



MEMORIA PRÁCTICA APRENDIZAJE AUTOMÁTICO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Aplicación del aprendizaje automático para la detección de aviones en imágenes de satélite

Autor 1: David García Ramallal

Autor 2: Pablo Páramo Telle

Autor 3: Pelayo Vieites Pérez

A Coruña, mayo de 2023.

Índice general

1	Introducción	1
2	Descripción del problema	3
2.1	Restricciones al problema	3
2.2	Descripción de la BD	3
2.3	Origen de la BD	4
2.4	Condiciones de elaboración de la BD	4
2.5	Propiedades relevantes de los datos	5
3	Análisis bibliográfico	6
4	Desarrollo	8
4.1	Primera aproximación	8
4.1.1	Descripción	8
4.1.2	Resultados	9
4.1.3	Discusión	12
4.2	Segunda aproximación	13
4.2.1	Descripción	13
4.2.2	Resultados	13
4.2.3	Discusión	16
4.3	Tercera aproximación	17
4.3.1	Descripción	17
4.3.2	Resultados	17
4.3.3	Discusión	20
4.4	Cuarta aproximación	21
4.4.1	Descripción	21
4.4.2	Resultados	21
4.4.3	Discusión	24

4.5	Aproximación con Deep Learning	25
4.5.1	Descripción	25
4.5.2	Resultados	25
4.5.3	Discusión	28
5	Conclusiones	30
6	Trabajo futuro	31
	Lista de acrónimos	32
	Bibliografía	33

Índice de figuras

1.1	Esquema con las partes de un avión	1
1.2	Ejemplo de detección con escasa visibilidad	2
2.1	Ejemplo de negativo (a la izquierda) y positivo (a la derecha)	3
2.2	Ejemplo de escena de la que se toman las capturas	4
4.1	Ejemplo de avión del dataset	8
4.2	Gráfica de los valores de loss de la primera ejecución del fold inicial	10
4.3	Píxeles elegidos para tomar los valores RGB en la 2 ^a aproximación	13
4.4	Píxeles elegidos para las nuevas características en la 3 ^a aproximación	17
4.5	Píxeles elegidos para las nuevas características en la 4 ^a aproximación	21

Índice de tablas

4.1	Valores RGB de media y desviación típica de la imagen anterior	9
4.2	Modelos de RNA entrenados en la 1 ^a aproximación	9
4.3	Modelos de SVM en la 1 ^a aproximación	10
4.4	Modelos de Árbol de Decisión en la 1 ^a aproximación	11
4.5	Modelos de kNN en la 1 ^a aproximación	11
4.6	Matriz de confusión SVM-M6 de la 1 ^a aproximación	12
4.7	Modelos de RNA entrenados en la 2 ^a aproximación	14
4.8	Modelos de SVM en la 2 ^a aproximación	14
4.9	Modelos de Árbol de Decisión en la 2 ^a aproximación	15
4.10	Modelos de kNN en la 2 ^a aproximación	15
4.11	Matriz de confusión SVM-M8 de la 2 ^a aproximación	16
4.12	Modelos de RNA entrenados en la 3 ^a aproximación	18
4.13	Modelos de SVM en la 3 ^a aproximación	18
4.14	Modelos de Árbol de Decisión en la 3 ^a aproximación	19
4.15	Modelos de kNN en la 3 ^a aproximación	19
4.16	Matriz de confusión SVM-M5 de la 3 ^a aproximación	20
4.17	Modelos de RNA entrenados en la 4 ^a aproximación	22
4.18	Modelos de SVM en la 4 ^a aproximación	22
4.19	Modelos de Árbol de Decisión en la 4 ^a aproximación	23
4.20	Modelos de kNN en la 4 ^a aproximación	23
4.21	Modelos entrenados en la aproximación de Deep Learning	28

Capítulo 1

Introducción

La detección de objetos en imágenes por ordenador se ha convertido, en los últimos años, en un campo en auge debido, principalmente, a la inteligencia artificial y el aprendizaje automático. Desde encontrar matrículas en fotografías hasta localizar señales de tráfico y personas en un coche en movimiento, el número de posibles aplicaciones es inimaginable.

Gracias a los avances en dicha materia no sólo podemos emplear estos sistemas para identificar elementos que el propio ojo humano puede apreciar con facilidad, sino que también son útiles en problemas de más complicada solución, como el que presentamos: la detección de aviones en imágenes de satélite, la cual a día de hoy se realiza de manera manual y ofrece poca seguridad y eficiencia.

Las aeronaves, a pesar de que pueden presentar formas ligeramente diferentes, suelen disponer de partes bien diferenciadas, que las hacen reconocibles frente a otros objetos, tal y como son el fuselaje, las alas o los estabilizadores.

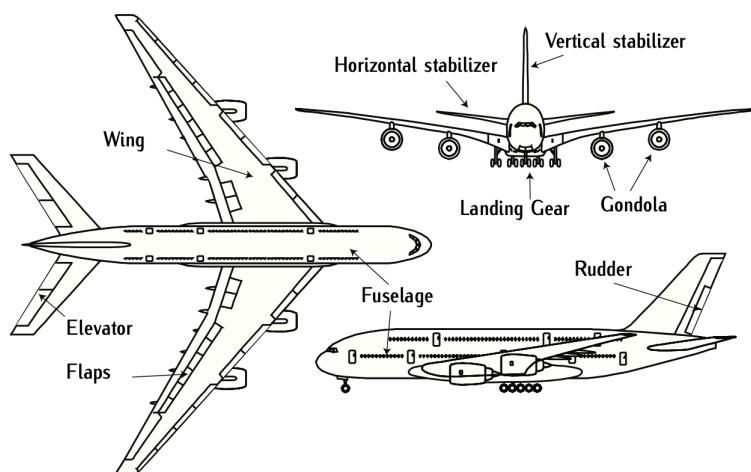


Figura 1.1: Esquema con las partes de un avión

Algunas de las problemáticas que se manifiestan en esta tarea son la variación de tamaño y color de las aeronaves, además de su orientación en el momento de la fotografía y la falta de contraste con el fondo, que en muchos casos dificultan enormemente la tarea y dejan sin identificar algunos de estos objetos. Por ejemplo, en la siguiente imagen se encuentran dos aviones que resultan casi imperceptibles al ojo humano [1].

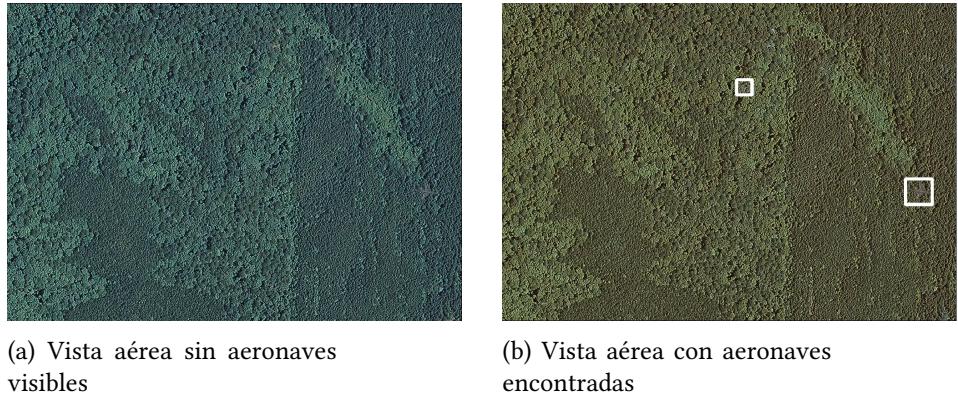


Figura 1.2: Ejemplo de detección con escasa visibilidad

En cuanto a las posibles aplicaciones prácticas se refiere, un sistema que automáticamente detecte aeronaves podría tener en el ámbito militar su principal foco de atención. La gestión y seguimiento de los aviones en las pistas de los aeropuertos sería otra utilidad potencial.

Nuestro objetivo con este proyecto es obtener un sistema de Aprendizaje Automático que detecte de forma autónoma la presencia o no de aeronaves en una determinada fotografía.

Para la realización de este problema emplearemos principalmente 4 técnicas: [Red de Neuronas Artificiales \(RNA\)](#), [Árboles de Decisión](#), [k-nearest neighbors algorithm \(kNN\)](#) y [Máquinas de Soporte Vectorial \(SVM\)](#).

Esta memoria estará organizada en diferentes apartados, empezando por la sección que nos ocupa, donde realizamos una presentación del problema e introducimos los métodos seguidos. A continuación nos centraremos en la descripción del problema, analizando las restricciones a las que está sujeto, además de la propia base de datos. Despues pasaremos al análisis bibliográfico, en el que referenciaremos diferentes artículos científicos e investigaciones que buscan resolver la misma problemática. En el propio desarrollo de la solución, explicaremos detalladamente las diferentes aproximaciones que hemos realizado. Por último tendremos un apartado de conclusiones, donde nos centraremos en lo que habremos logrado con este proyecto, y uno de trabajo futuro, en el que visualizaremos una hipotética implementación de la solución en la vida real.

Capítulo 2

Descripción del problema

2.1 Restricciones al problema

Las imágenes empleadas en este proyecto están tomadas de día, desde un punto de vista cenital y disponen de la suficiente resolución y calidad como para no distorsionar la forma de las aeronaves. No hay presencia de nubes y los fragmentos presentan una buena iluminación.



Figura 2.1: Ejemplo de negativo (a la izquierda) y positivo (a la derecha)

2.2 Descripción de la BD

La base de datos utilizada dispone de 32.000 imágenes satelitales RGB de 20x20 de aeropuertos de California, EE.UU. Estas están extraídas de varias escenas (similares a las de la figura 2.2), estando algunas de ellas también incluídas junto al resto de fotografías. Los fragmentos cuentan con una clasificación de "1" o "0" dependiendo de si contienen una aeronave o no. También presentan un ID de escena para poder conocer de cuál fue extraída y unas coordenadas para localizarlos dentro de la propia escena.

Con el fin de acortar los tiempos de realización de los apartados experimentales de este proyecto se tomó la decisión de emplear únicamente 2000 imágenes, divididas a partes iguales entre fragmentos que contienen aviones y fragmentos que no.



Figura 2.2: Ejemplo de escena de la que se toman las capturas

2.3 Origen de la BD

Los datos empleados en este proyecto proceden de la base de datos "Planes in Satellite Imagery", realizada por Robert Hammell, y ha sido obtenida de la página web www.kaggle.com [2].

2.4 Condiciones de elaboración de la BD

Las imágenes empleadas en esta BD han sido tomadas de día y presentan ejemplos variados de distintas aeronaves (están tomadas en aeropuertos), además de múltiples escenarios circundantes tales como masas de agua, vegetación o edificaciones.

2.5 Propiedades relevantes de los datos

Las imágenes que forman parte de nuestra base de datos son capturas cuadradas de 20 píxeles de lado. Están en color, en escala [RGB](#).

De la base de datos original hemos seleccionado semi-aleatoriamente mil imágenes positivas y mil negativas, eliminando de los negativos aquellas aeronaves que sólo aparecían parcialmente y de las positivas las imágenes con peor visibilidad.

A la hora de analizar los resultados hemos determinado que además de la precisión, otra métrica que podría resultarnos útil es *F1-score*, puesto que esta evita los resultados engañosos que produce la precisión cuando la distribución de clases está demasiado desbalanceada.

Capítulo 3

Análisis bibliográfico

Con el fin de recabar información para elaborar nuestro proyecto hemos buscado y analizado diversos trabajos relacionados con el reconocimiento de aviones en imágenes, todos ellos enfocados desde el punto de vista del Aprendizaje Automático.

Una de las técnicas más empleadas en estos documentos es [You Only Look Once \(YOLO\)](#), un modelo de detección de objetos en imágenes y videos basado en Deep Learning. En el trabajo de S. A. Mansoori et al [3], al igual que en el de A. Tahir [4] se utiliza dicha técnica aprovechando que el algoritmo de [YOLO](#) divide la imagen en una cuadrícula y realiza una sola pasada de predicción para identificar los objetos. De esta forma se pueden localizar diversas aeronaves al mismo tiempo en un único fotograma.

En las investigaciones de U. Alganci [5] y en la llevada a cabo por universidades de Pakistán e Irlanda [6] se combinan los modelos [YOLO](#) con otros basados en redes neuronales convolucionales denominados [Region-based Convolutional Neural Network \(RCNN\)](#), además de sus versiones mejoradas: [Fast RCNN](#) y [Faster RCNN](#), modelos más eficientes, veloces y precisos.

Este último es en el que se basa el artículo de S. Rawat [1]. En él, a partir de otra base de datos de Kaggle, se explica cómo usar las [Faster RCNN](#) para detectar aviones en imágenes satelitales, siendo su sistema distinto al que nosotros planteamos: este no se centra en clasificar las imágenes individualmente en *aeronave* o *no aeronave*, sino que las detecta y señala en la imagen en la que aparecen. Previo al entrenamiento de su sistema de detección de aeronaves, el autor comenta que usa técnicas de *Data Augmentation*, que permiten mejorar la calidad del sistema mediante la creación de nuevos datos de entrenamiento a partir de los ya existentes.

Otra técnica empleada conjuntamente con [YOLO](#) es [Super-Resolution Generative Adversarial Network \(SRGAN\)](#), consistente en aumentar la resolución de imágenes de baja calidad distorsionando la información lo menos posible. Los autores Y. Wang et al [7] hicieron uso de esta combinación para crear el modelo Super-Resolution YOLO (SR-YOLO), el cual usan en su proyecto.

CAPÍTULO 3. ANÁLISIS BIBLIOGRÁFICO

Por su parte, H. Wu et al [8] emplean el algoritmo **Binarized Normed Gradient (BING)**, que genera unas regiones candidatas a contener aeronaves analizando la forma de los bordes en la imagen.

Por último, en el estudio dirigido por A. Zhao [9] se utilizan funciones **Aggregate Channel Features (ACF)**, que extraen 3 canales de cada píxel:

- Canales de color (escala de grises, **RGB** y **LUV**)
- Magnitud de gradiente normalizado
- Histogramas de gradiente orientado

Para caracterizar las imágenes, utilizan un algoritmo **Nonmaximum Suppression (NMS)**, que devuelve, en muchos casos, caracterizaciones dobles dentro de una misma imagen, debido a la parte posterior de las aeronaves o a su sombra. Para solucionar este problema, proponen un algoritmo que actúa en relación al grado de superposición de dos caracterizaciones positivas, bien eliminando aquella con menor grado de seguridad, bien combinando ambas, dependiendo del valor de superposición que se alcance.

Capítulo 4

Desarrollo

PARA el desarrollo del proyecto, realizaremos diferentes aproximaciones, en las que analizaremos los resultados obtenidos a partir de 4 técnicas de Aprendizaje Automático: [Red de Neuronas Artificiales \(RNA\)](#), [Árboles de Decisión](#), [k-nearest neighbors algorithm \(kNN\)](#) y [Máquinas de Soporte Vectorial \(SVM\)](#). Para cada una de ellas, extraeremos diferentes características de las imágenes, y discutiremos cuáles son las mejores para que el sistema identifique las aeronaves, objetivo principal del trabajo.

4.1 Primera aproximación

4.1.1 Descripción

Para esta primera aproximación, emplearemos como características la media y desviación típica de los valores RGB de cada imagen. Al presentar todos los fragmentos condiciones similares de iluminación, resolución y calidad podemos concluir que los valores serán diferentes en el caso de que la fotografía presente un avión o no lo haga, por lo que es un buen punto de partida para el sistema.

La base de datos empleada en esta primera aproximación es la misma que la definida en la sección "Descripción del problema", contando con 1000 imágenes para cada clase (positivo si es un avión, y negativo en el caso contrario).

En cuanto al preprocesado de los datos se refiere, la normalización no es necesaria puesto que tenemos la certeza de que los datos, al ser RGB, se encuentran ya acotados tanto inferior como superiormente (entre 0 y 1).



Figura 4.1: Ejemplo de avión del dataset

	Rojo	Verde	Azul
Media	0.782206	0.754657	0.729000
Desviación Típica	0.153383	0.153863	0.161140

Tabla 4.1: Valores RGB de media y desviación típica de la imagen anterior

Para estimar el error en test y minimizar la importancia del azar en nuestros experimentos hemos empleado el método de validación cruzada (*cross-validation*), mediante el cual nos aseguramos de que cada dato es utilizado al menos una vez para entrenar y una vez para testear. En concreto recurriremos a *10-fold cross-validation*, siendo k igual a 10.

4.1.2 Resultados

Entrenamos varias **RNA** con diferentes topologías de 1 y 2 capas ocultas. Recurrimos al método de *10-fold cross-validation*, con 50 ejecuciones por cada fold. Además, todos los modelos tienen en común que el porcentaje del datos del conjunto de validación es del 20%, que el número máximo de ciclos sin mejorar el error de validación está fijado en 15, y que el error mínimo es de 0. Mostramos los resultados obtenidos para cada una de ellas en la tabla 4.2.

Modelo	Topología	Tasa aprendizaje	max_{ciclos}	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	[5]	0.01	800	0.8157	0.0181	0.8251	0.0173
2	[8 4]	0.01	800	0.7963	0.0152	0.8099	0.0179
3	[6 3]	0.01	800	0.7941	0.0167	0.7973	0.0293
4	[12]	0.01	800	0.8111	0.0155	0.8251	0.0130
5	[4 8]	0.01	800	0.7958	0.0178	0.7944	0.0310
6	[5]	0.02	500	0.8096	0.0150	0.8245	0.0130
7	[4]	0.01	800	0.8132	0.0169	0.8222	0.0171
8	[6]	0.01	800	0.8144	0.0175	0.8273	0.0166

Tabla 4.2: Modelos de RNA entrenados en la 1^a aproximación

Los valores de precisión media y de *F1-Score* medio rondan, para cada uno de los modelos, el 80% (entre el 79% y el 82% y entre el 79% y 83% respectivamente).

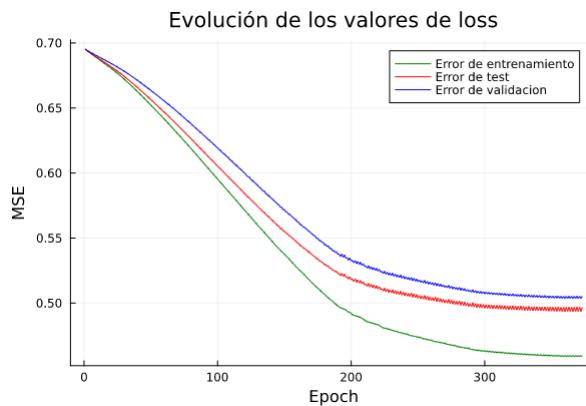


Figura 4.2: Gráfica de los valores de loss de la primera ejecución del fold inicial

Después de realizar los experimentos con RNA decidimos probar [SVM](#), árboles de decisión y [kNN](#). Para ello utilizamos la librería de Python *Scikit-Learn*, en su implementación para Julia. En cada modelo empleamos diferentes combinaciones de los principales parámetros existentes.

Para Máquinas de Soporte Vectorial, como se puede ver en la tabla 4.3, probamos con diferentes valores de C y tipos de kernel, además de grados y gammas del mismo.

Modelo	Kernel	Grado del kernel	γ_{kernel}	C	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	rbf	-	2	1	0.8445	0.0225	0.8549	0.0205
2	rbf	-	auto	1	0.7880	0.0229	0.8164	0.0185
3	rbf	-	scale	1	0.8455	0.0241	0.8567	0.0218
4	linear	-	2	1	0.8010	0.0289	0.8238	0.0252
5	poly	3	2	1	0.8580	0.0246	0.8642	0.0228
6	poly	3	2	10	0.8640	0.0197	0.8680	0.0189
7	rbf	-	auto	10	0.8430	0.0190	0.8551	0.0170
8	poly	4	2	1	0.8595	0.0223	0.8626	0.0219

Tabla 4.3: Modelos de SVM en la 1^a aproximación

En esta técnica, se mejoran los resultados de las RNA, siendo el mejor modelo el de kernel polinómico con grado 2 y gamma 2, teniendo la constante C el valor máximo de los modelos que entrenamos, con lo que comprobamos que, con este parámetro, se controla el sobreajuste y la generalización en el conjunto de test es mejor.

En Árboles de Decisión la variación entre los diferentes modelos se corresponde con la profundidad máxima del árbol. Los resultados aparecen en la tabla 4.4.

Modelo	Profundidad máxima	$\mu_{\text{precisión}}$	$\sigma_{\text{precisión}}$	$\mu_{F1-\text{score}}$	$\sigma_{F1-\text{score}}$
1	2	0.8075	0.0287	0.8233	0.0242
2	4	0.8280	0.0291	0.8416	0.0266
3	8	0.8335	0.0181	0.8418	0.0167
4	16	0.8310	0.0185	0.8330	0.0204
5	32	0.8200	0.0111	0.8188	0.0121
6	64	0.8200	0.0111	0.8188	0.0121

Tabla 4.4: Modelos de Árbol de Decisión en la 1^a aproximación

En la figura vemos que no existe ninguna mejora notable comparado con los métodos anteriores, teniendo una precisión media entre las obtenidas con [RNA](#) y [SVM](#).

Por último, en el Algoritmo de k-vecinos más cercanos, cuyos resultados se adjuntan en la tabla 4.5, el parámetro variable es el número de vecinos.

Modelo	Número de vecinos	$\mu_{\text{precisión}}$	$\sigma_{\text{precisión}}$	$\mu_{F1-\text{score}}$	$\sigma_{F1-\text{score}}$
1	2	0.8050	0.0367	0.7915	0.0435
2	3	0.8485	0.0307	0.8563	0.0293
3	5	0.8565	0.0274	0.8650	0.0249
4	10	0.8585	0.0273	0.8662	0.0259
5	15	0.8590	0.0250	0.8698	0.0215
6	30	0.8455	0.0223	0.8557	0.0202

Tabla 4.5: Modelos de kNN en la 1^a aproximación

Los resultados para [kNN](#) podríamos afirmar que son, después de los de [SVM](#), los mejores. La precisión media ronda el 85% en prácticamente todos los modelos y el *F1-Score* el 86%

4.1.3 Discusión

En este análisis se evaluaron diferentes modelos de aprendizaje automático: [SVM](#), Árboles de decisión, [kNN](#) y [RNA](#). Cada uno tiene sus ventajas y desventajas, y su empleo en un problema depende de la naturaleza del mismo y de su complejidad.

Para nuestro caso, hemos probado los cuatro modelos con parámetros y configuraciones diferentes con el fin de comprobar cuál era el mejor para la primera aproximación. De entre todos ellos, el que ha devuelto mejores resultados en las métricas utilizadas ha sido el Modelo 6 de los [SVM](#), con un kernel polinómico de grado 3, un valor de C de 10 y un valor de gamma del kernel de 2. El valor de la precisión media ascendió hasta el 86.40%, con una desviación típica del 1.97% mientras que el *F1-Score* medio fue del 86.80% con una desviación típica del 1.89%.

	Predictión Negativo	Predictión Positivo
Real Negativo	834	166
Real Positivo	106	894

Tabla 4.6: Matriz de confusión SVM-M6 de la 1^a aproximación

Los otros modelos consiguieron dar resultados aceptables en términos de precisión y F1-Score, pero no tan buenos como los del SVM, como se puede observar en las tablas. Sin embargo, a pesar de estos buenos resultados, hay que tener en cuenta que en este punto exacto del desarrollo del proyecto, la solución presentada no es más que un clasificador de colores, puesto que las únicas características empleadas son la media y la desviación típica de los valores RGB de las imágenes. En próximas aproximaciones sumaremos más características, hasta que finalmente tengamos un detector fiable de aeronaves en imágenes de satélite, que es el objetivo principal de este proyecto.

4.2 Segunda aproximación

4.2.1 Descripción

En esta segunda aproximación tendremos como objetivo mejorar la calidad de la solución elaborada en la primera iteración. Para conseguirlo añadiremos las siguientes características a las ya utilizadas anteriormente: media y desviación típica de los valores RGB de los píxeles centrales de las imágenes.

Las fotografías que conforman nuestra base de datos presentan una dimensión de 20x20 píxeles, por lo que elegiremos como centro los píxeles entre el 8 y el 13, tanto en filas como en columnas.

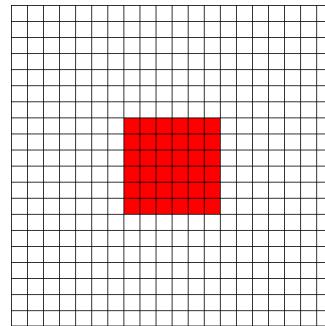


Figura 4.3: Píxeles elegidos para tomar los valores RGB en la 2^a aproximación

La selección de estas características se debe a que en el centro de las imágenes clasificadas como positivas siempre está presente la parte central del avión, independientemente de la orientación del mismo.

En lo relativo al preprocesado de los datos, y de la misma manera que ocurre en la aproximación anterior, no realizaremos normalización. Por otro lado, como en la primera iteración, haremos validación cruzada.

4.2.2 Resultados

Para esta segunda aproximación, al igual que hicimos en la primera, ejecutamos configuraciones diferentes para cada modelo: 8 arquitecturas distintas para Redes de Neuronas Artificiales, con 1 o 2 capas ocultas; 8 configuraciones de hiperparámetros para Máquinas de Soporte Vectorial, con kernels y valores de C variados; 6 valores de profundidad diferentes para Árboles de Decisión y 6 valores de k para el Algoritmo de k-vecinos más cercanos.

En la siguiente tabla podemos ver los resultados para [RNA](#). En todas las arquitecturas se recurre a *10-fold cross-validation* con 50 ejecuciones por fold, un ratio del 20% para el conjunto de validación, 15 ciclos máximos sin mejorar el error de validación y un error mínimo de 0.

Modelo	Topología	Tasa aprendizaje	max_{ciclos}	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	[5]	0.01	800	0.8985	0.0197	0.9042	0.0172
2	[8 4]	0.01	800	0.8827	0.0244	0.8866	0.0259
3	[6 3]	0.01	800	0.8729	0.0163	0.8737	0.0144
4	[12]	0.01	800	0.8897	0.0174	0.8974	0.0146
5	[4 8]	0.01	800	0.8790	0.0136	0.8789	0.0129
6	[5]	0.02	500	0.8796	0.0182	0.8882	0.0148
7	[4]	0.01	800	0.9036	0.0192	0.9087	0.0170
8	[6]	0.01	800	0.8951	0.0188	0.9013	0.0162

Tabla 4.7: Modelos de RNA entrenados en la 2^a aproximación

Los resultados de las Redes Neuronales son muy constantes entre todos los modelos entrenados, con variaciones del 3% entre los menores y los mayores valores de precisión y *F1-Score* medio.

En la tabla 4.8 podemos ver los resultados de las ejecuciones de los modelos de Máquinas de Soporte Vectorial que hemos escogido para esta aproximación y que son los mismos que en la anterior (variaciones de tipo de kernel, grado y gamma del kernel y valor de C)

Modelo	Kernel	Grado del kernel	γ_{kernel}	C	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	rbf	-	2	1	0.9335	0.0133	0.9353	0.0122
2	rbf	-	auto	1	0.8805	0.0206	0.8885	0.0178
3	rbf	-	scale	1	0.9220	0.0197	0.9247	0.0181
4	linear	-	2	1	0.9000	0.0230	0.9056	0.0199
5	poly	3	2	1	0.9520	0.0174	0.9528	0.0168
6	poly	3	2	10	0.9540	0.0182	0.9546	0.0177
7	rbf	-	auto	10	0.9165	0.0204	0.9200	0.0186
8	poly	4	2	1	0.9545	0.0194	0.9550	0.0188

Tabla 4.8: Modelos de SVM en la 2^a aproximación

Como se puede ver en la tabla, tanto los resultados de la precisión como los del *F1-score* han mejorado notablemente con los modelos de [SVM](#), destacando el modelo 6, con kernel polinómico de grado 3, gamma 2 y valor de C 10.

Para Árboles de decisión, cuyos resultados aparecen en la tabla 4.9, las profundidades máximas empleadas son iguales a las ejecutadas en la primera iteración.

Modelo	Profundidad máxima	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	2	0.8870	0.0193	0.8882	0.0188
2	4	0.9005	0.0203	0.9046	0.0188
3	8	0.9155	0.0148	0.9162	0.0147
4	16	0.9095	0.0169	0.9100	0.0173
5	32	0.9100	0.0162	0.9104	0.0167
6	64	0.9100	0.0162	0.9104	0.0167

Tabla 4.9: Modelos de Árbol de Decisión en la 2^a aproximación

En los modelos de Árboles de Decisión podemos observar resultados que se encuentran a medio camino entre los obtenidos con [RNA](#) y los obtenidos con [SVM](#).

Finalmente, la tabla 4.10 muestra los resultados obtenidos al aplicar el algoritmo [kNN](#). En este caso también se han utilizado las mismas configuraciones de hiperparámetros que en la aproximación inicial (número de vecinos).

Modelo	Número de vecinos	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	2	0.9305	0.0172	0.9293	0.0187
2	3	0.9345	0.0154	0.9367	0.0143
3	5	0.9380	0.0151	0.9399	0.0144
4	10	0.9365	0.0178	0.9385	0.0170
5	15	0.9310	0.0165	0.9337	0.0155
6	30	0.9190	0.0185	0.9217	0.0173

Tabla 4.10: Modelos de kNN en la 2^a aproximación

Por último, las estadísticas obtenidas al ejecutar los modelos de [kNN](#) nos presenta esta técnica como la segunda con mejores resultados de las cuatro empleadas.

4.2.3 Discusión

De igual manera que se realizó en la aproximación inicial, en el análisis de esta iteración se evaluaron los siguientes 4 modelos de aprendizaje automático con diferentes arquitecturas y configuraciones : [RNA](#), [SVM](#), Árboles de decisión y [kNN](#).

Tal y como sucedió en el primer avance, el modelo que mejores resultados ha entregado ha sido un [SVM](#) con kernel polinómico y valor de gamma 2. Sin embargo, en este caso el grado del kernel es de 4 en vez de 3, y el valor de C de 1 en lugar de 10. El valor medio para la precisión ascendió desde un 86.40% hasta un 95.45%, mientras que la desviación típica pasó del 1.97% al 1.94%. En cuanto al *F1-Score* se refiere, su media y desviación típica mejoraron de un 86.80% y 1.89% a un 95.50% y 1.88% respectivamente. En la siguiente tabla podemos observar la matriz de confusión del modelo.

	Predictión Negativo	Predictión Positivo
Real Negativo	946	54
Real Positivo	37	963

Tabla 4.11: Matriz de confusión SVM-M8 de la 2^a aproximación

A diferencia de lo que ocurría en la primera aproximación, en la que el sistema era poco más que un mero clasificador de colores, en este punto del proyecto ya podemos afirmar que entrega unos resultados relativamente satisfactorios.

En futuras entregas, y mediante la suma de características adicionales, confiamos en pulir la solución hasta obtener un detector de aeronaves con una alta fiabilidad.

4.3 Tercera aproximación

4.3.1 Descripción

Para mejorar el sistema de detección de aeronaves en esta aproximación, y de la misma manera que se hizo en iteraciones anteriores, añadiremos nuevas características a las ya empleadas hasta este punto del proyecto.

Tras analizar los posibles posicionamientos de los aviones en imágenes cuadradas, como las que conforman nuestra Base de Datos, llegamos a la conclusión de que en la gran mayoría de ocasiones el aparato ocupa una de las diagonales o una de las dos rectas que divide la imagen en cuatro cuadrantes.

Con el objetivo de aprovecharnos de esta circunstancia añadiremos 24 características nuevas: la media y la desviación típica de los valores RGB de cada una de las rectas comentadas, y que se pueden observar, junto con los píxeles que abarca cada una, en la siguiente figura.

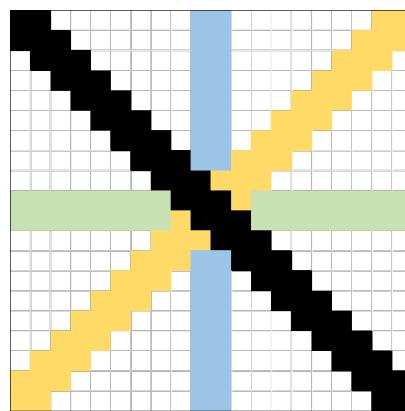


Figura 4.4: Píxeles elegidos para las nuevas características en la 3^a aproximación

Por otro lado, y como es habitual para datos de esta índole (RGB), no realizaremos ningún tipo de normalización. Sí que recurriremos, al igual que en las aproximaciones anteriores, validación cruzada con 10 folds.

4.3.2 Resultados

Tal y como hicimos en las dos primeras aproximaciones, en esta iteración trabajamos con las siguientes 4 técnicas: Redes de Neuronas Artificiales, Máquinas de Soporte Vectorial, Árboles de Decisión y el Algoritmo de k-vecinos más cercanos.

Para [RNA](#) utilizamos 8 arquitecturas distintas, con topologías de 1 o 2 capas ocultas. En todas ellas se emplea *cross-validation* de 10 folds con 50 ejecuciones por fold, un ratio del 0.2 para el conjunto de validación, 15 ciclos como máximo sin mejorar el error en validación y un error mínimo de 0.0. En la siguiente tabla podemos ver los resultados.

Modelo	Topología	Tasa aprendizaje	max_{ciclos}	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	[5]	0.01	800	0.8325	0.0206	0.8442	0.0177
2	[8 4]	0.01	800	0.8074	0.0267	0.8233	0.0236
3	[6 3]	0.01	800	0.8110	0.0242	0.8228	0.0224
4	[12]	0.01	800	0.8254	0.0218	0.8377	0.0193
5	[4 8]	0.01	800	0.7952	0.0238	0.8104	0.0219
6	[5]	0.02	500	0.7917	0.0234	0.8041	0.0187
7	[4]	0.01	800	0.8401	0.0199	0.8517	0.0169
8	[6]	0.01	800	0.8276	0.0212	0.8402	0.0175

Tabla 4.12: Modelos de RNA entrenados en la 3^a aproximación

La figura nos muestra que la variación entre el mejor resultado y el peor es relativamente amplia en cuanto a precisión media se refiere (un 5%).

En cuanto al resto de técnicas se refiere, para **SVM** recurrimos a 8 configuraciones de hiperparámetros con kernels y valores de C variados, manteniendo las configuraciones de la primera y la segunda aproximación. Los resultados se adjuntan en la tabla 4.13.

Modelo	Kernel	Grado del kernel	γ_{kernel}	C	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	rbf	-	2	1	0.9425	0.0150	0.9441	0.0143
2	rbf	-	auto	1	0.8665	0.0197	0.8769	0.0173
3	rbf	-	scale	1	0.9170	0.0167	0.9196	0.0159
4	linear	-	2	1	0.9120	0.0199	0.9160	0.0182
5	poly	3	2	1	0.9640	0.0171	0.9644	0.0169
6	poly	3	2	10	0.9625	0.0162	0.9629	0.0158
7	rbf	-	auto	10	0.9180	0.0193	0.9210	0.0181
8	poly	4	2	1	0.9595	0.0080	0.9596	0.0080

Tabla 4.13: Modelos de SVM en la 3^a aproximación

Ante estos resultados podemos observar una gran mejoría respecto a los de **RNA**, en concreto de casi el 12% en precisión media en los casos más óptimos de cada técnica.

Para Árboles de Decisión mantenemos los 6 diferentes modelos ya comentados en las aproximaciones 1 y 2. En la tabla 4.14 se pueden ver los valores obtenidos en las ejecuciones.

Modelo	Profundidad máxima	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	2	0.8905	0.0171	0.8910	0.0171
2	4	0.8995	0.0220	0.9032	0.0201
3	8	0.9075	0.0287	0.9098	0.0273
4	16	0.9025	0.0296	0.9033	0.0290
5	32	0.8955	0.0301	0.8952	0.0305
6	64	0.8955	0.0301	0.8952	0.0305

Tabla 4.14: Modelos de Árbol de Decisión en la 3^a aproximación

Las ejecuciones de Árboles de Decisión no han mejorado la precisión obtenida con los [SVM](#), pero tampoco se han quedado por debajo de las [RNA](#), manteniéndose entre ambas técnicas.

En [kNN](#), con resultados en la tabla 4.15, utilizamos los 6 valores de k de las iteraciones anteriores.

Modelo	Número de vecinos	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	2	0.9340	0.0237	0.9332	0.0240
2	3	0.9340	0.0163	0.9360	0.0157
3	5	0.9315	0.0180	0.9337	0.0174
4	10	0.9285	0.0163	0.9307	0.0159
5	15	0.9240	0.0158	0.9270	0.0152
6	30	0.9140	0.0182	0.9163	0.0176

Tabla 4.15: Modelos de kNN en la 3^a aproximación

Los resultados de [kNN](#) nos muestran mejores estadísticas que los de [RNA](#) y Árboles de Decisión, aunque no llegan a ser tan buenos como los de [SVM](#).

4.3.3 Discusión

En el análisis de esta aproximación, y como se hizo en las dos anteriores, se evaluaron las 4 técnicas siguientes: Redes de Neuronas Artificiales, Máquinas de Soporte Vectorial, Árboles de Decisión y el Algoritmo de k-vecinos más cercanos.

A diferencia de lo que sucedió con la iteración 2, en este caso no se ha producido una mejora de las métricas en cada una de las técnicas. Para [RNA](#) los resultados han sido muy inferiores, con decrementos de entre el 5% y el 10% en la precisión y el *F1-Score* para cada modelo. El resto de técnicas, por su parte, no han sufrido una disminución tan drástica en los valores de las métricas, pero estos se han mantenido bastante estables, con mejoras ligeras en algunos casos.

Esta circunstancia podría deberse a que las nuevas características empleadas en esta aproximación no aportan información nueva que sea relevante para solucionar el problema.

Sin embargo, si que se ha obtenido un nuevo máximo absoluto tanto en precisión (96.40% de media y 1.71% de desviación típica) como en *F1-Score* (96.44% y 1.69%). Este ha correspondido con el modelo 5 de [SVM](#), con kernel polinómico de grado 3, valor 2 de gamma y valor 1 para C. Su matriz de confusión es la siguiente:

	Predictión Negativo	Predictión Positivo
Real Negativo	956	44
Real Positivo	28	972

Tabla 4.16: Matriz de confusión SVM-M5 de la 3^a aproximación

Las características nuevas añadidas en esta aproximación serán descartadas para sucesivas iteraciones puesto que los resultados obtenidos no han sido todo lo satisfactorios que nos gustaría.

4.4 Cuarta aproximación

4.4.1 Descripción

Para la cuarta aproximación descartaremos las características agregadas en la iteración anterior, puesto que su empleo no suponía una mejoría relevante, y añadiremos otras nuevas a las de las dos primeras. En concreto, la media y la desviación típica de los valores RGB de cada una de las cuatro esquinas que conforman las imágenes de nuestra Base de Datos.

En la siguiente figura se pueden ver los píxeles seleccionados.

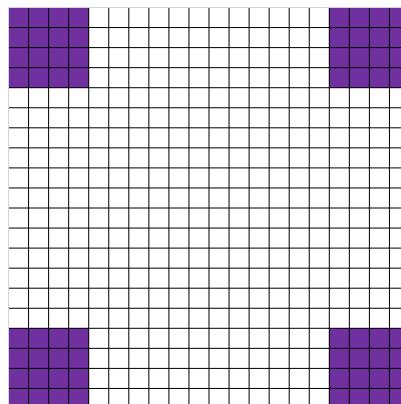


Figura 4.5: Píxeles elegidos para las nuevas características en la 4^a aproximación

Puesto que los datos son de tipo RGB, y como ya hemos comentado anteriormente para casos similares, no es necesario normalizarlos.

Por último, al igual que se hizo en las tres aproximaciones anteriores, recurriremos al método de validación cruzada, en concreto con 10 folds.

4.4.2 Resultados

De igual manera que en el resto de aproximaciones, en esta iteración trabajamos con las siguientes 4 técnicas de Aprendizaje Automático: Redes de Neuronas Artificiales, Máquinas de Soporte Vectorial, Árboles de Decisión y el Algoritmo de k-vecinos más cercanos.

En [RNA](#) ejecutamos 8 arquitecturas diferentes. Todas ellas presentan una red de 1 o 2 capas ocultas, un ratio del 20% para el conjunto de validación, 15 ciclos máximos sin mejorar el error de validación y un error mínimo de 0. Las variaciones entre modelos se encuentran en el número de neuronas por capa, en la tasa de aprendizaje y en el número máximo de ciclos. Además, en todos los casos el número de ejecuciones por fold es igual a 50. En la siguiente tabla presentamos los resultados.

Modelo	Topología	Tasa aprendizaje	max_ciclos	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	[5]	0.01	800	0.8671	0.0208	0.8755	0.0168
2	[8 4]	0.01	800	0.8440	0.0209	0.8539	0.0170
3	[6 3]	0.01	800	0.8453	0.0287	0.8529	0.0265
4	[12]	0.01	800	0.8456	0.0173	0.8574	0.0141
5	[4 8]	0.01	800	0.8376	0.0247	0.8437	0.0221
6	[5]	0.02	500	0.7864	0.0138	0.7915	0.0128
7	[4]	0.01	800	0.8723	0.0219	0.8804	0.0189
8	[6]	0.01	800	0.8629	0.0211	0.8717	0.0167

Tabla 4.17: Modelos de RNA entrenados en la 4^a aproximación

Podemos observar que los valores obtenidos al entrenar [RNA](#) son bastante variables, con diferencias de casi el 9% entre el mejor y el peor modelo.

Para [SVM](#), el número de configuraciones diferentes de hiperparámetros también será de 8. Los cambios entre cada una de ellas serán en el tipo, grado y gamma del kernel, además de en el valor de C. En la tabla 4.18 tenemos lo obtenido al ejecutar.

Modelo	Kernel	Grado del kernel	γ_{kernel}	C	$\mu_{precisión}$	$\sigma_{precisión}$	$\mu_{F1-score}$	$\sigma_{F1-score}$
1	rbf	-	2	1	0.9545	0.0144	0.9555	0.0136
2	rbf	-	auto	1	0.8580	0.0153	0.8711	0.0127
3	rbf	-	scale	1	0.9250	0.0163	0.9274	0.0149
4	linear	-	2	1	0.9120	0.0160	0.9158	0.0148
5	poly	3	2	1	0.9600	0.0201	0.9604	0.0200
6	poly	3	2	10	0.9500	0.0173	0.9504	0.0169
7	rbf	-	auto	10	0.9215	0.0145	0.9241	0.0129
8	poly	4	2	1	0.9460	0.0171	0.9463	0.0163

Tabla 4.18: Modelos de SVM en la 4^a aproximación

Los resultados conseguidos con dicha técnica son, como viene siendo habitual a lo largo de esta memoria, ampliamente superiores a los de [RNA](#).

En Árboles de Decisión, al igual que en el resto de aproximaciones, solo variamos la profundidad del árbol: desde 2 hasta 64 pasando por el resto de potencias de 2 entre medio.

Modelo	Profundidad máxima	$\mu_{\text{precisión}}$	$\sigma_{\text{precisión}}$	$\mu_{F1-\text{score}}$	$\sigma_{F1-\text{score}}$
1	2	0.8870	0.0193	0.8882	0.0188
2	4	0.8970	0.0208	0.9015	0.0192
3	8	0.9085	0.0275	0.9114	0.0254
4	16	0.9125	0.0247	0.9132	0.0239
5	32	0.9115	0.0248	0.9122	0.0242
6	64	0.9115	0.0248	0.9122	0.0242

Tabla 4.19: Modelos de Árbol de Decisión en la 4^a aproximación

Como se puede observar en la tabla, los Árboles de Decisión han sido más o menos constantes, sin una gran diferencia entre los diferentes modelos.

Por último tenemos **kNN**. Disponemos de 6 modelos diferentes, siendo el número de vecinos la única diferencia entre ellos. En la figura 4.20 mostramos los resultados.

Modelo	Número de vecinos	$\mu_{\text{precisión}}$	$\sigma_{\text{precisión}}$	$\mu_{F1-\text{score}}$	$\sigma_{F1-\text{score}}$
1	2	0.9125	0.0241	0.9125	0.0240
2	3	0.9170	0.0197	0.9212	0.0177
3	5	0.9215	0.0189	0.9254	0.0171
4	10	0.9205	0.0210	0.9235	0.0196
5	15	0.9140	0.0190	0.9184	0.0173
6	30	0.9100	0.0189	0.9150	0.0170

Tabla 4.20: Modelos de kNN en la 4^a aproximación

Por su parte, los resultados de **kNN** han sido relativamente similares a los de Árboles de Decisión, incluso ligeramente más constantes entre modelos.

4.4.3 Discusión

Para esta aproximación, tal y como hicimos en las anteriores, valoramos cuatro técnicas de Aprendizaje Automático: [RNA](#), [SVM](#), [kNN](#) y Árboles de Decisión.

Por un lado, en [kNN](#) tanto la media como la desviación típica de la precisión y del *F1-Score* han sido ligeramente inferiores respecto tanto a la aproximación 3 como ante la aproximación 2.

En Árboles de Decisión, en cambio, se podría decir que no se ha notado prácticamente ningún cambio entre las métricas de las tres iteraciones. De igual manera ha sucedido en el caso de los [SVM](#), con la salvedad de que, a diferencia de la aproximación 3, en este caso no se ha obtenido ninguno máximo en alguna de las métricas.

Por último, en lo relativo a las [RNA](#), los resultados han sido superiores a los de la aproximación anterior, con un aumento de alrededor del 3% tanto en precisión como en *F1-Score*. Sin embargo, no han llegado a los obtenidos en la segunda iteración, quedándose relativamente lejos.

Por lo tanto, y una vez analizados los resultados, consideramos que las características agregadas en esta aproximación no han añadido información de suficiente relevancia y, como consecuencia, descartamos la iteración.

En la próxima recurrimos al *Deep Learning* con el objetivo de mejorar la calidad de nuestra solución y cumplir nuestro objetivo, obtener un detector de aeronaves en imágenes de satélite de alta fiabilidad.

4.5 Aproximación con Deep Learning

4.5.1 Descripción

Para esta última aproximación no añadiremos nuevas características a las ya empleadas en iteraciones anteriores como hemos venido haciendo a lo largo de la memoria, sino que cambiaremos de técnica de Aprendizaje Automático.

Pasaremos de las cuatro utilizadas hasta este momento (Redes de Neuronas Artificiales convencionales, Máquinas de Soporte Vectorial, Árboles de Decisión y el Algoritmo de k-vecinos más cercanos) a emplear Aprendizaje Profundo (más conocido por su nombre en inglés, *Deep Learning*) y redes convolucionales.

Estas redes están especialmente diseñadas para la clasificación de imágenes y tienden a producir muy buenos resultados en todo lo relativo a reconocimiento de formas por lo que somos optimistas en su aplicación para nuestro problema, basado en el reconocimiento de aviones en imágenes de satélite.

En cuanto al tamaño de las imágenes se refiere, no realizaremos reescalado puesto que ya se encuentran en formato fijo de 20x20 píxeles.

Además, y de igual manera que se realizó en las anteriores iteraciones, se recurrirá a validación cruzada con 10 folds.

4.5.2 Resultados

En esta aproximación hemos experimentado con 8 arquitecturas diferentes. Para todas ellas el número máximo de ciclos sin mejora en el conjunto de entrenamiento es de 10, se para el entrenamiento si dicho conjunto llega a una precisión del 99.9% y la tasa de aprendizaje empieza en 0.001 para ir disminuyendo cada vez que no se produce mejora en 5 ciclos consecutivos. Para cada fold se realizan un total de 15 ejecuciones.

Se detallan a continuación las capas que componen los modelos utilizados durante esta aproximación. Cabe destacar que todas las capas convolucionales usan la función de activación ReLU:

- **Modelo 1:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 32->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa totalmente conectada

- **Modelo 2:**

- Capa convolucional: 3->16 filtros de tamaño 5x5.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 5x5.
- Capa convolucional: 32->32 filtros de tamaño 5x5.
- Capa totalmente conectada

- **Modelo 3:**

- Capa convolucional: 3->16 filtros de tamaño 2x2.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 2x2.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 32->32 filtros de tamaño 2x2.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa totalmente conectada

- **Modelo 4:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 32->32 filtros de tamaño 3x3.
- Capa totalmente conectada

- **Modelo 5:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa totalmente conectada

- **Modelo 6:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 32->64 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa totalmente conectada

- **Modelo 7:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 16->32 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa convolucional: 32->64 filtros de tamaño 3x3.
- Capa totalmente conectada

- **Modelo 8:**

- Capa convolucional: 3->16 filtros de tamaño 3x3.
- Capa de Max-Pooling que divide el tamaño en 2 en los ejes x e y
- Capa totalmente conectada

En la siguiente tabla presentamos los diferentes valores de media y desviación típica de la precisión y el *F1-Score* obtenidos con estos modelos:

Modelo	$\mu_{\text{precisión}}$	$\sigma_{\text{precisión}}$	$\mu_{\text{F1-score}}$	$\sigma_{\text{F1-score}}$
1	0.7637	0.0467	0.8264	0.0303
2	0.8394	0.0249	0.8596	0.0189
3	0.6995	0.0826	0.7882	0.0516
4	0.7893	0.0370	0.8349	0.0222
5	0.6403	0.0524	0.7459	0.0307
6	0.8375	0.0268	0.8705	0.0184
7	0.8239	0.0442	0.8548	0.0301
8	0.5622	0.0239	0.6976	0.0158

Tabla 4.21: Modelos entrenados en la aproximación de Deep Learning

4.5.3 Discusión

Tal y como hemos comentado en el apartado de Descripción, en esta aproximación no hemos trabajado con las mismas técnicas de Aprendizaje Automático que en las iteraciones anteriores, sino que hemos recurrido a *Redes Convolucionales*.

La expectativa inicial era que los resultados obtenidos fuesen ampliamente superiores a los conseguidos con redes no profundas, máquinas de soporte vectorial, árboles de decisión o kNN; ya que las CNN trabajan de buena manera en el ámbito del reconocimiento y la clasificación de formas en imágenes.

Sin embargo, nuestros resultados no han sido todo lo satisfactorios que esperábamos. De entre todos los modelos, los más destacados han sido el 6, el 2 y el 7, con precisiones medias entre el 82 y el 84% y *F1-Score* entre 85% y 87%. Estos tienen como punto común que los tres están conformados por tres capas convolucionales, además de una capa final totalmente conectada. La diferencia radica en el número de capas de *Max-Pooling* empleadas. Mientras que el M6 utiliza tres, el M7 solamente está formado por dos y el M2 únicamente por una. Por ello, el modelo 6 llega a trabajar con cuadros más pequeños, puesto que divide la imagen original entre 2 un total de 3 veces, hasta obtener cuadrados de 2x2 píxeles.

Por otro lado, el modelo que peores resultados ha producido ha sido el 8, que trabaja con la red más sencilla. Esta tiene únicamente una capa convolucional y una de *Max-Pooling* antes de la capa totalmente conectada.

Capítulo 5

Conclusiones

EN base al desarrollo del sistema llevado a cabo en el capítulo anterior podemos concluir que los resultados obtenidos han sido positivos. Con el paso de las aproximaciones la solución ha ido mejorando hasta el punto de convertirse en un detector de aeronaves en imágenes de satélite de relativa fiabilidad. Por lo tanto, podemos afirmar que los objetivos iniciales se han cumplido con creces.

Además, una vez realizado el proyecto, hemos constatado que este tipo de sistemas son completamente viables y que pueden ahorrar mucho tiempo frente a una detección realizada de forma manual.

En lo relativo a aspectos más técnicos, el método de Aprendizaje Automático que ha ofrecido mejores resultados ha sido la técnica de Máquinas de Soporte Vectorial. A falta de un test estadístico que lo confirme, los **SVM** de kernel polinómicos y de grado entre 3 y 4 parecen ser lo que muestran mejores valores para las métricas consideradas en nuestro problema.

Por último, y más enfocado en la experiencia personal, este proyecto ha supuesto para los integrantes del grupo su primera incursión en el mundo del Aprendizaje Automático. Hemos tenido que aprender un nuevo lenguaje de programación, *Julia*; técnicas que desconocíamos, como **SVM**, Árboles de Decisión o **kNN**; o mismamente como realizar un documento para un trabajo de investigación. Finalmente podemos concluir en que la experiencia ha sido completamente satisfactoria.

Capítulo 6

Trabajo futuro

UNA vez concluido que un sistema automático que detecte aeronaves en imágenes de satélite es viable, podemos valorar como extrapolar su uso al mundo real.

El primer campo que se nos viene a la mente, y en el cual su posible empleo es más evidente, es el militar. Coordinado con la utilización de drones automáticos o semi-automáticos, un sistema como el nuestro podría permitir a un bando descubrir bases aéreas del enemigo ocultas sin la necesidad de tener recursos humanos trabajando en ello de forma activa.

Otro ámbito podría ser el de la gestión de aeropuertos. La solución podría avisar cuando un avión se encuentre en un punto de la pista del que tiene que apartarse, o para facilitar una redistribución de los aparatos.

Sin embargo, en el caso de querer hacer utilizable nuestro sistema a un nivel serio, deberían llevarse a cabo diversos cambios. Por un lado utilizar para la Base de Datos imágenes tomadas en cualquier circunstancia, y no solo de día y con buena resolución. Por otro, entrenar el sistema con modelos de aeronaves más similares a los que se enfrentaría el software en su día a día.

Lista de acrónimos

ACF Aggregate Channel Features. [7](#)

BD Base de Datos. [i](#), [3](#), [4](#)

BING Binarized Normed Gradient. [7](#)

kNN k-nearest neighbors algorithm. [2](#), [8](#), [10–12](#), [15](#), [16](#), [19](#), [23](#), [24](#), [30](#)

LUV Espacio de color CIELUV. [7](#)

NMS Nonmaximum Suppression. [7](#)

RCNN Region-based Convolutional Neural Network. [6](#)

RGB Modelo de color Red, Green, Blue. [5](#), [7](#)

RNA Red de Neuronas Artificiales. [2](#), [8](#), [9](#), [11–13](#), [15–22](#), [24](#)

SRGAN Super-Resolution Generative Adversarial Network. [6](#)

SVM Máquinas de Soporte Vectorial. [2](#), [8](#), [10–12](#), [14](#), [16](#), [18–20](#), [22](#), [24](#), [30](#)

YOLO You Only Look Once. [6](#)

Bibliografía

- [1] S. Rawat, “Airplanes detection for satellite using faster rcnn,” 2019, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://towardsdatascience.com/airplanes-detection-for-satellite-using-faster-rcnn-d307d58353f1>
- [2] R. Hammell, “Planes in satellite imagery,” consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://www.kaggle.com/datasets/rhammell/planesnet>
- [3] S. A. Mansoori, A. Kunhu, and A. A. Hammadi, “Effective airplane detection in high resolution satellite images using yolov3 model,” 2021, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9652416>
- [4] A. Tahir, M. Adil, and A. Ali, “Rapid detection of aircrafts in satellite imagery based on deep neural networks,” 2021, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://arxiv.org/pdf/2104.11677.pdf>
- [5] U. Algancı, M. Soydas, and E. Sertel, “Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images,” 2020, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://www.mdpi.com/2072-4292/12/3/458>
- [6] B. Azam, M. J. Khan, F. A. Bhatti, A. R. M. Maud, S. F. Hussain, A. J. Hashmi, and K. Khurshid, “Aircraft detection in satellite imagery using deep learning-based object detectors,” 2020, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://www.sciencedirect.com/science/article/pii/S0141933122001673>
- [7] Y. Wang, H. Wu, L. Shuai, C. Peng, and Z. Yang, “Detection of plane in remote sensing images using super-resolution,” 2022, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0265503>
- [8] H. Wu, H. Zhang, J. Zhang, and F. Xu, “Fast aircraft detection in satellite images based on convolutional neural networks,” 2015, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7351599>

- [9] A. Zhao, K. Fu, H. Sun, X. Sun, F. Li, D. Zhang, and H. Wang, “An effective method based on acf for aircraft detection in remote sensing images,” 2017, consultado el 16 de mayo de 2023. [En línea]. Disponible en: <https://ieeexplore.ieee.org/abstract/document/7888531>