuntamos el resultado de la aplicación de varios filtros específicos en una imagen real:

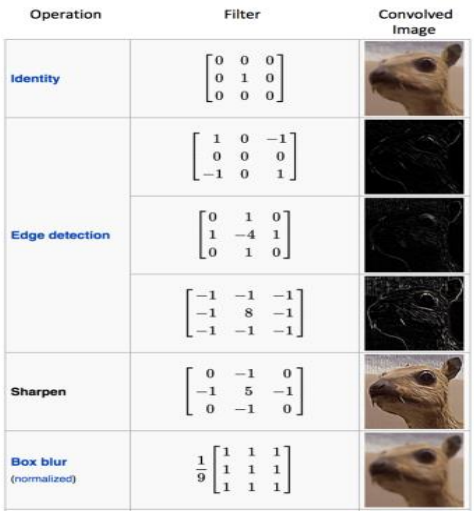


Figura 42 - Modificación de una imagen por convolución

Como se muestra en la figura anterior, podemos realizar operaciones como detección de bordes, nitidez y desenfoque simplemente cambiando los valores numéricos de nuestra matriz de filtro antes de la operación de convolución; esto significa que diferentes filtros pueden detectar diferentes características de una imagen, por ejemplo, bordes, curvas, etc… [19]

2.3.1.2 Red neuronal "convolucional"

En la figura siguiente, se expone uno de los primeros ejemplos de red neuronal "convolucional" simple usado para la clasificación de imágenes. En él se introduce una imagen de un perro o un gato y la red obtiene la probabilidad de que sea un perro o un gato y deduce la etiqueta (perro o gato) del valor más probable. Existen redes "convolucionales" más complejas para la clasificación, pero con ésta podemos estudiar varios de los elementos que componen otras redes para la clasificación, localización, detección y segmentación.

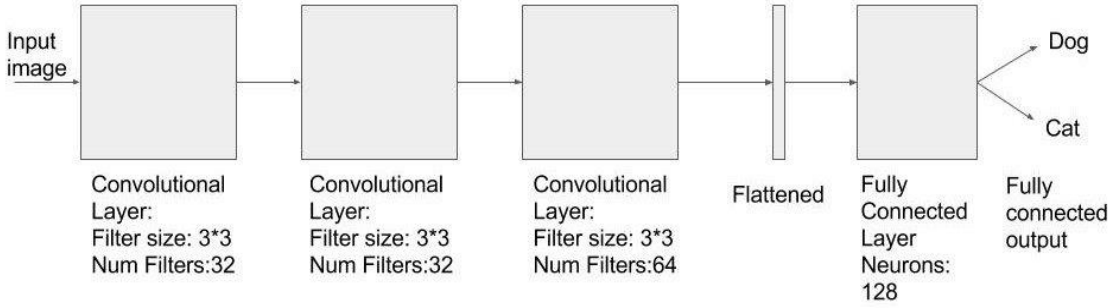


Figura 43 – Ejemplo de red "convolucional" para la clasificación de imágenes

Este ejemplo básico se divide principalmente en 3 partes:

- El bloque convolucional
- El bloque llamado "flattening"
- El bloque llamado "Fully Connected Layer"

2.3.1.2.1 Bloque convolucional

Este bloque se compone de diferentes funciones de convolución puestas en cascada. A la salida de cada función convolucional, se le aplica una función no lineal (en el caso de la convolución se suele utilizar la función ReLU). ReLU es una operación no lineal aplicada al valor de cada píxel y reemplaza todos los valores negativos de los píxeles en el mapa de características por cero y deja invariante los valores positivos.

Como ya se ha indicado anteriormente, para que los resultados sean más cercanos a los datos reales se necesita que cada bloque "convolucional" sea una función no lineal y la función de "convolución" es una operación lineal sobre matrices. En la figura siguiente, se ve un ejemplo del efecto de la función ReLU sobre una imagen.

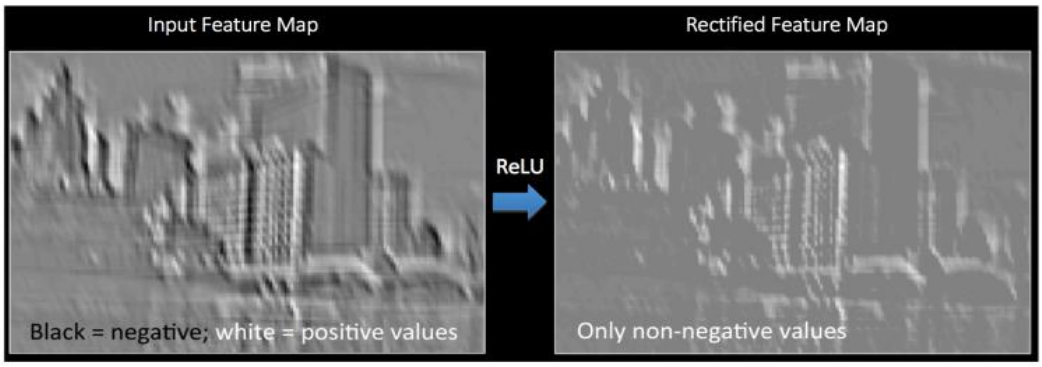


Figura 44 - Resultado de la aplicación de la función no lineal Relu

Pero el hecho de poner en cascada todas estas funciones convolucionales, se genera un número considerable de parámetros (características de la imagen de salida) que si no se controla pueden provocar un grave problema de memoria. Para solventar este inconveniente se añade lo que se llama una función de "pooling". Esta función de pooling reduce el número de parámetros de salida de una función convolucional antes de aplicarle la siguiente convolución. [19]

El "pooling" reduce la dimensionalidad de cada mapa de características, pero conserva la información más importante. El agrupamiento puede ser de diferentes tipos: Máximo, Media, Suma etc.

En el caso del tipo Máximo, que es el que se suele usar, definimos un vecindario espacial (por ejemplo, una ventana de 2 × 2) y tomaremos el elemento más grande del mapa de características rectificadas dentro de esa ventana.

Para entenderlo mejor, tenemos, en la figura 45, un ejemplo de un Max Pooling a través de una ventana 2x2 de una imagen a la cual se le ha aplicado previamente una convolución:

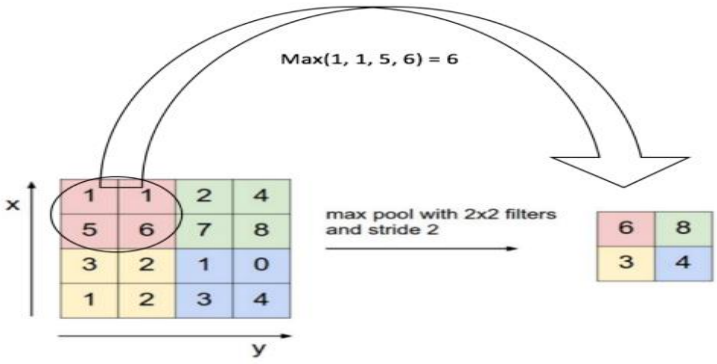


Figura 45 – Max Pooling

El "pooling" tiene varias ventajas: [19]

- las representaciones de entrada (dimensión de la función) son más pequeñas y más manejables
- reduce la cantidad de parámetros y cálculos en la red, por lo tanto, controla el sobreajuste

- hace que la red sea invariante para pequeñas transformaciones, distorsiones y traducciones en la imagen de entrada (una pequeña distorsión en la entrada no cambiará la salida de Max Pooling, ya que tomamos el valor máximo en un vecindario local).
- nos ayuda a llegar a una representación invariante a escala de nuestra imagen. Esto es muy importante ya que podemos detectar objetos en una imagen sin importar dónde se encuentren.

2.3.1.2.2 Bloque "flattening"

El Flattening permite de conectar la salida del bloque "convolucional" a la entrada de la red neuronal "fully connected". Dicha red neuronal admite como entrada un tensor de una dimensión que está formado de una serie de atributos. De esta manera se convierte m matrices 2d de sxr (s número de líneas y r número de columnas) en un vector 1d de m*s*r elementos.

En la figura siguiente hay un ejemplo de una matriz 2d de 3 líneas y de 3 columnas que es supuestamente la salida de un bloque "convolucional"

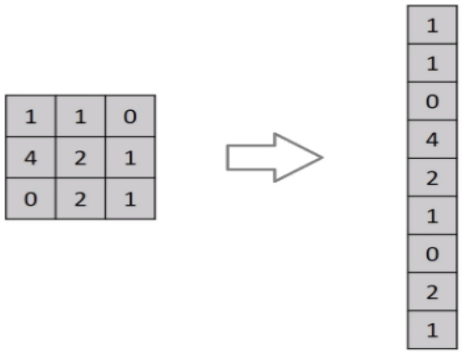


Figura 46 – Función de "Flattening"

2.3.1.2.3 Bloque "Fully Connected Layer"

El bloque final "Fully Connected Layer" corresponde a la clasificación propiamente dicha y está formado por una red neuronal artificial, la cual se ha explicado ya anteriormente. La combinación de los valores de las neuronas y sus coeficientes es lineal y se utiliza en la salida de la red una función de activación no lineal. En el caso de clasificación de imágenes, a la salida de las redes neuronales artificiales se acostumbra a usar 2 tipos de función de activación. Para una salida binaria (2 clases), se usa la función" Sigmoid" y para más de 2 clases, se utiliza la función "Softmax". [19]

2.3.1.2.4 Comportamiento global

Una red convolucional recibe una imagen de entrada y le aplica una serie de funciones en cascada llamadas convoluciones mezcladas con funciones no lineales("reLU") y con funciones reductoras de parámetros("pooling"). Al final de todas esas funciones, obtenemos un grupo de matrices que representan toda una serie de características de las diferentes partes de la imagen. Esas matrices se transforman en un vector de 1d y de múltiples líneas que se aplicarán a la entrada de una red neuronal artificial que a partir de un entreno previo nos dará la probabilidad de que la fotografía pertenezca a una clase en concreto (en nuestro ejemplo, nos dirá si el objeto de la foto pertenece a la clase perros o a la clase "gatos")

Todas las redes neuronales se entrenan con múltiples imágenes lo que nos permite de encontrar todos los parámetros w de las redes neuronales artificiales y todos los valores de los filtros de las funciones “convolucionales”.

Una parte de las imágenes de entrada se utilizarán para hacer el entreno de dichas redes y el resto servirá para probarlas, es lo que se acostumbrar a llamar imágenes de “training” e imágenes de “test”.

En los algoritmos de Machine Learning y en particular de “Deep Learning” siempre se hace este tipo de división para evitar el problema de sobreajuste o “overfitting”[20]. El “overfitting” se manifiesta cuando un sistema se entrena demasiado o se entrena con algunos datos extraños. El algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. Durante la fase de sobreajuste el éxito al responder las muestras de entrenamiento sigue incrementándose mientras que su actuación con muestras nuevas va empeorando.

Una división habitual es la llamada 70%-30% (70% de datos para el entreno y 30% de datos para la validación). Hay mejores reparticiones de los datos como la “cross-validation n folders” [21] que permite un ajuste más fino de los datos y cuya repartición se ajusta más a la repartición real de los datos.

Cuando hemos abordado el funcionamiento de las redes neuronales, hemos visto que recalculábamos durante el entrenamiento de nuestra red el valor de todos los pesos hasta optimizarlos(back-propagation). En la práctica, con una gran cantidad de datos, es más eficiente, a nivel computacional, de dividir el conjunto de datos de entrenamiento en “batches” e iterar esos batches hasta alcanzar el número de datos totales.

Además, en la implementación real, se puede añadir 2 “ajustes” que no se han citado durante la explicación del funcionamiento de las redes neuronales y que son el “batch-normalisation” y el “drop-out”.

Batch-normalisation:

> A medida que los datos fluyen a través de una red neuronal, los pesos y parámetros se ajustan a esos datos, haciendo a veces que los ajustes sean demasiado grandes o demasiado pequeños, llegando a provocar valores tan pequeños que no se observan avances en la optimización o valores tan grandes que provoquen errores de cálculo. Al normalizar los datos en cada “batch”, este problema se evita en gran medida. El “batch-normalisation” puede ayudar en un aprendizaje más rápido y en una mayor precisión general. [22]

Drop-out:

> El “Drop-out” se refiere a ignorar neuronas durante la fase de entrenamiento en cierta capa de neuronas. Dichas neuronas se eligen al azar y las neuronas elegidas son desactivadas durante la “forward propagation” o la “backward propagation”.
>
> Más técnicamente, en cada etapa de entrenamiento, las neuronas individuales se eliminan de la red con probabilidad 1-p o se mantienen con la probabilidad p, de modo que queda una red reducida. Las conexiones entrantes y salientes a esa neurona también se eliminan. [23]

El “Drop-out” se utiliza porque cuando todas las neuronas de una capa están conectadas desarrollan codependencia entre ellas durante el entrenamiento, lo que limita la potencia individual de cada neurona y da lugar también al “overfitting”.

2.3.2 Clasificación y localización de imágenes

La clasificación y la localización de imágenes es el paso siguiente a la clasificación. Lo que se busca en este caso no sólo consiste en catalogar una fotografía por un objeto que se haya en ella, pero también determinar la posición de dicho objeto. Aunque se utiliza normalmente como paso intermedio entre la clasificación y la detección, la localización es muy poco frecuente comparada con la detección ya que en la localización el número de objetos es fijo en cada fotografía y abarca una sola clase de objeto. [24]

(A partir de este momento hasta el final llamaremos CNN a la red neuronal “convolucional”, iniciales de Convolution Neuronal Network)

Si analizamos una CNN diseñada para este nuevo cometido, podemos encontrar que globalmente es muy parecida a la que se utilizó para la clasificación, pero en este caso existe ciertas diferencias en la salida.

Como se puede ver en el ejemplo de la figura 47, en la salida no sólo aparecerá la clase de objeto contenida en la foto. Para cada objeto de la foto, aparecerán 4 variables de salida que serán las posiciones del centro de gravedad del objeto(x,y), así como la longitud(h) y la anchura(w) de la caja o rectángulo que encuadra al objeto.

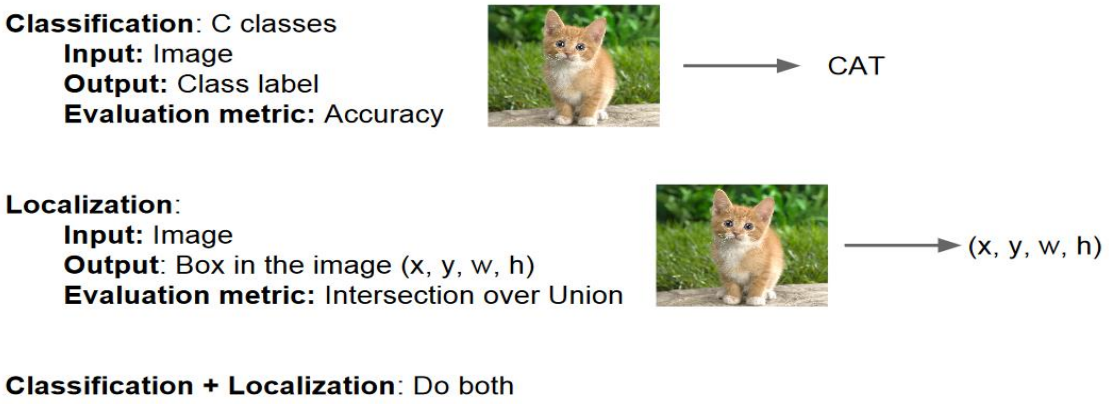


Figura 47 – Clasificación y localización

Una de las soluciones en este caso especial (Figura 48) es dividir la salida del último bloque de convolución“en 2 partes (“2 fully connected layers” o FC). Por una de las 2 FC haremos la clasificación por clases (“classification head”) y por la otra obtendremos los parámetros x,y,w,h de la caja(“regression head”) que encuadra el objeto clasificado por la primera red.

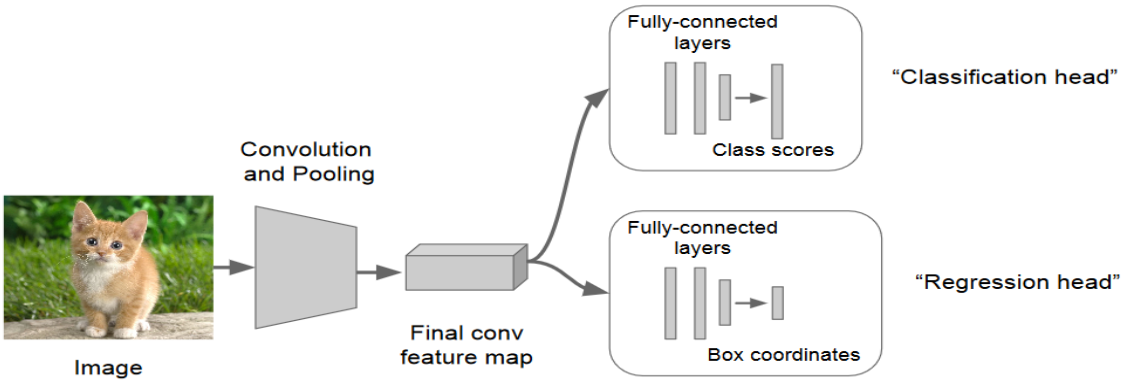


Figura 48- Red "convolucional" para clasificación y localización

La función de coste se compondrá de 2 funciones, una que nos permitirá hacer la comparación de la clase predicha del objeto con la clase real y la otra la comparación su posición predicha en la foto con su posición real "ground truth". Para la primera, ya hemos visto que podemos utilizar la función "Cross-Entropy". Para la segunda, utilizaremos la diferencia entre la posición predicha y la posición real. [24]

La función "Cross-Entropy" tiene, como recordatorio la fórmula siguiente:

$$\mathcal{L} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

donde M es el número de clases, $y_{o,c}$ es el indicador binario(0 o 1) si la clase c es la correcta clasificación para la observación "o" y $p_{o,c}$ la probabilidad predicha de la observación "o" sea de clase c.

En el caso binario (2 clases) la fórmula es la siguiente:

$$\mathcal{L} = -(y \log(p) + (1 - y)\log(1 - p))$$

Para la segunda función utilizaremos la métrica euclidiana con la función:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

2.3.3 Detección de objetos

La detección de objetos es parecida a la localización, pero en el caso de la localización existe un número fijo de objetos (que en la mayoría de los casos suele ser uno) y en el caso de la detección el número de objetos en cada fotografía puede ser variable.

Para detectar todo tipo de objetos en una imagen, podemos usar directamente lo que hemos aprendido hasta ahora de la localización de objetos. La diferencia es que queremos que nuestro algoritmo sea capaz de clasificar y localizar todos los objetos en una imagen, no solo uno.

La idea para lograrlo es recortar la imagen en múltiples imágenes y ejecutar la CNN en todas las imágenes recortadas para detectar un objeto.

El algoritmo sería el siguiente: [24][25][26]

1) Escoger una ventana de tamaño mucho más pequeño que el tamaño real de la imagen. Recortar la imagen a ese tamaño y pasarla por la CNN y hacer que la CNN haga las predicciones.
2) Deslizar la ventana y pasar las siguientes imágenes recortadas por la CNN.
3) Después de recortar todas las partes de la imagen con este tamaño de ventana, repetir todos los pasos nuevamente para un tamaño de ventana un poco más grande. Volver a pasar las imágenes recortadas por la CNN y dejar que haga predicciones.
4) Al final, se obtendrá un conjunto de regiones recortadas que tendrán algún objeto, junto con la clase y la caja delimitadora del objeto.

Esta solución se conoce como detección de objetos con ventanas deslizantes. Es una solución muy básica que tiene diferentes inconvenientes como los siguientes:

- Recortar múltiples imágenes y pasarlas a través de la CNN va a ser muy costoso desde el punto de vista computacional.

Estamos deslizando ventanas de forma cuadrada por toda la imagen, tal vez el objeto sea rectangular o tal vez ninguno de los cuadrados coincida perfectamente con el tamaño real del objeto. Aunque este algoritmo tiene la capacidad de buscar y localizar

- múltiples objetos en una imagen, la precisión del rectángulo delimitador sigue siendo mala.

Existen diferentes soluciones para resolver estos 2 problemas principales. Vamos a explicar a continuación algunas de las redes para detectar objetos que hasta el momento están entre las mejores.

Antes de detallarlas y aunque ya hayamos hablado de una de ellas, recordaremos diferentes funciones de coste y métricas que se suelen utilizar para la detección de objetos.

En lo que concierne las métricas, existen varias métricas entre las cuales hay la precisión y la IoU.

- Métricas

  Precisión ("Accuracy")

  La precisión es la métrica más usual en "Deep Learning" y mide directamente la diferencia entre el resultado real y el resultado previsto a través de funciones lineales, cuadráticas o logarítmicas y se utiliza en la mayoría de las funciones de coste.

  IoU (Intersection over Union):

  IoU es una métrica usada principalmente en la detección.

  Pongamos por ejemplo una señal de tráfico que está marcada con una caja verde para indicar su posición real ("ground truth") y con una caja de color naranja su posición predicha.

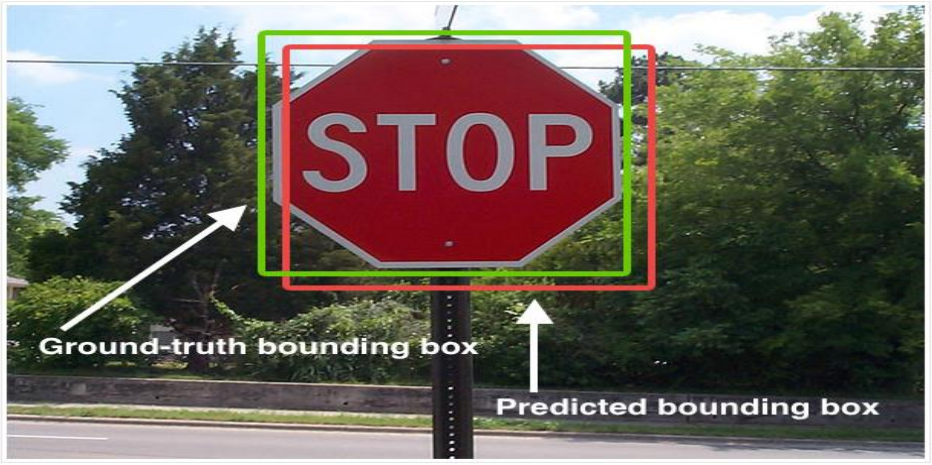


Figura 49 – Ejemplo de detección de objetos con las cajas delimitadoras

Este tipo de ejemplo se puede traducir en la siguiente visualización geométrica.

Figura 50 – Función IoU

Como es imposible que las cajas cuadren completamente, se considera que para un valor de IoU por encima de un umbral prefijado, la posición de la caja delimitadora es correcta. [28]

A continuación, tenemos algunos ejemplos de valores para diferentes situaciones:

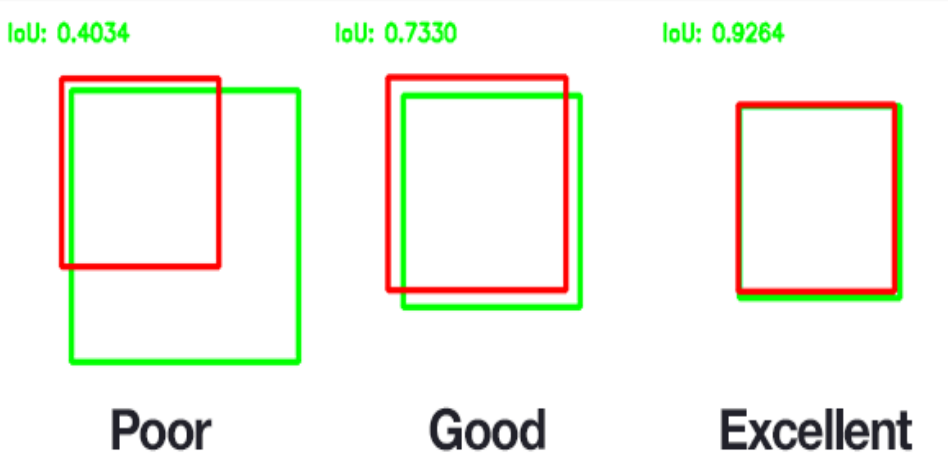


Figura 51 – Valores obtenidos de la función en diferentes casos

- Funciones de coste [29][30]

a) Clasificación

Cross-Entropy:

Es una de las funciones más utilizadas para la clasificación y puede usarse para una clasificación binaria o multiclase y su forma, aunque ya la hemos anticipado en apartados anteriores, es la siguiente:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

b) Regresión

L1:

La función de coste L1 se utiliza para minimizar el error que es la suma de todas las diferencias absolutas entre el valor verdadero ("ground truth") y el valor predicho.

$$L_1 = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

L2:

La función de coste L2 se utiliza para minimizar el error que es la suma de todas las diferencias al cuadrado entre el valor verdadero ("ground truth") y el valor predicho.

$$L_2 = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

Smooth L1:

La función de coste Smooth L1 es de una cierta manera una mezcla de las funciones L1 y L2 y aprovecha las ventajas de cada una de ellas. Cuando la diferencia entre el valor verdadero y el valor predicho está cerca de 0 ésta se asemeja a la función L2 y cuando se aleja se asemeja a la función L1.

$$L_{1smooth}(x) = 0.5x^2 \quad if \ \ |x| < 1$$

$$L_{1smooth}(\mathrm{x}) = |\mathrm{x} - 0{,}5| \quad \text{if } |\mathrm{x}| \geq 1$$

La función L1 es más robusta a los "outliers" y la función L2 es más precisa cuando está cerca de la solución. La función Smooth L1 combina lo mejor de las 2 funciones anteriores.

2.3.3.1 R-CNN, Fast R-CNN y Faster R-CNN

R-CNN

Hace unos años, al explotar algunos de los posibles avances en la visión por computadora a través de CNN, los investigadores desarrollaron R-CNN (siglas de "Region-based Convolutional Neuronal Network") para hacer frente a las tareas de detección, localización y clasificación de objetos. En términos generales, un R-CNN es un tipo especial de CNN capaz de localizar y detectar objetos en imágenes: el resultado generalmente es un conjunto de cajas delimitadoras que coinciden estrechamente con cada uno de los objetos detectados, así como un resultado de clase para cada uno de los objetos detectados. [25][26] [27]

La idea principal se compone de dos pasos. En primer lugar, al utilizar la búsqueda selectiva, identifica un número manejable de regiones candidatas a tener un objeto que se pueda encuadrar por una caja delimitadora. (a estas regiones se las llama regiones de interés" o por

sus siglas en inglés "RoI"). Y luego extrae las características de CNN de cada región de forma independiente para la clasificación.

Cómo funciona R-CNN se puede resumir de la siguiente manera: [25][26][27]

1) Se pre-entrena una red CNN específica en tareas de clasificación de imágenes; por ejemplo, VGG o ResNet entrenadas en el conjunto de datos ImageNet. La tarea de clasificación implica N clases.

2) Se propone regiones de interés independientes de la clase mediante búsqueda selectiva (aproximadamente 2000 regiones por imagen). Esas regiones pueden contener objetos buscados y ser de diferentes tamaños.

3) Las regiones candidatas son deformadas para tener un tamaño fijo según lo requerido por la CNN.

4) Se continúa ajustando la CNN en las regiones de propuestas deformadas para las clases N+1; La clase adicional se refiere al fondo de la imagen (sin objeto de interés). En la etapa de ajuste, se usa una tasa de aprendizaje (“learning rate”) mucho más pequeña para el cálculo de los coeficientes de la red y se ajusta los “batches “usados ya que se utilizan más muestras en los casos donde hay objetos ya que la mayoría de las regiones propuestas sólo tienen fondo.

5) Dada cada región de la imagen, una “Forward Propagation” a través de la CNN genera un vector de características. Este vector de características alimenta un SVM binario [31] entrenado para cada clase de forma independiente.

6) Las muestras positivas son regiones propuestas con umbral de superposición IoU (intersección sobre unión)> = 0.4.

7) Para reducir los errores de localización, se entrena un modelo de regresión con una función de coste L1 para corregir la caja delimitadora utilizando las características de CNN.

La siguiente imagen muestra una arquitectura típica de una R-CNN:

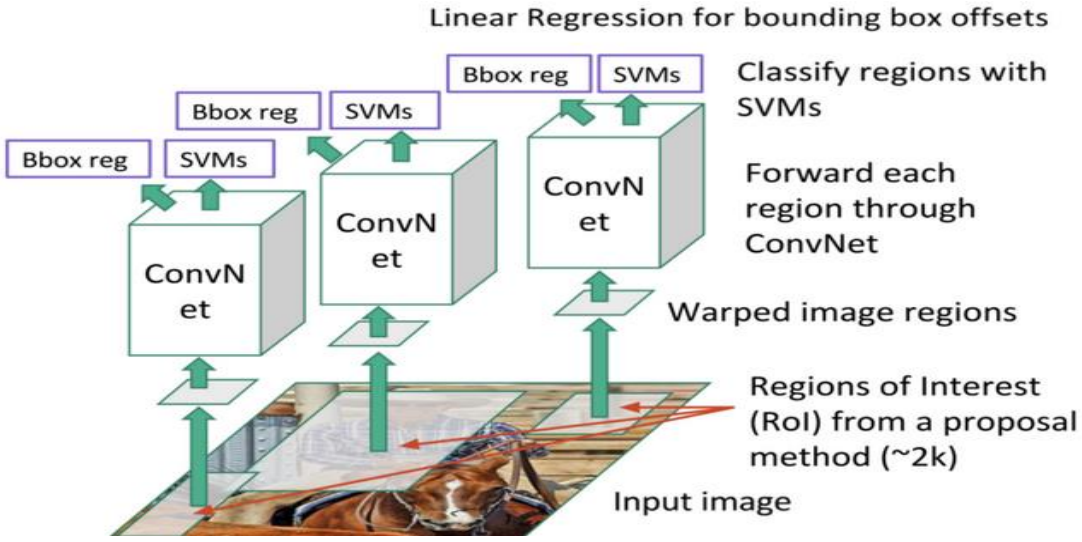


Figura 52 – Arquitectura R-CNN

La misma estructura de funcionamiento de una R-CNN nos muestra que el entreno de un modelo R-CNN es costoso y lento, ya que los siguientes pasos implican mucho trabajo: [24][25][26]

- Ejecución de búsqueda selectiva para proponer 2000 candidatos de región para cada imagen;
- Generación del vector de características de CNN para cada región de imagen (J imágenes * 2000).
- Todo el proceso involucra tres modelos por separado sin mucha computación compartida: la CNN para la clasificación de imágenes y la extracción de características; el clasificador SVM superior para identificar objetos de destino; y el modelo de regresión para ajustar las cajas delimitadoras de la región.

Estos inconvenientes hicieron buscar una mejora de la R-CNN que desembocó en la Fast R-CNN

Fast R-CNN

Para hacer que la R-CNN fuera más rápida, se mejoró el procedimiento de entreno unificando los tres modelos independientes de la R-CNN en un marco de entrenamiento conjunto y aumentando los resultados del cálculo compartido y se le llamó Fast R-CNN. En lugar de extraer vectores de características CNN de forma independiente para cada propuesta de región, este modelo los agrega en la "Forward Propagation" de la CNN sobre la imagen completa y las propuestas de región comparten esta matriz de características. Luego, se ramifica la misma matriz de características para usarla en el clasificador de objetos y en el cálculo de regresión de la caja delimitadora. [25][26][27]

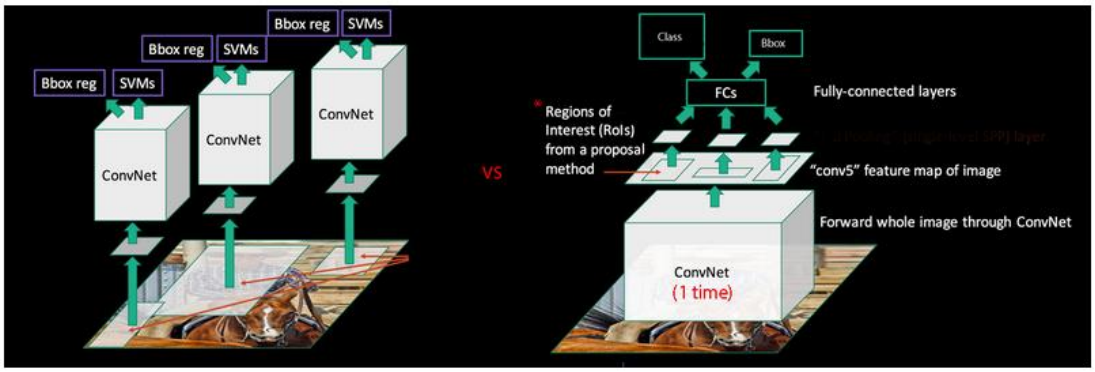


Figura 53 – Diferencia entre el modelo R-CNN y el modelo Fast R-CNN

En la figura 54, se muestra la arquitectura más detallada del modelo Fast R-CNN.

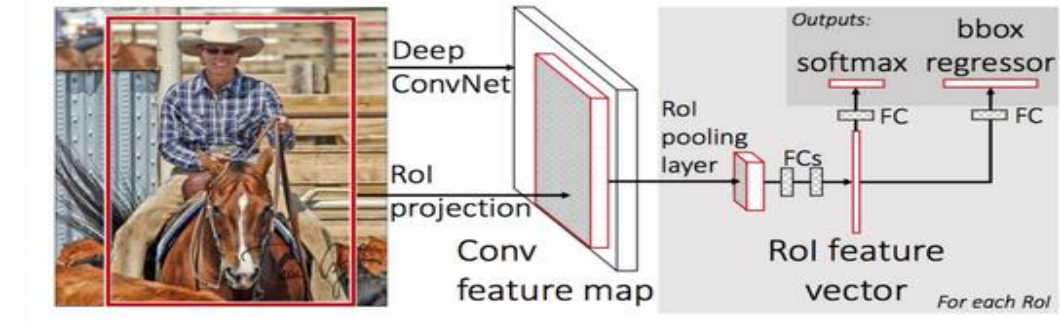


Figura 54 – Arquitectura Fast R-CNN

En la figura aparece lo que se denomina ROI Pooling Layer que es una de las aportaciones principales de este modelo.

Es un tipo de “Max-pooling” para convertir entidades en la región proyectada de la imagen de cualquier tamaño, H x W, en una pequeña ventana fija, h x w. La región de entrada se divide en cuadrículas h x w, y a continuación se aplica el “Max-pooling” en cada cuadrícula.

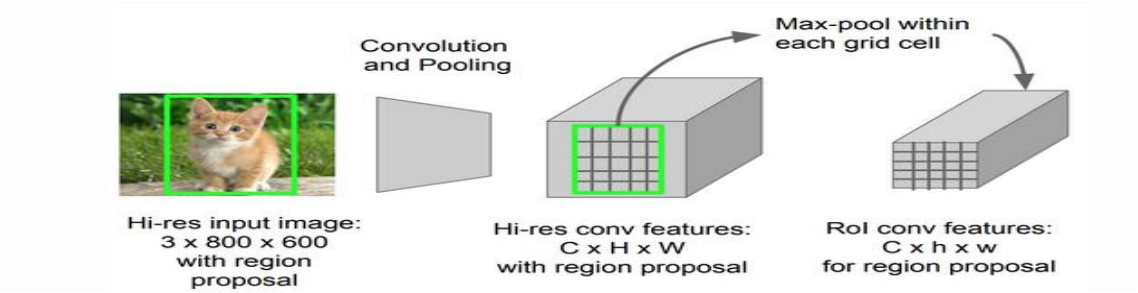


Figura 55 - Funcionamiento de la ROI Pooling Layer

Para cada región de interés, toma una sección del mapa de entidades de entrada que le corresponde y lo escala a un tamaño predefinido. De esta manera, divide la propuesta de región en secciones de igual tamaño (cuyo número es el mismo que la dimensión de la salida), encuentra el valor más grande en cada sección y copia los valores máximos de cada cuadrícula en el buffer de salida. [25][26][27]

El algoritmo de funcionamiento del modelo Fast R-CNN sería el siguiente (muchos pasos son los mismos que en R-CNN): [25][26][27]

1)En primer lugar, se pre-entrena una red neuronal convolucional en tareas de clasificación de imágenes (VGG, RESNET u otras)

2) Se propone regiones por búsqueda selectiva de objetos potenciales (aproximadamente 2000 candidatas por imagen).

3) Se modifca la red CNN pre-entrenada:
Se reemplaza la última capa de “max-pooling” de la CNN pre-entrenada por una capa de “Max-pooling” RoI. La capa de “Max-pooling” RoI genera vectores de características de longitud fija de propuestas de regiones en la imagen. Se reemplaza la última capa FC y la última capa de softmax de K clases por una capa FC y softmax de K + 1 clases.

4) Finalmente, el modelo se bifurca en dos capas de salida:
- Un estimador softmax de K + 1 clases (igual que en el caso de R-CNN, la clase añadida corresponde al fondo de la imagen), generando una distribución de probabilidad discreta por cada salida de la capa RoI.
- Un modelo de regresión de caja delimitadora que predice las compensaciones con respecto al RoI original para cada una de las K clases.

La función de coste del modelo es una combinación de la función de coste de clasificación y de la función de coste de regresión (la función de clasificación es una función “Cross-Entropy” y la función de regresión es una función Smooth L1). Cada una de estas funciones tiene un coeficiente multiplicador para balancear el peso que puedan tener en el coste global

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$$

Si no existe objeto en la región propuesta, la función de coste de regresión es nula.

Aun mejorando enormemente el tiempo de respuesta de la red R-CNN, al generarse las regiones fuera de la línea principal de la CNN se perdía bastante tiempo de ejecución. Lo que llevó a la aparición del modelo Faster R-CNN.

Faster R-CNN

Una solución de aceleración intuitiva era integrar el algoritmo de propuesta de región en el modelo de CNN. Esto fue lo que se acabó haciendo en el modelo Faster R-CNN:

Se construyó un modelo único y unificado compuesto de RPN (red de propuesta de región) y rápido R-CNN con capas de funciones convolucionales compartidas. [25][26][27]

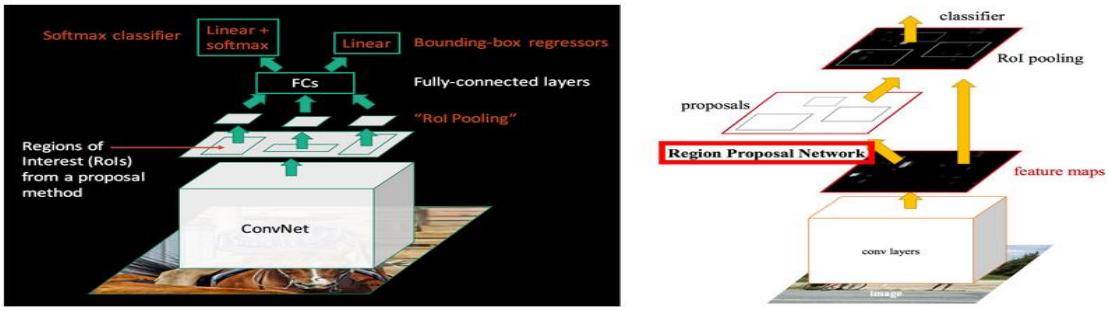


Figura 56 – Comparación de los modelos Fast R-CNN y Faster R-CNN

El algoritmo de funcionamiento viene a ser el siguiente: [25][26][27]

1)Se pre-entrena una red CNN en tareas de clasificación de imágenes (VGG, RESNET u otras)

2) Se ajusta la RPN ("Region Proposal Network") de extremo a extremo para la tarea de propuesta de región, que se inicializa mediante el clasificador de imagen de pre-entreno. Las muestras positivas tienen una IoU (intersection-over-union)> 0.7, mientras que las muestras negativas tienen IoU <0.3.
Se desliza una pequeña ventana n x n sobre el mapa de características de las convoluciones de toda la imagen (ver Figura 58)
En el centro de cada ventana deslizante, predecimos múltiples regiones de varias escalas y proporciones simultáneamente. Un anclaje es una combinación de (centro de ventana deslizante, escala, ratio). Por ejemplo, 3 escalas x 3 relaciones implican 9 anclajes en cada posición deslizante.

3) Se entrena un modelo rápido de detección de objetos R-CNN usando las propuestas generadas por la RPN actual.

4) Se usa la parte común al modelo Fast R-CNN para inicializar el entrenamiento de la RPN. Mientras mantiene las capas convolucionales compartidas, solo se afinan las capas específicas de RPN. (En esta etapa, RPN y la red de detección tienen capas convolucionales compartidas)

5) Finalmente, se afina las capas únicas del modelo común al Fast R-CNN

6) El paso 4-5 se puede repetir para entrenar la RPN y las capas únicas del modelo comunes a Fast R-CNN alternativamente si es necesario.

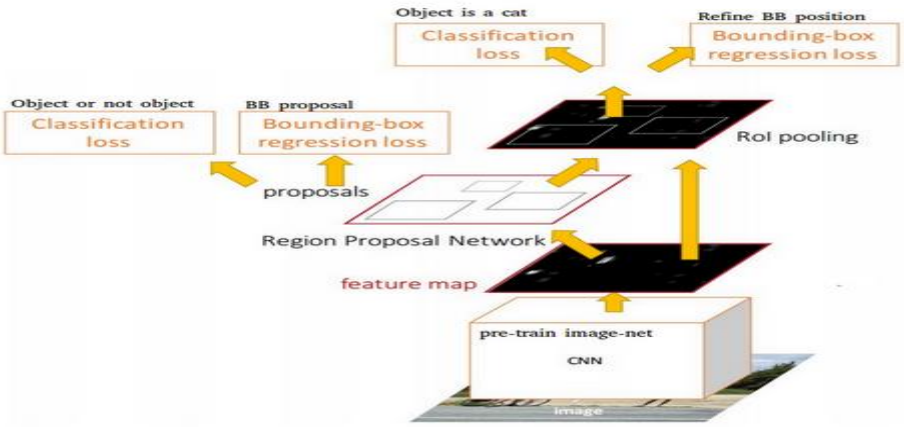


Figura 57- Arquitectura del modelo Faster R-CNN

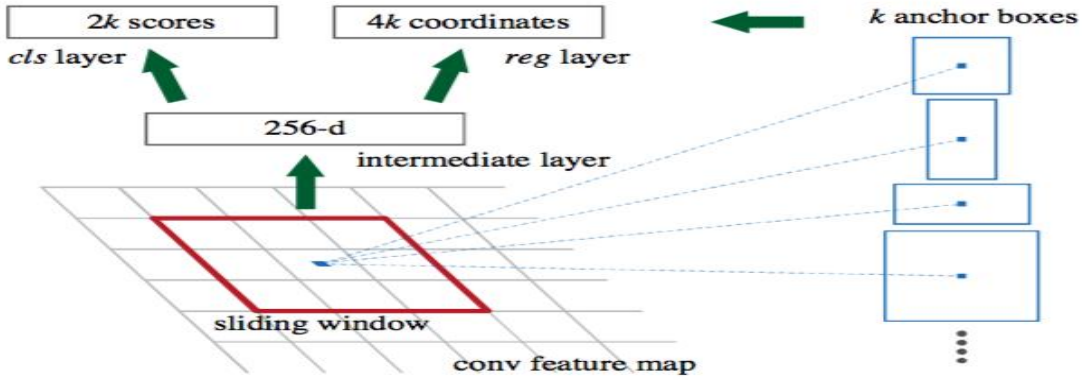


Figura 58 – Generación de los anclajes en el modelo Faster R-CNN

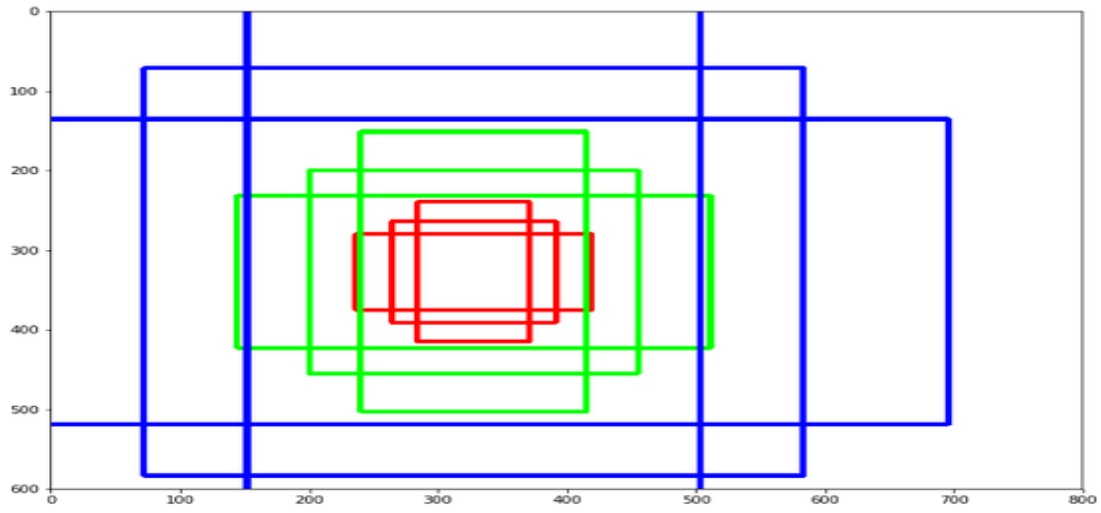


Figura 59- Anclajes de 320 x 320 generados en una imagen de 600 x 800 píxeles

La función de coste para el modelo faster R-CNN es muy parecida a la del modelo Fast R-CNN

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$$

Para finalizar el apartado común al modelo R-CNN y sus mejoras, se añade alineados los 3 modelos estudiados:

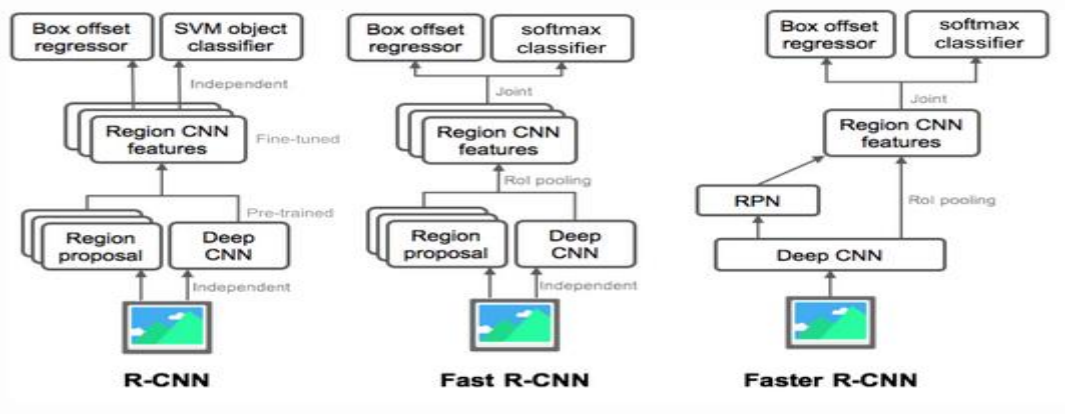


Figura 60- Comparativa de los 3 modelos R-CNN

Existe una función importante que no aparece explícitamente en ninguna de las 3 arquitecturas y que sirve para reducir el número de cajas delimitadoras asociadas a la detección de un objeto. Esta función se llama NMS (siglas de "Non Maximum Suppression"). A partir de la métrica IoU, se escoge el valor de la caja delimitadora que tenga el valor IoU más elevado asociado a un objeto y se eliminan las otras. [25][26][27]

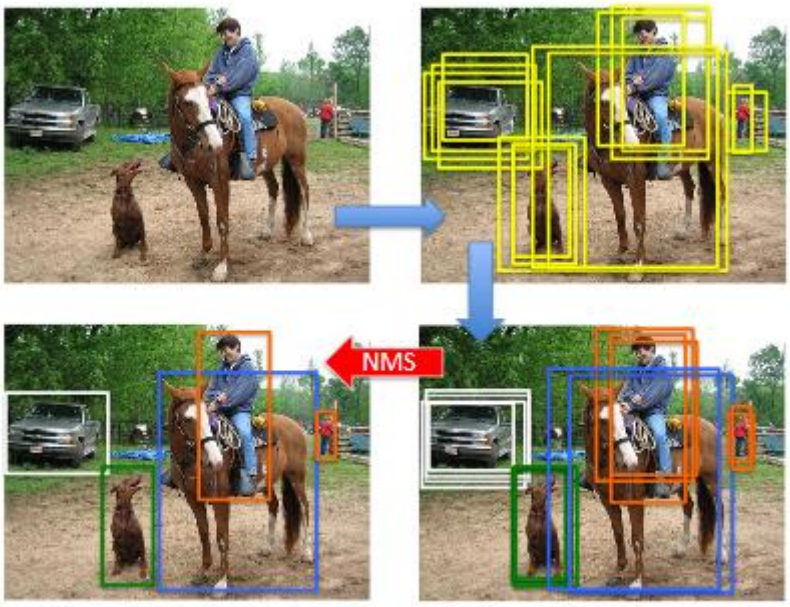


Figura 61 – Resultados de la aplicación de la función NMS

2.3.3.2 YOLO

YOLO (siglas de "You Only Look Once") es un modelo de detección que trata la tarea de reconocimiento de objetos como un problema de regresión unificado, diferente de los modelos de la familia R-CNN que aprenden a resolver una tarea de clasificación. Mientras tanto, YOLO ve toda la imagen durante el entrenamiento y, por lo tanto, tiene un mejor rendimiento en el reconocimiento del fondo con el conocimiento del contexto completo.

El algoritmo funciona de la manera siguiente: [25][26][27]

Se pre-entrena una red CNN en tareas de clasificación de imágenes.

Se divide una imagen en cuadrículas S x S. Cada cuadrícula es responsable de identificar el objeto (si lo hay) con su centro ubicado en esta celda. Cada celda predice la ubicación de B cajas delimitadoras y una puntuación de confianza, y una probabilidad de clase de objeto condicionada a la existencia de un objeto en la caja delimitadora.
Una caja delimitadora está definida por una tupla de (centro x, centro y, ancho, alto) - (x, y, w, h)
x e y se normalizan para ser los desplazamientos de una ubicación de una cuadrícula; w y h se normalizan por el ancho y alto de la imagen, y por lo tanto entre (0, 1).
La puntuación de confianza es la probabilidad de contener un objeto x IoU (predicción, realidad).
Si la cuadrícula contiene un objeto, se predice una probabilidad de que este objeto pertenezca a una clase Ci, i = 1,2, ..., K: probabilidad (el objeto pertenece a la clase Ci | contiene un objeto). En esta etapa, el modelo solo predice un conjunto de probabilidades de clase por cuadrícula, independientemente del número B de cajas delimitadoras.
En total, una imagen contiene S x S x B cajas delimitadoras, cada caja corresponde a 4 predicciones de ubicación, 1 puntuación de confianza y K probabilidades condicionales para la clasificación de objetos. Los valores de predicción totales para una imagen son S x S x (5B + K).

La capa final de la CNN pre-entrenada se modifica para generar un tensor de predicción de tamaño S x S x (5B + K).

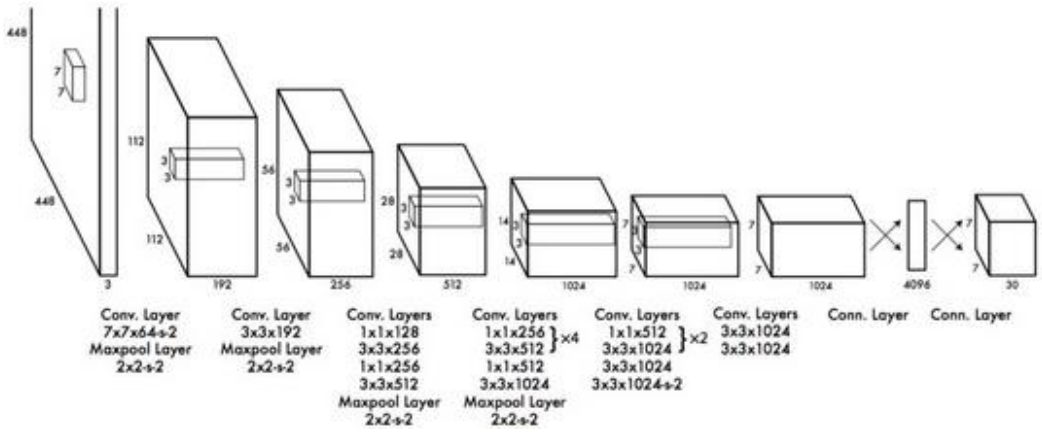


Figura 62 - Arquitectura del modelo YOLO

La red tiene 24 capas convolucionales seguidas por 2 capas FC. La mayoría de las capas convolucionales están pre-entrenadas utilizando el conjunto de datos ImageNet para la clasificación. Cuatro capas convolucionales seguidas por dos capas FC se agregan a la red anterior y se vuelve a entrenar completamente con los datasets específicos PASCAL VOC.

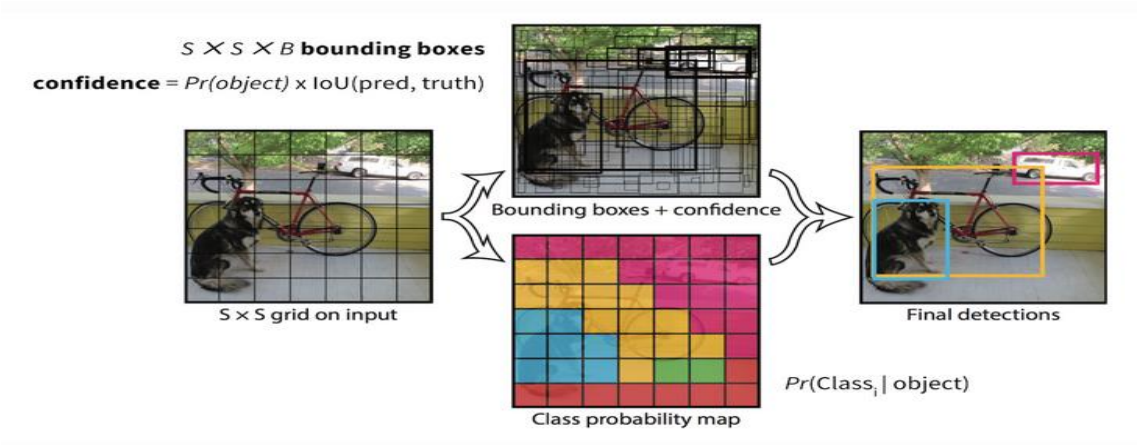


Figura 63 – Funcionamiento del modelo YOLO

Con los modelos anteriores, las cajas delimitadoras previstas a menudo contenían un objeto. Sin embargo, el modelo YOLO predice una gran cantidad de cajas delimitadoras. Por lo tanto, hay muchas cajas delimitadoras sin ningún objeto. El método de supresión de los máximos (NMS) se aplica al final de la red como se ha indicado en los modelos R-CNN

YOLO está entrenado para minimizar la suma de los errores al cuadrado, con parámetros de escala para controlar cuánto queremos aumentar la pérdida de las predicciones de coordenadas de la caja delimitadora (λcoord) y cuánto queremos disminuir la pérdida de las predicciones de confianza para las cajas que no contienen objetos (λnoobj). [25][26][27]

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} (C_{ij} - \hat{C}_{ij})^2$$
$$+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{noobj} (C_{ij} - \hat{C}_{ij})^2 + \sum_{i=0}^{S^2} l_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

l_i^{obj} *whether the cell i contains an object*

l_{ij}^{obj} $j-th$ *bounding box predictor of the cell is "responsible" for that prediction*

C_{ij} *confidence score of the* $j-th$ *box in the cell* $i, probability\ (containing\ an\ object) * IoU(pred, truth)$

$\widehat{C_{ij}}$ *predicted confidence score*

p_i(c) conditional probability of whether cell i contains an object of class c

$\hat{p}_i$(c) predicted conditional probability of whether cell i contains an object of class c

Figura 64 – Función de coste del modelo YOLO

Sin embargo, una de las limitaciones de YOLO es que solo predice 1 tipo de clase en una cuadrícula, por lo tanto, tiene problemas con objetos muy pequeños.

2.3.3.3 SSD

De manera similar al modelo YOLO, se desarrolló posteriormente una red de detección llamada SSD ("Single Shot MultiBox Detector") para predecir de una sola vez las cajas delimitadoras y las probabilidades de cada clase con una arquitectura de CNN de extremo a extremo. [26][27][32]

El nombre Single Shot o disparo único significa que las tareas de localización y clasificación de objetos se realizan en un único paso hacia adelante de la red.
El nombre MultiBox le vienen de una técnica el cálculo de la regresión de la caja delimitadora.
El nombre Detector es debido a que la red detecta los objetos y también los clasifica.

Como puede ver en la figura 67, la arquitectura más corriente de SSD se basa en la arquitectura VGG-16 (red pre-entrenada que permite la clasificación de objetos en un gran número de clases), pero descarta sus capas finales FC. La razón por la que VGG-16 se usó como red base se debe a su gran desempeño en tareas de clasificación de imágenes de alta calidad y su popularidad para problemas donde el aprendizaje por transferencia ayuda a mejorar los resultados. En el caso de la figura 67, se agregó un conjunto de capas "convolucionales" auxiliares desde la capa 6 de la red pre-entrenada, lo que permitió extraer características a múltiples escalas y disminuir progresivamente el tamaño de la entrada a cada capa subsiguiente. [26][27][32]

El modelo SSD en lo que concierne la búsqueda de las cajas delimitadoras es una modificación del modelo "Multi Box".

En el caso "MultiBox", los investigadores crearon lo que llamamos "cajas delimitadoras a priori" o "priors" (son lo que se denominó anclajes en la terminología de Faster-R-CNN), que son cajas de límite de tamaño fijo pre-calculadas que coinciden estrechamente con la distribución de las cajas delimitadoras reales ("ground truth"). De hecho, esos "priors" se seleccionan de tal manera que su IoU es mayor que 0.5. Una IoU de 0,5 todavía no es lo suficientemente buena, pero proporciona un punto de partida sólido para el algoritmo de regresión del cuadro delimitador (es una estrategia mucho mejor que comenzar las predicciones con coordenadas aleatorias) Por lo tanto, MultiBox comienza con los "priors" como predicciones e intenta acercarse más cerca de las cajas delimitadoras reales.

Si consideramos la estructura del modelo "Multibox" de la figura siguiente:

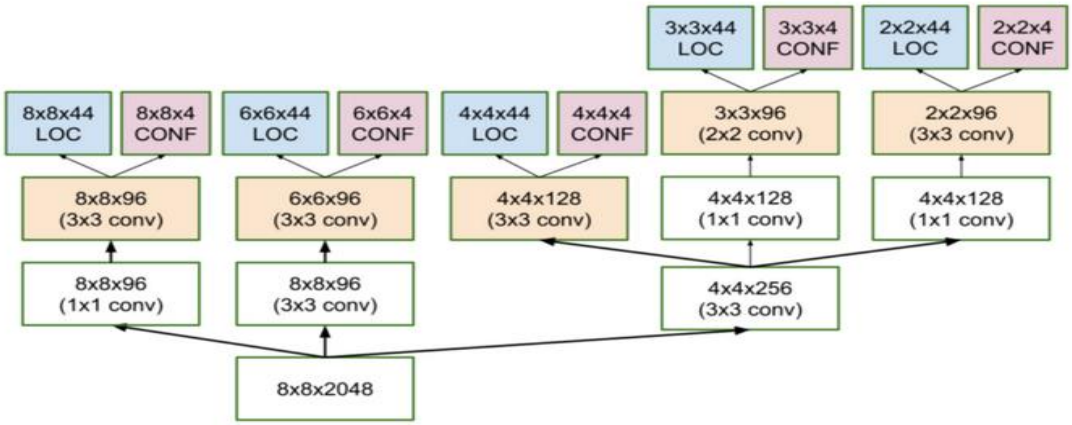


Figura 65- Arquitectura Multibox

La arquitectura "Multibox" resultante (ver figura 65) contiene 11 "priors" por cuadrícula de características (8x8, 6x6, 4x4, 3x3, 2x2) y solo una en el mapa de funciones 1x1, lo que da un

total de 1420 “priors” por imagen. lo que permite una cobertura robusta de las imágenes de entrada en múltiples escalas, para detectar objetos de varios tamaños. [32]

En SSD con relación a Multibox, se agregaron varios ajustes para que esta red fuera aún más eficiente en la localización y clasificación de objetos.

A diferencia de “MultiBox”, cada cuadrícula del mapa de características está asociada a un conjunto de cajas delimitadoras predeterminadas de diferentes dimensiones y relaciones de aspecto. Estos “priors” se eligen de forma manual, mientras que en MultiBox, se eligieron porque su IoU era superior a 0,5. Esto en teoría debería permitir que las SSD se generalicen para cualquier tipo de entrada, sin requerir una fase de pre-entreno para la generación de los “priors”. Por ejemplo, suponiendo que hemos configurado 2 puntos diagonalmente opuestos (x1, y1) y (x2, y2) para cada b cajas de delimitadoras predeterminados por cuadrícula de características, y C clases para clasificar, en un mapa de características dado de tamaño f = m * n, SSD calcularía f * b * (4 + C) valores para este mapa de características. [26][27][32]

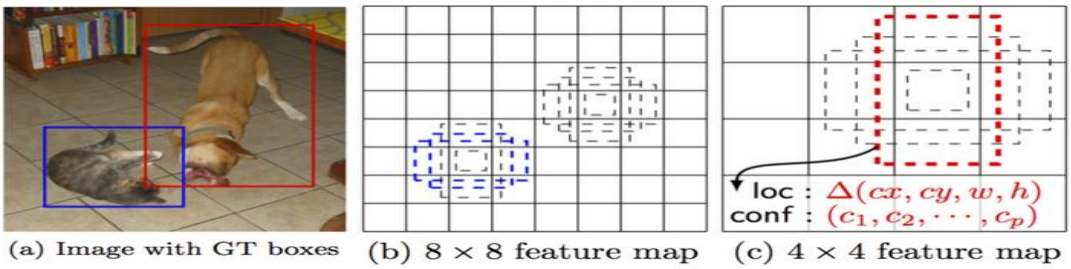


Figura 66- Elección de las cajas delimitadoras

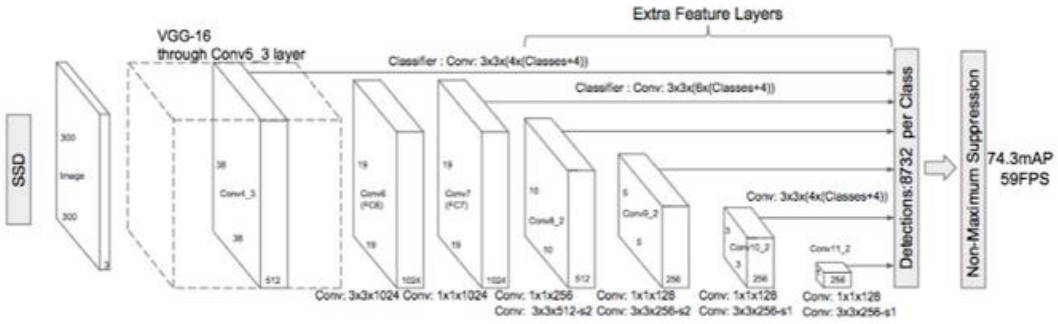


Figura 67 – Arquitectura de una red SSD (Multi Box Single Shot Detector)

Durante el entreno de la red SSD se utilizan 2 técnicas de las cuales ya hemos comentado una de ellas.

La primera que es nueva es la que se llama “Hard Negative Mining”. Durante el entrenamiento, como la mayoría de las cajas delimitadoras tendrán un IoU bajo y, por lo tanto, se interpretarán como ejemplos de valor negativo, podemos terminar con una cantidad desproporcionada de ejemplos negativos en nuestro conjunto de entreno. Por lo tanto, en lugar de utilizar todas las predicciones negativas, se recomienda mantener una proporción de ejemplos negativos a positivos de alrededor de 3: 1. La razón por la que necesita mantener muestras negativas es porque la red también necesita aprender y se le debe decir explícitamente qué constituye una detección incorrecta. [26][27][32]

La segunda técnica es la llamada NMS (“Non Maximum Suppression”) que permite de eliminar toda una serie de cajas delimitadoras generadas por el modelo. Las que tienen un IoU inferior a un valor de determinado no son tenidas en cuenta. [26][27][32]

La función de coste del modelo SSD es parecido al de Fast R-CNN.

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$$

donde el primer término es una función de entropía y el segundo una función Smooth L1. En algunos casos se utilizan coeficientes para balancear la función de clasificación y la función de regresión.

2.3.3.4 Comparativa de las diferentes redes de detección

De los 3 grupos de modelos el primero (R-CNN, Fast R-CNN y Faster R-CNN) es el que provee la mejor resolución, aunque es más lento en el procesado. Faster R-CNN es de las 3 variantes la que da un mejor compromiso entre rapidez y precisión.

YOLO es el modelo más simple y por lo tanto más rápido en obtener los resultados, aunque tiene menos precisión que las demás modelos de redes utilizados para la detección. Puede procesar imágenes a más de 45ms por lo que puede ser utilizado para la detección en tiempo real en videos. No da buenos resultados para objetos con formas irregulares y para objetos pequeños.

SSD es modelo intermedio entre Faster R-CNN y YOLO. Para objetos grandes, SSD funciona bastante bien, pero se comporta peor en comparación con Faster R-CNN. Para objetos pequeños, los resultados no son especialmente buenos.

Figura 68 – Comparación de los modelos de detección

Todos estos modelos necesitan subredes pre-entrenadas que son las que permiten muy buenas clasificaciones por clases una vez entrenadas. Éstas se entrenan con un gran número de fotografías de diferentes conjuntos de datos (por ejemplo, ImageNet) y una vez acabado el entreno se puede aprovechar sus coeficientes y sólo se debe entrenar las redes posteriores para poder clasificar nuevas clases con características comunes a algunas de las clases detectadas por la red pre-entrenada. Este proceso es el que se llama transferencia de conocimiento o “Transfer Learning”.

En el caso de la detección de asteroides, la red pre-entrenada a priori no es de mucha utilidad ya que tenemos 2 clases por separar: el ruido de fondo y los asteroides. El ruido de fondo se puede separar con CNNs relativamente simples pero los asteroides y las estrellas a priori no se pueden distinguir por ninguna característica física y menos aun cuando sólo tenemos grises. La intensidad y la forma no son indicativos ya que cualquiera de los dos es intercambiable en esas 2 características.

Además, la mayoría de los modelos necesitan, vista su configuración (en la CCN global hay una o varias FCs), un valor fijo para las dimensiones de las fotografías de entrada y acostumbra a ser un valor relativamente bajo para las dimensiones de las fotografías de nuestro problema. Lo que implicaría redimensionar las imágenes al tamaño recomendado de cada una de estas redes. Como el factor de reducción es bastante grande y como tenemos objetos pequeños es muy posible que no se pueda efectuar la detección.

2.3.4 Segmentación de objetos

Después de la clasificación, la clasificación y la localización y finalmente la detección, le sigue la segmentación que es el paso siguiente dentro del reconocimiento de las imágenes. Ya no sólo vamos a buscar la clasificación de un objeto asociándole una posición y una caja delimitadora, ahora vamos a asociar a cada píxel de la imagen una etiqueta o clase. Es decir que todos los píxeles con características similares van a tener una misma etiqueta y van a estar diferenciados de los píxeles de las demás clases. Esto se va a llamar más específicamente segmentación semántica. Existe otro tipo de segmentación, que no detallaremos en este trabajo, en el que dentro de cada clase vamos a separar todos los objetos que están dentro de esa clase y la llamaremos segmentación de instancias.

En la figura 54, se ve la imagen original con diferentes personas detectadas y encuadradas por cajas delimitadoras, seguida por una segmentación semántica y finalmente una segmentación de instancias.

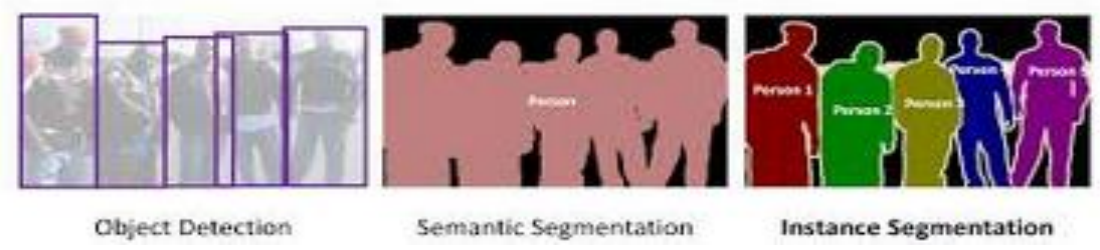


Figura 69 – Etapas dentro del reconocimiento de imágenes

Una arquitectura de segmentación semántica general puede considerarse en general como una red de codificación (“encoder”) seguida de una red de decodificación(“decoder”):

- La codificación la suele cumplir una CNN de clasificación (podría ser en bastantes casos una red pre-entrenada como la VGG o la ResNet). En la mayoría de las redes de segmentación, el bloque de codificación es muy parecido. [33]
- La decodificación proyecta semánticamente las características discriminatorias (resolución más baja) aprendidas por el decodificador en el espacio de píxeles (resolución más alta) para obtener una clasificación densa. [33]

A diferencia de la clasificación donde el resultado final de la CNN es lo único importante, la segmentación semántica no solo requiere discriminación a nivel de píxel sino también un mecanismo para proyectar las características discriminatorias aprendidas en diferentes etapas del codificador en el espacio de píxeles.

Aunque existen unas cuantas redes que cubren la segmentación semántica, abordaremos 3 de ellas. La primera de ellas será la red FCN (siglas de "Full Convolutional Network"), luego la red SEGNET que corrige ciertos defectos de la red FCN y finalmente una red más avanzada llamada U-NET y utilizada originalmente para la segmentación de imágenes de medicina.

Antes de proceder a detallarlas, explicaremos ciertas operaciones que se hacen en la segmentación, principalmente en el apartado de decodificación y en el enlace de la codificación con la decodificación.

- Conversión de una FC ("Full Connected Layer") en convolución
- Convolución Transpuesta
- Interpolación

a) Conversión de una FC en convolución
   Como se puede ver en la figura 70, se puede reemplazar la capa FC de 4096 nodos con una capa "convolucional" con un tamaño de filtro de 7x7, un "padding" de cero, un "stride" de 1 y una profundidad de salida de 4906. Se puede hacer un cálculo rápido para ver que la salida será simplemente 1x1x4096, equivalente a la salida de la capa FC.

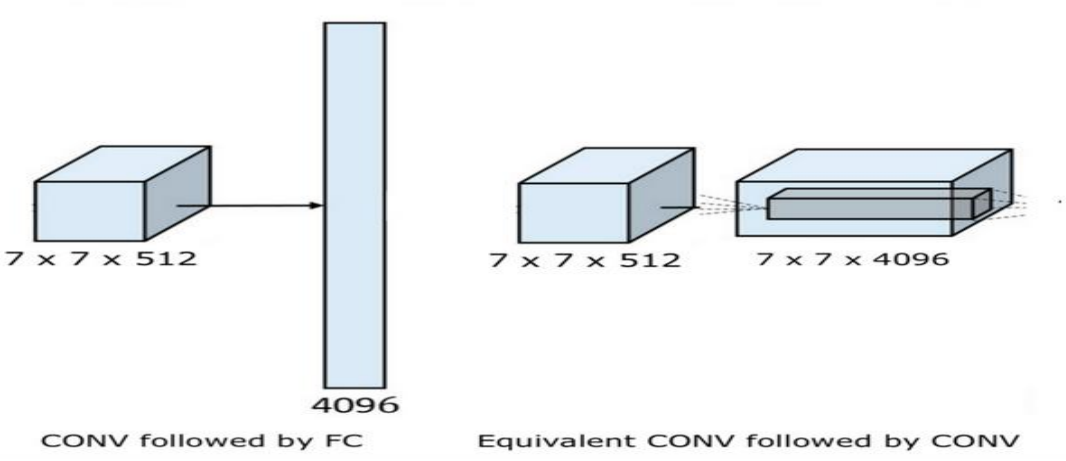


Figura 70- Reemplazo de la capa FC por una capa "convolucional" equivalente

b) Convolución transpuesta [34]

La operación de convolución transpuesta forma la misma conectividad que la convolución normal, pero en la dirección inversa de la convolución.

Podemos usarla para realizar un muestreo en la decodificación. Además, los valores en la convolución transpuesta se pueden obtener por aprendizaje. Por lo tanto, no necesitamos un método de interpolación predefinido.

En las figuras siguientes, vemos cómo se puede buscar un modelo equivalente de una convolución y obtener de esta manera una convolución traspuesta.

Si partimos, por ejemplo, de una matriz de entrada 4x4 y le aplicamos un filtro 3x3.

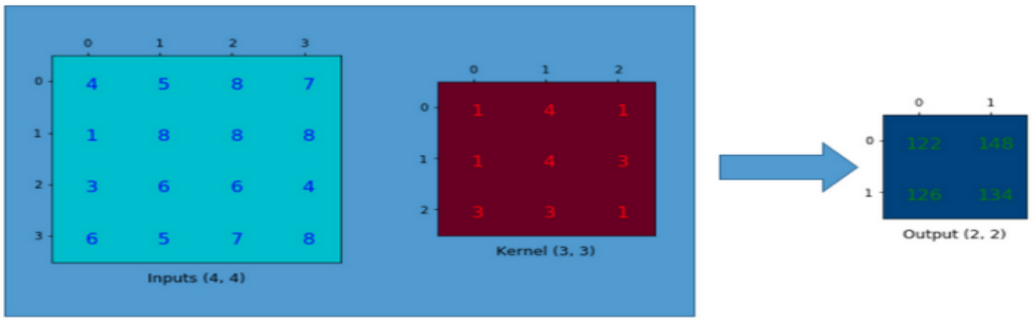


Figura 71 – Convolución directa

Si examinamos el procedimiento de convolución, podemos reescribir la matriz del filtro en el modelo siguiente que luego utilizaremos para hacer un producto matricial con la matriz de entrada.

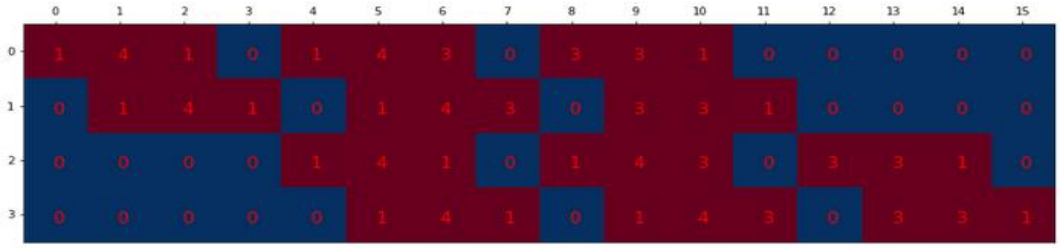


Figura 72 – Equivalencia del filtro (3,3) anterior

Transformamos la matriz de entrada (4x4) en un vector unidimensional.

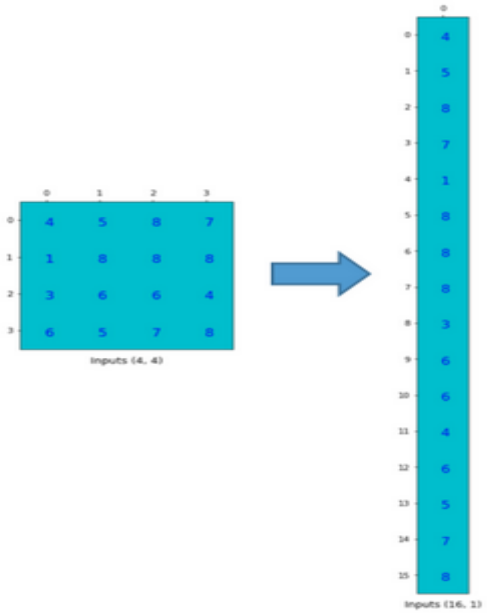


Figura 73 – Redimensionamiento de la matriz 4x4 de entrada en un vector unidimensional

Efectuamos el producto matricial entre la matriz equivalente del filtro y el vector unidimensional de los valores de entrada.

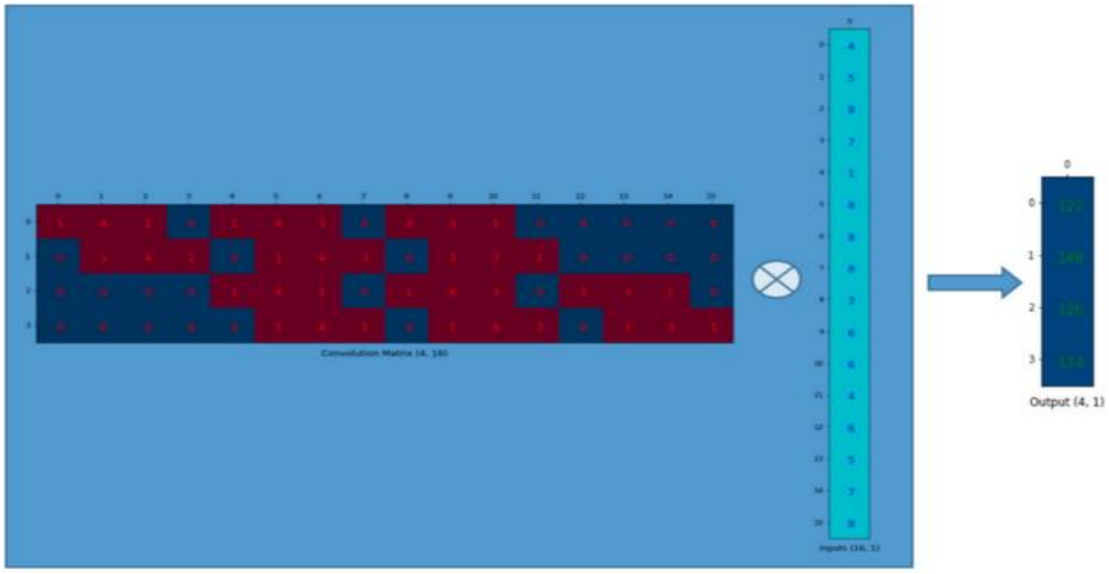


Figura 74 – Cálculo la convolución con el nuevo modelo

Obtenemos un vector unidimensional 4x1, que, transformado en una matriz 2x2, da el mismo resultado que la convolución inicial.

Ahora, supongamos que tengamos una matriz 2x2, de entrada:

2	1
4	4

Si quisiéramos aplicar el mismo filtro 3x3 que en la convolución anterior, pero ahora aplicando la convolución transpuesta. Haremos los mismos pasos que hemos seguido anteriormente. Transformaremos la entrada en un vector 4x1 y buscaremos la equivalencia del filtro 3x3 en su versión transpuesta y obtendremos el resultado de la figura siguiente:

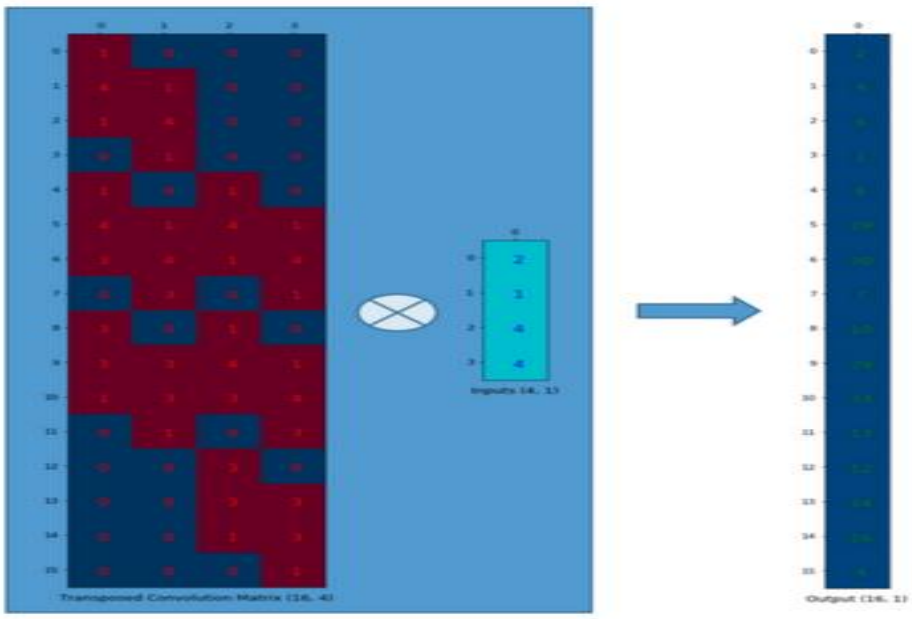


Figura 75 – Convolución transpuesta

Calculamos el producto matricial y obtenemos un vector 16x1, que transformamos finalmente en una matriz 4x4.

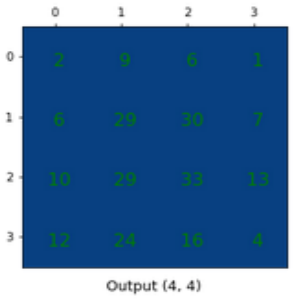


Figura 76 – Resultado obtenido de una convolución transpuesta

Aunque se denomina convolución transpuesta, no significa que tomemos una matriz de convolución existente y usemos la versión transpuesta. El punto principal es que la asociación entre la entrada y la salida se maneja de manera inversa en comparación con una matriz de convolución estándar (asociación de uno a muchos en lugar de muchos a uno).

Como tal, la convolución transpuesta no es una convolución. Pero podemos emular la convolución transpuesta utilizando una convolución. Muestreamos la entrada agregando ceros entre los valores en la matriz de entrada de forma que la convolución directa produce el mismo efecto que la convolución transpuesta.

c) Interpolación [35] [36]

En la operación de “upsampling”, se utiliza el método de la interpolación. En la segmentación, se utilizan principalmente varios métodos de interpolación de los cuales nos interesaremos por la del “nearest neighbour” y por la de interpolación bilineal.

En la figura siguiente, se muestra la interpolación del vecino más cercano o “nearest neighbour”.

El algoritmo de “remuestreo” de los vecinos más cercanos es un método de interpolación que, como la convolución, realiza una operación matemática en cada píxel (y sus vecinos) dentro de la imagen para agrandar el tamaño de la imagen. En el caso más simple, hacemos tres copias de cada píxel y las colocamos cerca (Figura 77). Casos más complejos implican combinaciones ponderadas de píxeles para generar colores de degradado entre vecinos.

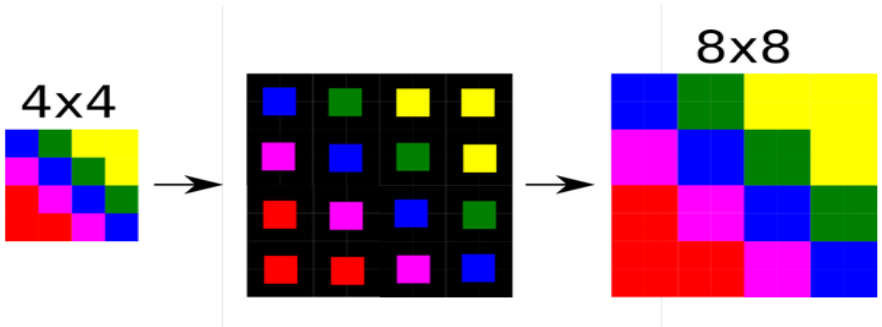


Figura 77 – Interpolación “Nearest Neighbour”

Las áreas negras se llenan con copias del píxel central. Esto le da el efecto de "líneas dentadas" o "pixeladas". Algoritmos más complejos usan combinaciones ponderadas de los

píxeles circundantes para rellenar las áreas negras y proporcionar una imagen más amplia con una apariencia más lisa y realista.

La interpolación bilineal es más compleja a nivel de procesamiento, pero da resultados bastante más precisos (figura 78)

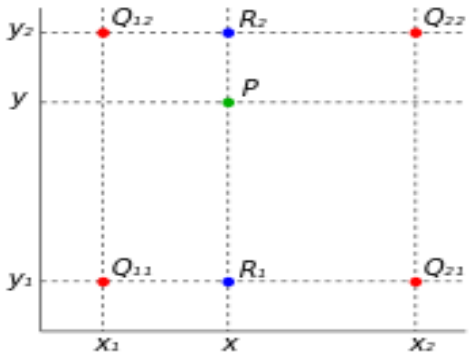


Figura 78- Interpolación bilineal

Supongamos que queremos encontrar el valor de la función bilineal desconocida f en el punto (x, y). Se supone que conocemos el valor de f en los cuatro puntos Q11 = (x1, y1), Q12 = (x1, y2), Q21 = (x2, y1) y Q22 = (x2, y2).

Primero hacemos interpolación lineal en la dirección x.

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Y posteriormente en la dirección y:

$$\begin{aligned} f(x, y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \big(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1) \big) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}. \end{aligned}$$

2.3.4.1 FCN

Las redes totalmente “convolucionales” o FCN (“Fully Connected Network”) deben su nombre a su arquitectura, que está construida solo a partir de capas conectadas localmente, como convolución, “pooling” y “upsampling”. Hay que tener en cuenta que no se utiliza una capa densa (FC) en este tipo de arquitectura. Esto reduce la cantidad de parámetros y el tiempo de cálculo. Además, la red puede funcionar independientemente del tamaño de la imagen original, sin requerir ninguna cantidad fija de unidades en ninguna etapa, lo que significa que todas las conexiones son locales. Para obtener un mapa de segmentación (salida), las redes de segmentación generalmente tienen 2 partes: [37]

- Ruta de reducción de la resolución(“downsampling”): captura la información semántica /

contextual

- Ruta de aumento de la resolución (“upsampling”): recupera la información espacial

La ruta de “downsampling” se usa para extraer e interpretar el contexto, mientras que la ruta de “upsampling” se usa para permitir una localización precisa. Además, para recuperar completamente la información espacial de grano fino perdida en las capas de “max-pooling” o de reducción de muestreo, a menudo usamos conexiones de “salto”.

Una conexión de “salto” es una conexión que salta al menos una capa. Aquí, a menudo se usa para transferir información local al concatenar o sumar mapas de características de la ruta de baja resolución con mapas de características de la ruta de alta resolución. La combinación de características de varios niveles de resolución ayuda a combinar información de contexto con información espacial. [37]

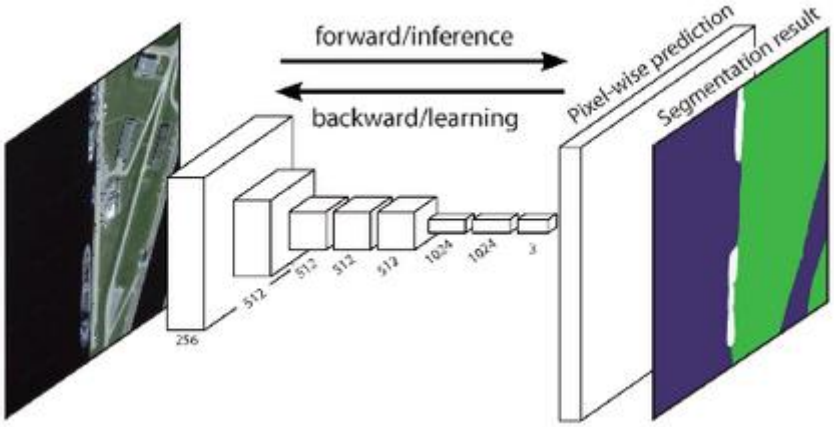


Figura 79 – Ejemplo de una arquitectura de FCN

Hay variantes de la arquitectura FCN, que difieren principalmente en la precisión espacial de su salida. Por ejemplo, la siguiente figura muestran las variantes FCN-32, FCN-16 y FCN-8. En las figuras, las capas “convolucionales” se representan como líneas verticales entre las capas de “pooling”, que muestran explícitamente el tamaño relativo de los mapas de características. [37]

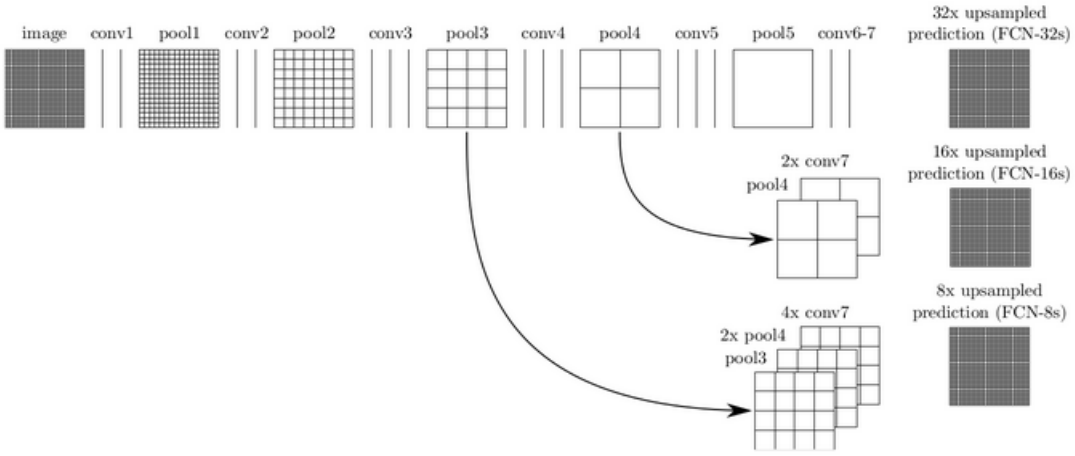


Figura 80 – Variantes de la arquitectura FCN

Las 3 variantes de la arquitectura principal difieren en el “stride” aplicado en la última convolución y las conexiones de “salto” utilizadas para obtener los mapas de segmentación de

salida. Se usa el término ruta de reducción de resolución para referirnos a la red hasta la capa de convolución nº7 y usaremos el término ruta de aumento de resolución para referirnos a la red compuesta por todas las capas después de capa de convolución nº7. Vale la pena señalar que las 3 arquitecturas FCN comparten la misma ruta de "downsampling", pero difieren en sus respectivas rutas de "upsampling". [37]

1. FCN-32: produce directamente el mapa de segmentación de la capa de convolución nº7, utilizando una capa de convolución transpuesta con "stride" 32.

2. FCN-16: Suma la predicción de un "upsampling" multiplicativo por 2 desde la convolución nº7 (usando una convolución transpuesta con "stride" 2) con la capa pool4 y luego produce el mapa de segmentación, utilizando una capa de convolución transpuesta con "stride" 16 al resultado anterior.

3. FCN-8: suma la salida de la convolución nº7 con un "upsampling" multiplicativo por 2 (obtenido con una convolución transpuesta de "stride" 2) con la capa "pool"4, les aplica un "upsampling" con una convolución traspuesta de "stride" 2 y los suma con la capa "pool" 3, y aplica una capa de convolución transpuesta con "stride" 8 en los mapas de características resultantes para obtener el mapa de segmentación

La función de coste que se acostumbra a usar es la función de Jaccard o IoU (Intersection Over Union, la cual ya se ha descrito en el apartado de detección)

$$jacc(P(class), GT(class)) = \frac{|P(class) \cap GT(class)|}{|P(class) \cup GT(class)|}$$

donde P es el mapa de segmentación predicho, GT es el mapa de segmentación real ("Ground Truth")

Los resultados obtenidos principalmente de la variante FCN-8 son relativamente buenos, pero aun así los resultados obtenidos no son lo suficientemente precisos, por lo que se ha buscado redes más eficientes.

2.3.4.2 SEGNET

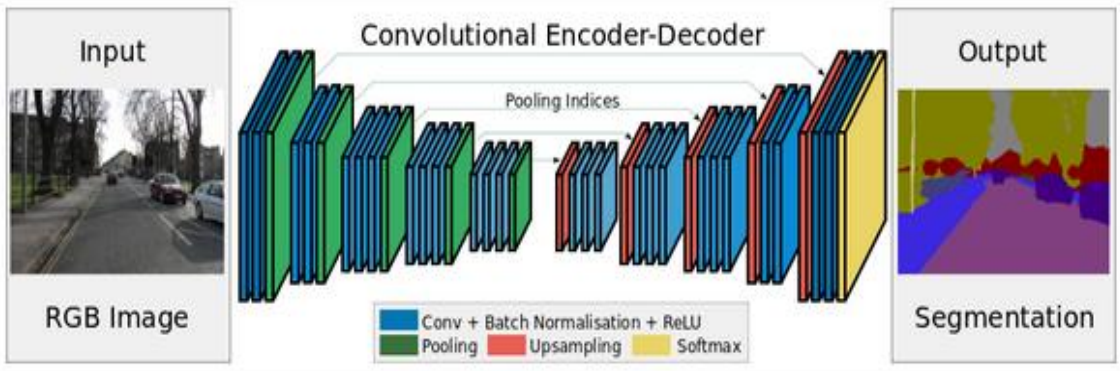


Figure 81 – Arquitectura tipo de Segnet

La novedad de la red SegNet radica en la manera en que el decodificador hace el "upsampling" de sus mapas de entrada de resolución más baja. Específicamente, el decodificador utiliza índices de "pooling" calculados en la etapa de "max-pooling" del codificador correspondiente

para realizar un “upsampling”. Esto elimina la necesidad de hacer un aprendizaje como se hace en la red FCN. Los mapas en el “upsampling” son escasos y luego se combinan con filtros entrenables para producir mapas de características más densos. [37]

El codificador acostumbra a ser una red pre-entrenada del tipo VGG sin las capas FC.

SegNet utiliza el “unpooling” para mostrar mapas de características en decodificador para usar y mantener intactos los detalles de alta frecuencia en la segmentación. Este método de “unpooling” es un método de interpolación diferente de los que se han explicado anteriormente.

Como se muestra en la figura 82, los índices en cada capa de “max-pooling” en el codificador se almacenan y luego se usan para sobre muestrear el mapa de características correlacionadas en el decodificador deshaciendo el camino del “max-pooling” usando esos índices almacenados. Si bien esto ayuda a mantener intacta la información de alta frecuencia, también omite información vecina cuando la operación de “unpooling” se hace desde mapas de características de baja resolución. Los valores que no se corresponden con los máximos acostumbran a rellenarse de ceros. [37]

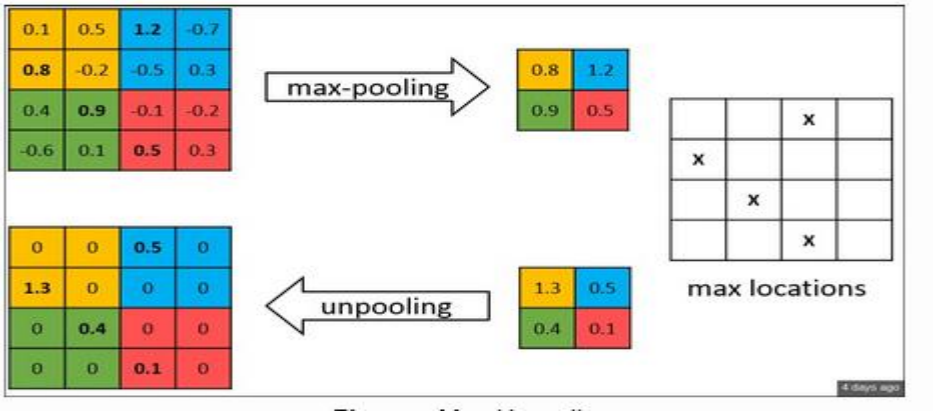


Figura 82 – Segnet Unpooling

En esta red no se utilizan convoluciones transpuestas, sino convoluciones normales con toda una serie de filtros y la recuperación de la dimensión se hace a través del “unpooling”.

Se utiliza una función de activación del tipo “Softmax” con una función de Coste Categorical Cross-Entropy.

2.3.4.3 U-NET

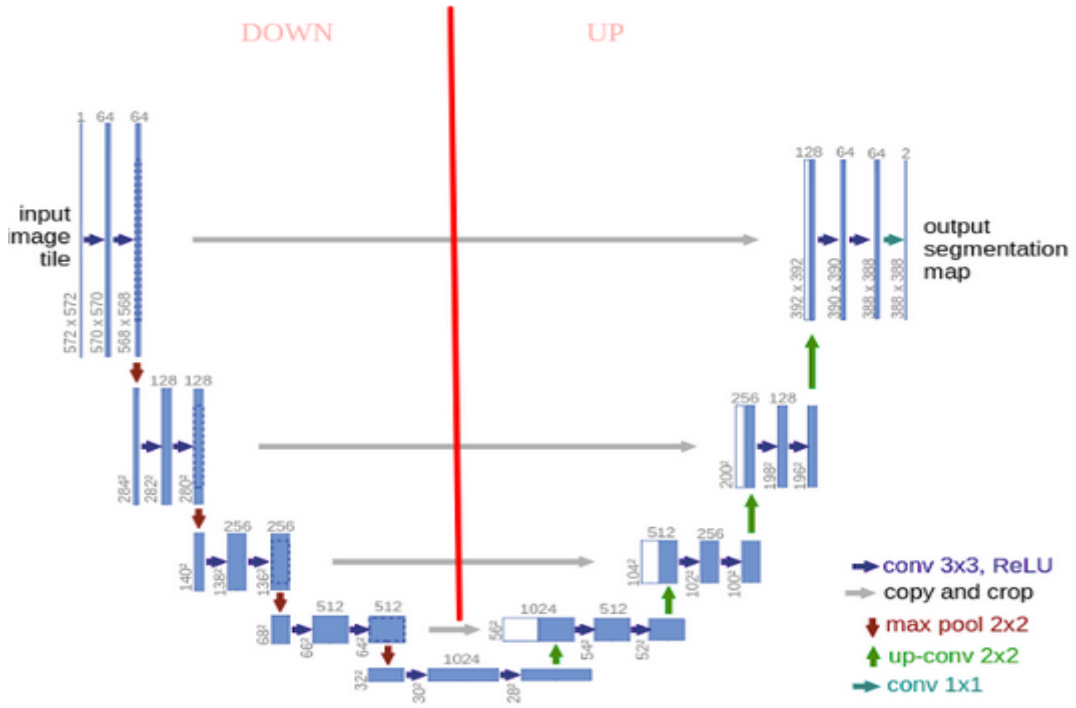


Figura 83 – Arquitectura de la U-NET

La arquitectura consiste en una ruta de contracción para capturar el contexto ("DOWN" en la figura 83) y una ruta de expansión simétrica que permite la localización precisa ("UP" en la figura 83). Una red de este tipo se puede entrenar de extremo a extremo a partir de muy pocas imágenes y supera los mejores métodos usados en la segmentación de estructuras neuronales en pilas de microscopio electrónico. Por otra parte, la red es rápida. La segmentación de una imagen de 572x572, como en la figura 83, lleva menos de un segundo en una GPU reciente. [38]

U-Net simplemente concatena los mapas de características del codificador con los mapas de características en el "upsampling" desde el decodificador en cada etapa para formar una estructura similar a una escalera. La arquitectura mediante sus conexiones de concatenación de salto permite al decodificador en cada etapa aprender las características relevantes que se pierden cuando se hace el "max-pooling" en el codificador. La concatenación se hace en el eje de los filtros, pero requiere que, en los demás ejes, las dimensiones sean las mismas. Este hecho implica que se tiene que hacer un recorte ("crop") en la entrada de la función de concatenación procedente del codificador de la red. [38]

En este tipo de red, el "upsampling" se puede hacer por convolución transpuesta o por interpolación "nearest neighbour".

Ambas técnicas generan una versión ampliada de la imagen original. El método de convolución transpuesta contiene parámetros que se pueden aprender, mientras que el enfoque de la interpolación del vecino más cercano es una operación fija sin parámetros "aprendibles". Todo parece indicar que usar la convolución transpuesta para realizar el "upsampling" puede producir predicciones más precisas en el conjunto de datos de prueba.

La función de coste utilizable podría ser una función "categorical cross-Entropy" o "binary cross-entropy".

2.4 Solución escogida

Existen otras soluciones que no se han estudiado en el caso de los asteroides y que se han descartado por diferentes motivos.

a) Flujo Óptico +CNN

La primera sería la utilización del cálculo del flujo óptico seguido de una CNN. Existen diferentes algoritmos entre los cuales el de “Lukas-Kanade” [39] o el de ”Gunner-Farneback” [39] que permiten de ver el desplazamiento de objetos. El resultado de la aplicación de dichos algoritmos nos da unos puntos(“vértices”) que se desplazan en la imagen y podemos deducir las zonas de la imagen que tienen desplazamiento. A partir del mapa creado en la imagen, podríamos entrenar una CNN y poder deducir posteriormente los cuerpos que se desplazan en fotos de validación.

En la figura 84, se dibuja para cada objeto uno o varios puntos(vértices) y se observa en sucesivos fotogramas su desplazamiento.

Figura 84 – Resultado del algoritmo de Lukas-Kanade

Así mismo, se puede ver en la figura siguiente el cálculo de la densidad de flujo óptico con el segundo método. Las zonas con color corresponden al desplazamiento de los objetos. Y en particular, el brillo del color de cada objeto está relacionado con su velocidad.

Figura 85 - Resultado del algoritmo de Gunner-Farneback

La aplicación de estos métodos requiere ciertas condiciones:

Al calcular el desplazamiento de los vértices es indispensable que la intensidad en sucesivos fotogramas o fotografías sea parecida. El fondo de la imagen en cada una de las fotografías

debería tener la misma intensidad y estar en la misma posición o desplazarse poco con relación a los objetos que se van a estudiar.

En el caso de los asteroides, se han hecho pruebas para ver su viabilidad y se ha podido constatar varios problemas:

1)La intensidad de los objetos estudiados en cada quinteto de fotos es diferente según las fotos

2) La forma de los objetos varía según la fotografía del quinteto.

3) El "ruido" se desplaza ya que las estrellas forman parte de ese ruido, y muchas de ellas tienen pequeños desplazamientos y pueden estar cerca de alguno de los objetos que se desplazan realmente. Además, el valor de desplazamiento de algunos de los asteroides es muy parecido al de las estrellas, por lo que aparece en el mapa de desplazamientos toda una serie de falsos movimientos que no permiten obtener resultados aceptables.

b) Correlación entre fotografías +CNN

Esta segunda solución es parecida al flujo óptico. Se efectúa una correlación entre 2 fotos sucesivas y el resultado se usa como entrada de una CNN. Pero existe el mismo problema que con el flujo óptico. Los desplazamientos de los asteroides son en bastantes casos muy pequeños y parecidos a los "desplazamientos" que parecen tener las estrellas por problemas de ruido. Lo que hace esta solución poco útil para la detección de asteroides.

c) RNN

Las redes neuronales recurrentes han dejado de utilizarse solamante para dar previsiones de resultados de bolsa o de palabras utilizadas en frases o de otros problemas análogos para ser utilizadas para la detección de objetos y para el movimiento de objetos.

El único problema es que, por el momento, los modelos actuales se han utilizado en casos más corrientes y simples (2 fotogramas sucesivos) y con un gran número de imágenes de entreno. El desplazamiento de asteroides en grupos de 5 fotos muy ruidosas y con pocos quintetos es un caso que por ahora no entra dentro los casos utilizables.

d)Redes "convolucionales" de detección y segmentación

Antes de analizar las posibles opciones que se han visto, hay que resaltar las características especiales que tiene nuestro problema. Se trata de ver dentro de una serie de quintetos los objetos que se están moviendo de manera uniforme. Si observamos las fotografías, hay realmente 2 clases de objetos en las fotos. La primera clase está formada por el ruido de fondo (sin objetos) y los objetos que no se desplazan y que forman principalmente las estrellas y finalmente la segunda clase está formada por los objetos que nos interesan y que se desplazan uniformemente.

Además, a partir del estudio hecho para el etiquetaje de las fotos, se puede constatar que existen diferentes tamaños de asteroides y en las fotos de gran tamaño puede haber objetos del tamaño de 5 píxeles o menos. Esto supondría un problema, para los algoritmos, que tienen en su arquitectura FC y son entrenados con fotos de tamaño relativamente pequeño, ya que posiblemente no detectaríamos dichos objetos. Esto eliminaría la mayoría de las redes de detección dejando únicamente la SSD y las redes de segmentación. Por la estructura que tiene la SSD no está bien resuelta para detectar objetos de pequeño tamaño. Además, utiliza redes pre-entrenadas que en nuestro caso no nos aportan nada ya que una de las tareas principales

es separar las estrellas de los asteroides. Los objetos que pretendemos separar no generan muchas características diferenciales y las redes pre-entrenadas aportan una gran ayuda en el caso de objetos con suficientes características diferenciales.

El último problema es que sólo tenemos 192 quintetos y tenemos que buscar una solución que podamos entrenar y obtener buenos resultados con pocos datos. Por todas estas razones la solución que parece adaptarse mejor a nuestro problema parece ser la utilización de la red U-Net. Esta red hecha originalmente para problemas médicos se utiliza para el reconocimiento de objetos especiales (células,órganos, …) que no acostumbran a aparecer en las redes pre-entrenadas y logra una buena precisión con pocos datos. Las otras redes de segmentación suelen utilizar redes pre-entrenadas ya que las fotos analizadas acostumbran a tener objetos más comunes que son de la misma clase o de clases parecidas a las clases clasificadas en dichas redes y necesitan sin estas redes pre-entrenadas, grandes cantidades de datos para lograr buenos resultados.

Por lo tanto, U-Net será la red escogida y vamos a mirar de adaptarla al tipo de fotos que tenemos.

2.5 Implementación

Como hemos visto en el apartado anterior, el modelo seleccionado que se acerca más a las condiciones de nuestro problema es el modelo U-net. Vamos a mirar por lo tanto de implementarlo en 2 casos (fotos originales y fotos retocadas).

Para hacerlo vamos a utilizar los resultados del etiquetaje de las fotografías hecho en la primera parte de este trabajo.

Para simplificar el problema miraremos de utilizar parte del código disponible en Internet. Existen varias versiones en las librerías Keras-Tensor Flow y escogeremos la más fácilmente adaptable. Estas librerías incorporaran toda una serie de funciones de alto nivel que cubren los diferentes bloques de la red que pretendemos diseñar.

Como el proceso de entrenamiento de la red se prevé largo para poder obtener buenos resultados, se escogerá un entorno que tenga una GPU y que sea de fácil utilización. Utilizaremos, a priori, el servicio Collaboratory de Google, ya que es gratuito y lo hemos utilizado anteriormente en el etiquetaje. El problema principal de este servicio es la poca duración del servicio: una máquina virtual de Collaboratory se reinicializa a las 12h de utilización y si no se han salvado los ficheros de datos anteriormente, éstos acaban perdiéndose.

Los “Jupyter Notebook” que facilita Collaboratory permiten de dividir el código en diferentes módulos que podemos editar y “compilar” por separado permitiendo una mayor facilidad en la depuración del código que un editor normal de PC. La mayor desventaja del editor de los “Jupyter Notebook” es su extrema simplicidad que ralentiza las posibles modificaciones de código que se pretendan hacer.

Se utilizará Python ya que es fácil de utilizar y engloba muchas librerías (numpy, scipy, astropy, keras, etc…), lo que permite de simplificar mucho el código final.

Utilizaremos por lo tanto los Jupyter Notebook del Collaboratory de Google con Python y subiremos los ficheros de fotos para entrenar y validar nuestro algoritmo.

En un primer paso vamos a escoger las dimensiones de entrada de las fotografías y esto nos va a servir para modificar los parámetros de la arquitectura de nuestra red y obtener los mejores resultados posibles.

El conjunto de fotos tiene unas dimensiones muy variables dependiendo del quinteto que se vaya a utilizar. Pero después de analizar los 192 quintetos, constatamos que existen 158 quintetos cuyas fotografías tienen dimensiones inferiores a 1024 píxeles x 1024 píxeles. Una solución sería optimizar la arquitectura de la red U-Net para estas dimensiones y de esta manera obtendríamos mejores resultados para fotografías de dimensiones inferiores. Así mismo, como una de las piezas fundamentales de la arquitectura es el "max-pooling" y este se utiliza en N bloques tenemos que coger una fotografía de entrada cuyas dimensiones sea divisible por 2^N. Muchas fotografías de dimensiones inferiores a 1024 x 1024 no son divisibles por 2^N, por lo que, aunque originalmente podríamos entrenar la red de segmentación con valores de entrada diferentes, es mejor redimensionar las fotos, de dimensiones más pequeñas, al formato 1024 x 1024. Para ello, se rellena la foto original hasta los nuevos bordes con valores de ruido Gaussiano como el que tiene la propia fotografía y se rellena con 0s la máscara correspondiente.

Además, la salida en la arquitectura original de la U-Net no tiene la misma dimensión la entrada y la salida de la red (in =572 píxeles x 572 píxeles, out=388 píxeles x 388 píxeles). Esto se debe al hecho que el "padding" utilizado es 0. Modificaremos el "padding"para conservar las dimensiones de la imagen: "padding" de 1 (para un filtro de convolución de 3x3)

Para entender cómo funcionan los parámetros ("padding", "stride") de una convolución mostraremos varios ejemplos:

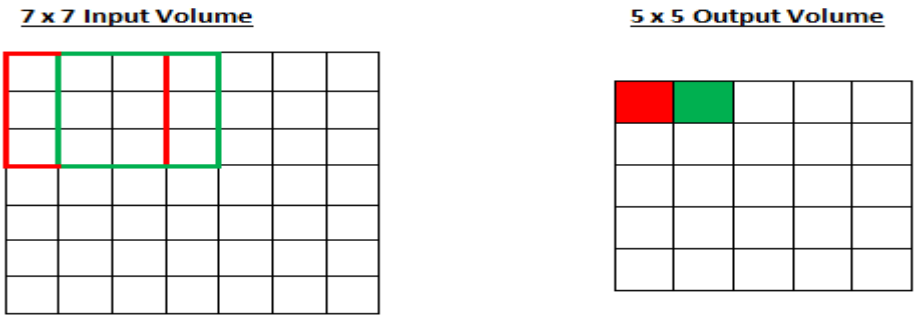


Figura 86 – Convolución filtro 3x3 con "padding"=0, "stride"=1

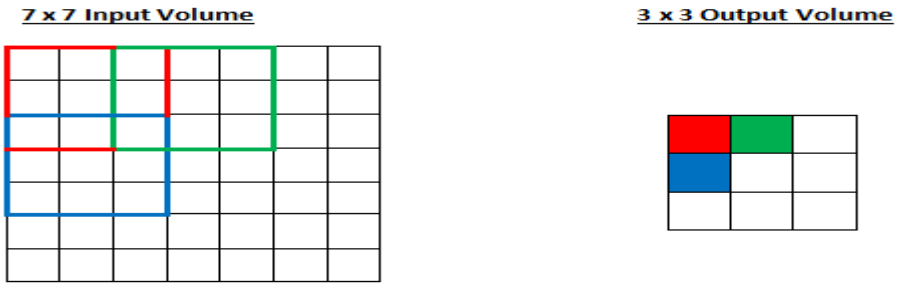


Figura 87 – Convolución filtro 3x3 con "padding"=0, "stride"=2

"Stride" representa el desplazamiento del filtro (en este ejemplo 3x3) con relación a la posición del filtro anterior.

Como podemos ver, en cualquiera de los 2 casos anteriores, existe una disminución en el tamaño de la foto. Para evitar esta pérdida de tamaño, se añade una o varias filas de 0s alrededor del borde de la fotografía antes de hacer la convolución. A este hecho lo llamamos “padding”. El número de filas de 0s alrededor de los bordes corresponde al valor del “padding”.

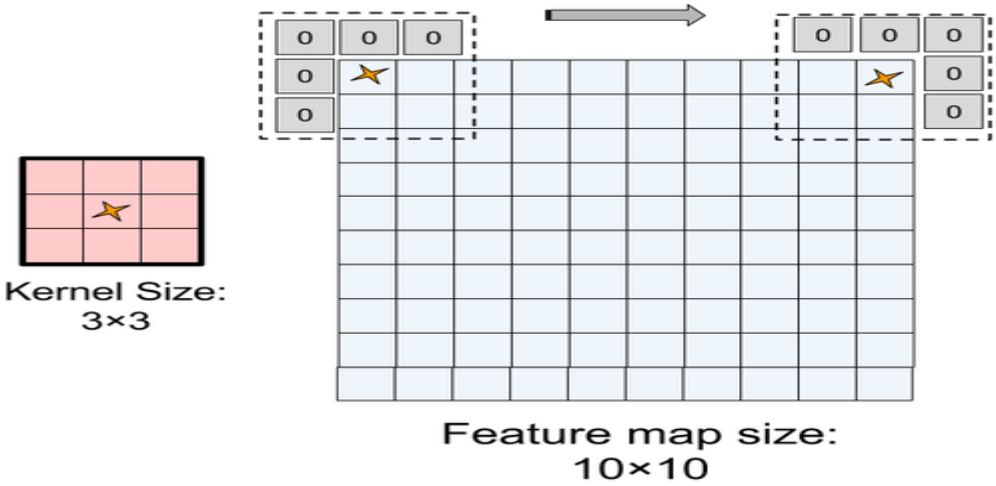


Figura 88 – Convolución filtro 3x3 con “padding”=1 y “stride”=1

Existen unas fórmulas que calculan las dimensiones de la imagen después de la convolución en función de las dimensiones de la imagen antes de la convolución y de los parámetros de la convolución (tamaño del filtro, “padding” y “stride”)

$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1$$

W y **H** son las dimensiones de la imagen de entrada, **output width** y **output height** las dimensiones de la imagen de salida, $\mathbf{F_w}$y $\mathbf{F_h}$ el tamaño del filtro de la convolución encada una de las dimensiones de la imagen, **P** el “padding” y $\mathbf{S_w}$ y $\mathbf{S_h}$ los “strides” en cada una de las dimensiones de la imagen.

En nuestra red U-Net, utilizaremos filtros de convolución con los parámetros siguientes:

padding=1 stride=1, filtros=3x3

por lo que obtendremos aplicando las fórmulas anteriores:

output width=(W-3+2) /1 +1=W

output height=(H-3+2) /1 +1=H

Como se puede ver a partir de las fórmulas, la modificación del “padding” permite de conservar las dimensiones.

Así mismo, en la parte de la arquitectura donde se hace el recorte y la concatenación de los elementos de baja resolución(codificador) con los de alta resolución(decodificador), al conservar

las dimensiones de la imagen en cada bloque del codificador, no hace falta hacer ningún recorte y sólo se hará la concatenación.

A partir de las características que se han descrito anteriormente, se puede presentar la arquitectura adaptada de la red U-Net.

La red se entrena con imágenes de formato 1024x1024 y se divide en 2 partes, la codificación y la decodificación.

La codificación está formada por 6 bloques compuestos de 2 funciones “convolucionales” seguidas de una función de “max-pooling” 2x2 que divide el tamaño de la imagen por 2 en cada uno de los ejes. Cada función “convolucional” está a su vez formado por 3 funciones: la convolución propiamente dicha con un número de filtros 3x3 variable según el número del bloque (cada convolución tiene un “padding” de 1 y un “stride de 1”, seguida por una función “Batch-Normalisation” y finalmente una función de activación del tipo “Relu”).

La parte inferior de la U llamada “cuello de botella” está formada por 2 funciones “convolucionales” (iguales que las descritas en el apartado de codificación) pero no están seguidas por un “max-pooling”. Por el contrario, al final están conectadas al apartado de decodificación a través de una función “up-convolution” 2x2.

La decodificación está formada por 6 bloques compuestos por una función “up-convolution” 2x2 que podría ser una interpolación o una convolución transpuesta, la salida de esta función está concatenada con la salida de uno de los grupos de 2 funciones “convolucionales” del apartado de codificación. Después de la concatenación, viene un bloque de 2 funciones “convolucionales” con la misma estructura que en la codificación.

Las concatenaciones se producen entre la salida del primer bloque de 2 funciones “convoluciones” del codificador y la entrada del último bloque de 2 funciones “convolucionales” del decodificador. Se conecta seguidamente el segundo bloque del codificador con el penúltimo bloque del decodificador y así sucesivamente hasta la conexión del último bloque del codificador con el primer bloque del decodificador.

Con relación a la imagen, pasamos de una imagen de 1024x1024 a una imagen de 16x16 en el cuello de botella con un número de filtros igual a 1024 para volver al final del decodificador a obtener un mapa de segmentación de 1024x1024. En la figura 89, se puede ver dicha red.

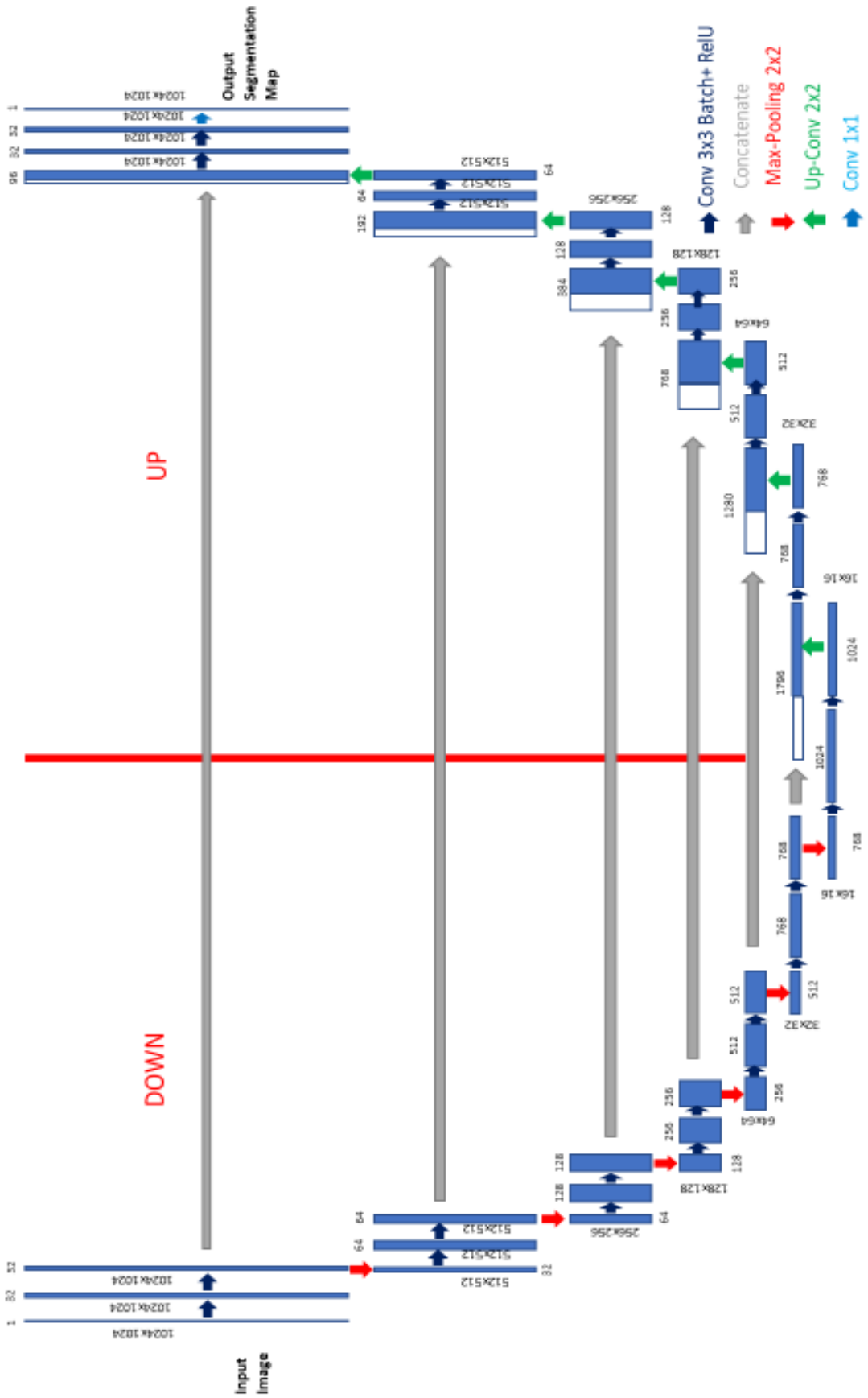


Figura 89 – Arquitectura U-Net "customizada"

La intención inicial es hacer diferentes tipos de pruebas:

- Pruebas de la red con las fotos tal cual se han recibido y posteriormente hacerlo con una modificación de dichas fotos, eliminando previamente todas las estrellas y todos los “blobs” que sean susceptibles de haberse generado de manera errónea a partir de la limpieza del ruido de cada una de las fotos

- Pruebas de la red usando y no usando la función “Batch-Normalisation” y en el caso de usarla modificando su posición, antes de la función de activación o después de la función de activación. Al comienzo se alegaba que aplicar la normalización a la salida de la convolución era mejor ya que era una salida lineal, tenía más probabilidades de ser simétrica, menos dispersa (en definitiva, más “Gaussiana”). Normalizarlo en ese punto era más probable que produjera activaciones con una distribución más estable. Posteriormente, se dijo que era mejor a la salida de la función de activación ya que la salida de convolución generaba valores negativos y normalizar unos valores para eliminarlos posteriormente en la activación, no tenía mucho sentido.

- Pruebas modificando la red aumentando el número de bloques para llegar a dimensiones inferiores a 16x16 en la zona del cuello de botella (por ejemplo, 8x8 o 4x4) y verificar la precisión obtenida, disminuyendo en las convoluciones, el número de filtros utilizados.

- Pruebas de las 2 opciones de “up-convolution” (interpolación simple, no “entrenable” y una convolución transpuesta, entrenable). La primera es menos precisa pero más simple y más rápida ya que no aporta más parámetros en el cálculo. La segunda es más precisa, pero es más lenta al ser “entrenable” y aportar más parámetros en el cálculo de la red. En el caso de Keras, parece no tener en su librería la función de interpolación “bilinear” que tiene “Torch”.

- Pruebas con otra función de coste y otra métrica. Existe una métrica parecida a la IoU recomendada desde Kaggle y una función de coste derivada de esta métrica que parece ofrecer una convergencia más rápida que la Binary Cross-Entropy.

El procedimiento que se va a seguir para las pruebas y validaciones de la red será el siguiente:

1.Una vez ejecutado el programa de detección de asteroides, se salvará el fichero correspondiente a la máscara con la posición de los píxeles que componen los asteroides en cada una de las fotos y así mismo se salvará el fichero de la fotografía modificada después de haber borrado todos los cuerpos que no son susceptibles de ser asteroides. La intensidad de los píxeles de los cuerpos “borrados” será reemplazada por un valor de una función Gaussiana con la media y la desviación estándar de la foto original. Los ficheros anteriores serán guardados en formato FITS y se salvará, de la misma manera, para la foto original y para la foto modificada los valores de la mediana (o la media) y de la desviación estándar en el apartado de los metadatos.

2.Se creerán los directorios IMAGES y MASKS donde se almacenarán respectivamente las fotografías y las máscaras correspondientes.

3.Se cargarán, según la prueba que se esté haciendo, las fotografías originales o las fotografías modificadas en el directorio IMAGES y las máscaras correspondientes en el directorio MASKS. Para el entreno se habrán preseleccionado las fotografías con las dimensiones inferiores o iguales a 1024 píxeles x 1024 píxeles.

4. Se leerán los ficheros FITS de los directorios IMAGES y MASKS (los ficheros de las fotografías y de las máscaras tendrán nombres parecidos de manera que cuando se lean se haga en el

mismo orden). En el momento de la lectura, para cada una de las fotografías seleccionadas, se leerá los metadatos de dicha fotografía para obtener la media y la desviación estándar y se generará una matriz una 1024x1024 de valores Gaussianos con esas mismas media y desviación estándar, y se rellenará dicha matriz desde el borde superior izquierdo con los valores de la matriz de la fotografía. De esta manera se habrán rellenado los valores de la matriz original en cada uno de los 2 ejes hasta el valor 1024 con los valores de una distribución Gaussiana similar. Para la máscara correspondiente se creerá una matriz de 1024x1024 de ceros y se insertará desde el borde superior izquierdo la máscara. Se obtendrá así, antes de aplicar a la red, las fotografías y las máscaras todas las matrices de una misma dimensión 1024x1024.

5. Las funciones de la librería Keras necesitan un formato determinado para la fotografías y máscaras. El formato requerido es un tensor de la forma:

[número de fotografía o máscara, número de líneas, número de columnas, número de canales]

En nuestro caso tendremos [nº fotografía o máscara, 1024,1024,1], ya que tenemos fotografías y máscaras en blanco y negro con dimensiones 1024x1024. Se transformará las 2 listas de fotografías y de máscaras en 2 tensores con el formato anterior.

6. Se utilizará a continuación la función “train_test_split” de la librería Sklearn de Python para separar las fotografías en un 70% para el entreno y en un 30% para la validación. En la misma proporción se separarán las máscaras.

7. Se generará, con las funciones concatenate, Conv2D, MaxPooling2D, UpSampling2D, Conv2DTranspose, BatchNormalisation, etc.., el modelo de la red que vamos a entrenar siguiendo la arquitectura de la figura 89. Se va a intentar hacer varias pruebas, como ya se ha anticipado anteriormente, cambiando el número de bloques y los parámetros de algunas de estas funciones. Se compilará el modelo utilizando la función “compile” pasando como parámetros un optimizador concreto, así como una función de coste y una métrica dadas. Por defecto, utilizaremos el optimizador Adam, la métrica “accuracy” y la función de coste Binary Cross-Entropy. Pero en varias publicaciones se recomienda el uso de otra métrica y de otra función de coste Dice_coef y Dice_loss. En todo caso, miraremos de poder probar los resultados con los 2 tipos de función de coste y de métrica.

8. Como no hay muchas fotografías para el entreno de la red se recurrirá a la función “ImageDataGenerator” de la librería Keras que permite de hacer transformaciones en las fotografías y en las máscaras al mismo tiempo (rotaciones, desplazamientos, simetrías, etc…). Para entrenar la red, se harán varias iteraciones sobre el conjunto de fotografías y de máscaras llamando “epoch” a ese número de iteraciones. En cada “epoch”, la función ImageData Generator genera de manera sincronizada una transformación para cada una de las fotografías y máscaras, de esta manera en cada “epoch” tendremos fotografías y máscaras diferentes, lo que aumentará artificialmente el número total de muestras en el entreno produciendo de esta manera mejores resultados.

9. Podemos ejecutar el entreno de la red con el cálculo de los coeficientes de los filtros de convolución en lo que se llama la “forward propagation” y la “backward propagation”. Pero podemos utilizar en este proceso todas las fotografías y las máscaras o simplemente unas cuantas y hacer este proceso de manera reiterativa hasta llegar al número total de fotografías y máscaras. Hacer estos cálculos parciales implica la utilización de “batches”. Este hecho implica en dividir el número total de muestras en n subgrupos llamados “batches” que tienen la misma longitud y calcular para cada uno de ellos los parámetros de la red.

Nº total de fotografías/máscaras= longitud_de_un_batch (batch_size) * Nºtotal_de_batches

Las ventajas de usar un tamaño batch_size inferior al número total de fotografías/máscaras son:
1. Requiere menos memoria. Dado que entrena la red con menos muestras, el procedimiento de entrenamiento general requiere menos memoria. Eso es especialmente importante si no puede ajustar todo el conjunto de datos en la memoria del ordenador.
2. Normalmente, las redes se entrenan más rápido con subgrupos. Eso es porque actualizamos los pesos después de cada propagación. Si por ejemplo tenemos, 512 muestras y utilizamos un "batch_size" de 16, propagaremos 32 subgrupos y actualizaremos 32 veces los parámetros de la red. Si utilizamos todas las muestras durante la propagación, haremos solo 1 actualización de los parámetros de la red.
Pero la utilización de subgrupos tiene una desventaja principal:
1. Cuanto menor sea el "batch-size", menos precisa será la estimación de los parámetros.

En nuestro caso haremos diferentes pruebas con un valor de "batch-size" entre 4 y 32 y miraremos qué valor nos da mejor compromiso entre precisión-rapidez-memoria.

2.5 Resultados

Como es obvio de ver por cómo ha transcurrido este trabajo, en este momento no existen resultados de las pruebas de los algoritmos de Deep Learning, ya que se está creando ahora las diferentes máscaras, y se están generando también las imágenes modificadas que se probarán en paralelo junto con las imágenes originales.

Pero, sí se puede comentar los resultados obtenidos para el programa de etiquetaje.

El programa parece ir bastante bien y encuentra, en varios de los quintetos, soluciones adicionales a las soluciones encontradas de manera manual. Existen algunas discrepancias en algunas soluciones ya que en ciertos casos los operarios han considerado como asteroides algunos cuerpos que se desplazan menos de 3 píxeles y el programa los considera estrellas. Así mismo, dichos operarios indican que unos cuerpos son asteroides cuando, en como mínimo una de las fotos, no se detecta ningún objeto y el programa considera esa posibilidad como ruido. Por último, existe un criterio de uniformidad en el movimiento que no cumplen algunas de las soluciones encontradas por los operarios.

El programa puede obtener un % de acierto bastante alto en la detección de asteroides, aunque en estos momentos es imposible que llegue al 100% por la existencia de casos imposibles de solventar. Se podría mirar de hacer una analogía con los 4 conceptos de la matriz de confusión: True Positive Rate, False Positive Rate, False Negative rate y True Negative Rate. Pero, existe en este caso, un problema principal: no se puede considerar fiables los datos facilitados por los operarios ya que queda por definir lo que es un asteroide en concepto de desplazamiento y de uniformidad de movimiento. Existe en estos momentos, a falta de unos ajustes finales en el programa, un 80% de acierto en la detección haciendo la relación del número de asteroides detectados por el programa (sin tener en cuenta los que encuentra adicionalmente) sobre el número de asteroides detectados manualmente. Pero, de esos 20% no detectados, el programa realmente detecta también el 13% pero considera que son estrellas o que el movimiento no es suficientemente uniforme o que falta un objeto en alguna de las fotografías.

Si consideramos los errores del programa y el caso en que sólo hay un objeto distinguible en las 5 fotografías, los asteroides no detectados representan el 7%.

La indefinición comentada anteriormente hace difícil utilizar los conceptos de la matriz de confusión para los asteroides detectados adicionalmente, ya que la mayoría de las detecciones adicionales dependen del concepto de uniformidad que se quiera fijar. Es decir, si se relaja el concepto de uniformidad la no detección originales bajará del 20% pero se encontrarán asteroides adicionales.

Seguidamente daremos algunas comparaciones del programa con los resultados obtenidos por los operarios.

A	B	C	D
80	q80_IAC80_20170622225643_154733+021056_+21_08_ast.fits	101.3982	68.60491
80	q80_IAC80_20170622225909_154733+021056_+21_08_ast.fits	375.7303	464.3189
80	q80_IAC80_20170622230135_154733+021056_+21_08_ast.fits	102.605	68.5673
80	q80_IAC80_20170622230402_154733+021056_+21_08_ast.fits	103.8074	69.80229
80	q80_IAC80_20170622230402_154733+021056_+21_08_ast.fits	381.004	466.2971
80	q80_IAC80_20170622230628_154733+021056_+21_08_ast.fits	100.8108	67.69332
80	q80_IAC80_20170622230628_154733+021056_+21_08_ast.fits	379.0645	466.3293

Figura 90 – Resultados obtenidos manualmente para el quinteto nº80

A	B	C	D
80	q80_IAC80_20170622225643_154733+021056_+21_08_ast.fits	375	464
80	q80_IAC80_20170622225643_154733+021056_+21_08_ast.fits	101.044	677.11
80	q80_IAC80_20170622225909_154733+021056_+21_08_ast.fits	375.667	464.333
80	q80_IAC80_20170622225909_154733+021056_+21_08_ast.fits	101.606	679.39
80	q80_IAC80_20170622230135_154733+021056_+21_08_ast.fits	376.667	465.333
80	q80_IAC80_20170622230135_154733+021056_+21_08_ast.fits	102.487	682.56
80	q80_IAC80_20170622230402_154733+021056_+21_08_ast.fits	378.5	466.5
80	q80_IAC80_20170622230402_154733+021056_+21_08_ast.fits	103.167	687.92
80	q80_IAC80_20170622230628_154733+021056_+21_08_ast.fits	379	467
80	q80_IAC80_20170622230628_154733+021056_+21_08_ast.fits	103.556	692.22

Figura 91 – Resultados obtenidos programáticamente para el quinteto nº80

A	B	C	D
23	q23_IAC80_20170525035311_145052+123400_+27_01_ast.fits	411.06	284.2311

Figura 92 – Resultados obtenidos manualmente para el quinteto nº23

A	B	C	D
23	q23_IAC80_20170525035311_145052+123400_+27_01_ast.fits	410.5	282.867
23	q23_IAC80_20170525035537_145052+123358_+27_01_ast.fits	416.128	284.744

Figura 93 – Resultados obtenidos programáticamente para el quinteto nº23

A	B	C	D
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	533.0831	346.409
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	533.3006	349.7086
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	533.5299	355.5173
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	535.7027	355.0034
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	533.8597	364.5475
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	535.4968	362.3716
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	535.4525	369.8034
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	537.9387	367.8433
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	530.1755	376.3888
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	535.4509	374.9703
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	536.6819	377.0244
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	538.1943	375.2037

Figura 94 – Resultados obtenidos manualmente para el quinteto nº1

A	B	C	D
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	551.333	605.667
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	455	603
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	785.444	412.222
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	419	373
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	651.5	358.5
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	533.417	348.311
1	q1_OIA_20170518010157_131252+395745_+43_00_ast.fits	885	273
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	552.333	606.667
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	456	604
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	785.667	412.889
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	419.333	373.667
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	652	359
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	533.885	354.865
1	q1_OIA_20170518010623_131252+395743_+43_00_ast.fits	886	274

A	B	C	D
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	552.333	607.667
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	456	605
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	787	415
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	420.5	374.5
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	653	360.5
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	534.409	362.22
1	q1_OIA_20170518011048_131252+395742_+43_00_ast.fits	887	276
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	554	609.008
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	457.556	606.222
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	788.292	416.375
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	421.667	376
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	535.464	369.268
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	654.095	362.095
1	q1_OIA_20170518011517_131251+395740_+43_00_ast.fits	888	277

A	B	C	D
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	554	610
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	458.5	607.5
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	789	417
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	422.133	376.867
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	535.376	375.926
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	655	363
1	q1_OIA_20170518011941_131251+395739_+43_00_ast.fits	888.667	277.667

Figura 95 – Resultados obtenidos programáticamente para el quinteto nº1

Como se puede constatar, el programa no repite las anotaciones en ninguna de las fotos y da resultados más coherentes en la uniformidad del movimiento. Y en el caso del quinteto nº1 como en el de unos cuantos quintetos más es capaz de encontrar soluciones adicionales a la(s) solución(es) original(es). En el caso del quinteto nº1, el programa es capaz de obtener 7 soluciones.

2.6 Costes

A nivel de costes, no me es demasiado fácil evaluar el número de horas efectivas utilizadas en la ejecución de este proyecto. Hay que decir que Google Collaboratory es una herramienta fácil de usar y gratuita pero que pueda implicar una gran pérdida de tiempo. El editor es muy simple, pero, en el caso de un programa con una cierta envergadura, cada modificación es mucho más laboriosa que para un editor de Python de un PC. Sus continuas desconexiones y la imposibilidad de guardar los datos en la máquina virtual alargan el proceso de prueba de una manera notable.

En todo caso, Intentaré dar una estimación de las horas utilizadas

Diseño del programa de etiquetaje: 30 horas

Codificación, pruebas y puesta a punto: 300 horas

Análisis de las redes neuronales, diseño y codificación de la red escogida: 80 horas

Documentación: 30 horas

Como es evidente, el proceso de codificación, pruebas y de puesta a punto del programa de puesta a punta ha sido lo más laborioso del proyecto. El número de horas utilizado no corresponde al número efectivo de horas ya que como he indicado anteriormente la utilización de Google Collaboratory ha sido muy poco óptima.

3.CONCLUSION

A partir de una serie de fotografías proporcionadas por los observatorios astronómicos de las Canarias y de un fichero de datos que indicaba la posición o las posiciones de posibles asteroides en dichas fotografías, se ha mirado de analizar diferentes algoritmos de Deep Learning que nos permitieran de prever a partir de un entreno previo, la posición de dichos asteroides en fotografías no utilizadas previamente.

En el momento de redactar esta memoria, la red escogida está es su fase inicial de entrenamiento y se espera poder tener resultados en el momento de la defensa del proyecto.

Esto ha sido debido a que el proyecto ha pasado de ser un proyecto de Machine Learning a ser una mezcla de Machine Learning y de tratamiento de imágenes, donde el apartado de tratamiento de imágenes ha sido preponderante, como mínimo en el tiempo utilizado.

En todo caso, el programa desarrollado para el etiquetaje de las fotografías, aunque no está totalmente probado ya que sólo se ha probado para menos de 200 quintetos de fotografías, puede servir de ayuda en un futuro a los operarios para poder centrarse en cuadrículas determinadas de las fotografías y poder determinar rápidamente la posición de posibles asteroides.

En lo que concierne, el apartado de Machine Learning y más concretamente de Deep Learning, la especificidad del problema nos ha llevado a un algoritmo de segmentación que se probará en breve y aunque viene precedido de buenos resultados para objetos que podrían asimilarse por su forma a los asteroides, el hecho de tener cuerpos celestes con tamaños tan pequeños podría generar una incertidumbre en el resultado. Aun así, se ha previsto toda una batería de pruebas cambiando diferentes parámetros en su estructura y en el contenido de las fotografías utilizando el programa de etiquetaje para poder ver su influencia en la precisión obtenida. Como ya se ha dicho anteriormente, se tiene previsto utilizar unas fotografías modificadas por el programa de etiquetaje en las que se ha suprimido muchos de los cuerpos celestes de los que se está seguro de que no son asteroides.

Durante la parte final de análisis, han aparecido varios artículos que citaban una variante de la convolución para poder obtener mejores resultados en segmentación de objetos pequeños, aunque no se ha podido acabar de analizar si era aplicable en este problema.

Finalmente, en los problemas de detección de movimiento de objetos, se ha constatado que han empezado a aparecer unas nuevas redes (redes neuronales recurrentes) que combinadas a las CNN pueden dar en un futuro relativamente cercano muy buenos resultados en este campo.

En lo que concierne a la herramienta gratuita Google Collaboratory, se aconseja no utilizarla en casos de proyectos con una cierta envergadura, ya que se producen muchas desconexiones de la máquina virtual sin previo aviso y que se tiene que cargar vez los datos originales ya que no permite de guardarlos.

4.REFERENCIAS

[1] https://es.wikipedia.org/wiki/Asteroide

[2] https://es.wikipedia.org/wiki/Cometa

[3] NASA's Asteroid Data Hunter challenge

https://github.com/PlanetaryResources/NTL-Asteroid-Data-Hunter

[4] https://en.wikipedia.org/wiki/Machine_learning

[5] https://en.wikipedia.org/wiki/Random_forest

[6] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[7] https://en.wikipedia.org/wiki/Deep_learning

[8] Observatorios de Canarias (Instituto de Astrofísica de Canarias)

http://vivaldi.ll.iac.es/OOCC/

[9] Understanding CCD Noise

http://www.qsimaging.com/ccd_noise.html

[10] https://en.wikipedia.org/wiki/FITS

[11] https://en.wikipedia.org/wiki/Ground_truth

[12] FITS file handling(astropy.io.fits)

http://docs.astropy.org/en/stable/io/fits/

[13] Sigma_clipped_stats

http://docs.astropy.org/en/stable/api/astropy.stats.sigma_clipped_stats.html

[14] Denoising an image with the median filter

https://www.scipy-lectures.org/advanced/image_processing/auto_examples/plot_denoising.html

[15] https://es.wikipedia.org/wiki/OpenCV

[16] OpenCv Contours: Getting Started

https://docs.opencv.org/3.0.0/d4/d73/tutorial_py_contours_begin.html

[17] OpenCv Contours: More functions

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_more_functions/py_contours_more_functions.html

[18] A quick introduction to Neural Networks

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

[19] An intuitive explanation of Convolutional Networks

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[20] https://en.wikipedia.org/wiki/Overfitting

[21] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[22] Batch Normalisation in Neural Networks

https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

[23] Drop-Out in (Deep) Machine Learning

https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[24] Object Localization and Detection

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html

[25] Object recognition for Dummies Part 3: R-CNN and Fast/Faster/Mask R-CNN and YOLO

https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html#yolo-you-only-look-once

[26] Review of Deep Learning Algorithms for Object detection

https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

[27] Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN, YOLO, SSD

https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

[28] Intersection over Union for Object Detection

https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[29] Loss Functions in Deep Learning

http://yeephycho.github.io/2017/09/16/Loss-Functions-In-Deep-Learning/

[30] Regression Functions All Machine Learners should know

https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

[31] https://en.wikipedia.org/wiki/Support_vector_machine

[32] Understanding SSD Multibox – Real Time Object Detection in Deep Learning

https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

[33] How to do Semantic Segmentation using Deep Learning?

https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef

[34] Up-Sampling with Transposed Convolution

https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

[35] https://en.wikipedia.org/wiki/Bilinear_interpolation

[36] Biomedical Segmentation with U-net

https://ai.intel.com/biomedical-image-segmentation-u-net/

[37] Semantic Segmentation using Fully Convolutional Networks over the years

https://meetshah1995.github.io/semantic-segmentation/deep-learning/pytorch/visdom/2017/06/01/semantic-segmentation-over-the-years.html

[38] U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

https://arxiv.org/pdf/1505.04597.pdf

[39] OpenCV Optical Flow

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html