

---

## MEMORIA PRÁCTICA: “Buscando a Nemo”



---

Alejandro Budíño Regueira, Manuel Couto Pintos, Juan Manuel Gómez Galeano,  
Ángela M<sup>a</sup> Martín García

Grupo 3.1 (Jueves 10:00)

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Descripción</b>	<b>3</b>
<b>3. Análisis bibliográfico</b>	<b>4</b>
<b>4. Desarrollo</b>	<b>6</b>
4.1. Aproximación 1 . . . . .	6
4.1.1. Discusión . . . . .	9
4.2. Aproximación 2 . . . . .	9
4.2.1. Discusión . . . . .	11
4.3. Aproximación 3 . . . . .	12
4.3.1. Discusión . . . . .	15
4.4. Aproximación 4 . . . . .	15
4.4.1. Discusión . . . . .	16
<b>5. Conclusiones</b>	<b>17</b>
<b>6. Trabajo futuro</b>	<b>18</b>
<b>7. Referencias</b>	<b>20</b>

## 1. Introducción

Los acuarios son instalaciones esenciales para el estudio y la divulgación de especies marinas. Estos albergan una gran diversidad de fauna y flora y se requiere un gran control sobre la evolución del número de individuos de cada especie. En el ámbito de los acuarios, esta es una tarea especialmente complicada, dado que se manejan espacios muy amplios en 3D y los peces suelen estar en movimiento constante, además existen especies con características muy similares, lo que dificulta su reconocimiento a simple vista. El reconocimiento de ciertas especies puede ser crucial para el desarrollo de un acuario, y la rapidez de dicho reconocimiento puede ser determinante en ciertas situaciones, además de reducir el tiempo de las investigaciones que se llevan a cabo.

El desarrollo de este sistema se ha centrado en el reconocimiento del *Amphirion Ocellaris*, comúnmente conocido como “pez payaso”. El objetivo es poder encontrar a los peces payaso dentro de la imagen de un acuario, con diferentes peces y objetos. La problemática asociada al caso planteado es la gran variedad de fauna y flora que comparte patrones de color con dicha especie, por tanto, debemos filtrar las características determinantes del pez payaso para poder obtener una gran precisión en nuestro sistema.

El trabajo constará de los siguientes apartados:

- Descripción, en la que se detalla el problema a resolver,
- Análisis bibliográfico, en el que se mencionarán los últimos avances en el ámbito de los sistemas de reconocimiento de imágenes.
- Desarrollo del sistema, que comprende una explicación detallada de cada aproximación y una discusión de sus resultados.
- Conclusiones, que se derivan de los resultados finales del trabajo.
- Referencias, que contiene los documentos de los que se extrajo la información para elaborar el trabajo.

## 2. Descripción

La elaboración de este trabajo se centra en la localización de peces payaso de la especie *Amphiprion Ocellaris* en imágenes de acuarios. Se trata de un problema de reconocimiento de imágenes cuyo objetivo es determinar si el pez está presente.

El trabajo ha sido abordado de modo iterativo, realizando varias aproximaciones en las que se han utilizado distintas técnicas de aprendizaje automático y se ha ido ajustando el sistema para la obtención de mejores resultados.

El proyecto se ha elaborado en el lenguaje de programación Julia language. Es un lenguaje compilado, lo que lo hace muy eficiente, con una sintaxis amigable y directa. Puede acceder a librerías de otros lenguajes. Su eficiencia, la diversidad de librerías disponibles y sus módulos para trabajar con imágenes lo convierten en una muy buena opción para esta primera aproximación.

Para resolver el problema se ha usado una base de datos de 26 imágenes de *Amphiprion Ocellaris*. No se ha encontrado un banco de imágenes en internet que satisfaga nuestras necesidades por lo que la base de datos empleada ha sido hecha a mano, buscando y recortando imágenes de peces payaso recogidas de diferentes páginas web, las imágenes son variadas, algunas hechas por ordenador, otras son fotos reales de acuarios o de entorno oceánico y la luz varía mucho entre ellas. El único criterio que se ha tomado ha sido que en la imagen el patrón de colores del pez en cuestión se mantengan en un rango similar, pese a la variación en la iluminación y que los peces se encuentran mirando en distintas direcciones.

Se ha dispuesto de tres conjuntos de imágenes:

- Un conjunto con 26 imágenes de acuarios que incluyan al menos un pez payaso común. Las imágenes del conjunto se han guardado en una carpeta denominada como “fotos”.
- Un conjunto con 41 recortes del primer conjunto de los *Amphiprion*. Las imágenes del conjunto se han guardado en una carpeta denominada como “positivos”. Las imágenes se han recortado de forma horizontal, de manera que el pez esté siempre mirando a la derecha y encajando perfectamente en los límites del recorte, sin mostrar demasiado fondo.
- Un conjunto con 60 recortes del primer conjunto de imágenes sin peces payaso. Las imágenes del conjunto se han guardado en una carpeta denominada como “negativos”. Se han incluido imágenes tanto del fondo marino, como anémonas u otros peces.

Estos conjuntos de imágenes han sido usados para entrenar la red, de forma que el conjunto de positivos recoge los patrones a reconocer, y los negativos patrones que puedan perturbar el reconocimiento efectivo de los peces payaso.

### 3. Análisis bibliográfico

A continuación se presenta una recopilación en orden cronológico de los trabajos más relevantes en el ámbito del reconocimiento de patrones en imágenes.

LeNet-5, desarrollada en 1998 por Y. LeCun, L. Bottou, Y. Bengio y P. Haffner fue una de las primeras arquitecturas de redes neuronales convolucionales propuestas, que pretendían reconocer caracteres escritos a mano y a máquina. Estas suelen emplearse como primera aproximación académica a las CNN (Convolutional Neural Network) por su arquitectura sencilla y fácil de entender. [1, 2]

AlexNet, desarrollada en 2012 por A. Krizhevsky es similar a LeNet pero más grande y profunda, fué capaz de obtener un error del 17% en el ILSVRC (ImageNet Large Scale Visual Recognition Competition) de 2012 frente al segundo clasificado que obtuvo un error del 26 %. Fue el primer modelo en apilar capas convolucionales una sobre otra en lugar de capas de pooling. [3]

GoogLeNet fue desarrollada por C. Szegedy en 2014 a través de Google Research. Ganó el ILSVRC 2014 bajando el ranking top-5 por debajo del 7%. El hecho es que se trata de una red mucho más profunda que sus predecesoras y se basa en el uso de sub-redes llamadas inception modules “Nombre sacado de la película Inception”, esto permite a la red usar parámetros de forma más eficiente que sus predecesoras, en concreto 10 veces menos parámetros que AlexNet ResNet 2015. [4]

Fully Convolutional Network 2015 Devuelve una imagen del mismo tamaño que la original, cada píxel es coloreado de un color C el cual C es el número de la clase que representa.

La idea principal de esta red es que todas sus capas son convolucionales, no tiene ninguna capa densa al final usada frecuentemente para clasificación, así que el resultado final en una clasificación pixel a pixel y si usamos una función softmax obtendremos la clase más probable. [5]

Para solucionar el problema de seleccionar un gran número de regiones de la imagen a analizar, que suele realizarse barriendo secuencialmente la imagen con cuadros de distinto tamaño, R. Girshick et al. proponen un método de búsqueda selectiva para extraer tan solo 2000 regiones candidatas en la imagen. [6] A estas redes se les denominó R-CNN (Region-CNN). La selección se hace mediante un algoritmo que realiza un subsegmentación inicial de imagen y combina recursivamente aquellas más afines para producir la mejor selección de regiones. Esta selección de regiones se le pasa a la red neuronal convolucional que extrae las características de las regiones. Por último este vector resultado se pasa a una SVM (Support Vector Machine) para clasificar los objetos presentes en las regiones. Además también se generan valores de redimensionado que potencialmente podrían aumentar la precisión de la región. Este método tiene el inconveniente de que sigue consumiendo una gran cantidad de tiempo debido al elevado número de regiones candidatas.

Las Faster R-CNN son una mejora de las R-CNN que el lugar de usar un algoritmo de segmentación clásico transfieren directamente la imagen de entrada a una red neuronal convolucional para generar un mapa convolucional de características. En este mapa se identifican las regiones candidatas, se envuelven en cuadrados y se redimensionan a un tamaño fijo mediante una capa de pooling RoI (Region of Interest) para pasárlas a una capa completamente conectada. Del vector de características RoI se pasa a una capa softmax para predecir la clase de la región propuesta y nuevos offsets para las regiones envolventes.

SENet, propuestas en 2017 por H. Jie et al. [7, 8] Su nombre viene de Squeeze-and-Excitation

Networks, fue desarrollada en la University of Oxford y ha sido la ganadora de la competición ILSVRC 2017 consiguiendo obtener un error del 2.251 % bajando un 20 % respecto de los ganadores del año anterior. Se propone una operación de excitación para capturar por completo las dependencias de canal y para aprender una relación no lineal y no mutuamente excluyente entre canales.

YoLo, abreviatura de You Only Look Once, su versión V1 fue desarrollada por J. Redmon en 2015, posteriormente se publicaron las versiones V2 y V3 en 2016 y 2018.[9] Enfoca el problema de reconocimiento de imagen de una forma novedosa, en vez de afrontarlo como un problema de clasificación lo hace como un problema de regresión. Se trata de una única red que detecta las bounding boxes y asigna porcentajes de una sola vez por lo que funciona muy bien para reconocer imágenes en tiempo real. Actualmente las redes YoLo son el desarrollo más prometedor en lo que respecta a la clasificación de patrones en imágenes.

Los principales desarrollos aquí presentados corresponden a sistemas avanzados de reconocimiento pensados para ser usados en casos generales. En nuestro caso se desarrollará un sistema para un caso específico basado en técnicas de aprendizaje más primitivas y con un nivel de refinamiento menor. Mientras que en los últimos tiempos han aparecido sistemas cuyo input para la clasificación es la imagen completa y las regiones de interés las determina la red en nuestro sistema habrá que barrer la imagen extrayendo secuencialmente subsecciones que se pasan a la red para su clasificación.

## 4. Desarrollo

El desarrollo del sistema se abordará de forma iterativa. Se realizarán cuatro aproximaciones usando técnicas distintas y se analizarán los resultados obtenidos por cada una. Las tecnologías en las que se basará serán RNA (Red de Neuronas Artificiales), SVP (Support Vector Machine), kNN (k-Nearest Neighbours) y CNN.

### 4.1. Aproximación 1

En una primera aproximación se ha usado una red neuronal de tipo MLP (MultiLayer Perceptron) para realizar la clasificación. Se determina que el vector de atributos que se le pasará a la red contendrá los datos de media y desviación típica para cada componente de color del espacio RGB (Red Green Blue) calculados para cada recorte de nuestra base de datos de imágenes. Por tanto nuestra red neuronal tendrá una capa de entrada con 6 entradas y una capa de salida con una única salida, ya que la red solo determinará si la imagen que le pasamos contiene o no un pez payaso. La red tiene conectividad densa entre capas y la misma función de transferencia para todas las capas.

Se ha entrenado la red mediante un aprendizaje supervisado, el dataset que se ha usado contiene un grupo de positivos y otro de negativos que se han mezclado y dividido en conjuntos de entrenamiento, validación y test a los que se han asignado unos porcentajes del 60 %, 20 % y 20 % respectivamente de los 101 recortes que componen el dataset.

En la fase de entrenamiento se generan múltiples modelos inicializados con selecciones aleatorias de la distribución de los patrones y se escoge como mejor modelo aquel que tiene mayor tasa de aciertos en el conjunto de test.

Para determinar el número adecuado de neuronas que deben llevar las capas ocultas se ha entrenado el modelo con distintas configuraciones y se ha analizado su eficiencia. Los resultados obtenidos en test para distintas arquitecturas pueden verse en el Cuadro 1.

Capa 1\Capa 2	3	4	5	6	7	8	9	10
3	76.5	67.5	73	74.5	68	69	73.5	66.5
4	67	80	66.5	72	72	71.5	71	71.5
5	75.5	71	71	69	72.5	74	74	72
6	71	79	75.5	74	76	78	72	73.5
7	72.5	77	68	68	73	78	74.5	72.5
8	75	75	65	67	75	72	71	66.5
9	71	74	75	70	68.5	70	78	68
10	74	74.5	76	71	73.5	70.5	73	73.5

Cuadro 1: Resultados por arquitectura

Puede verse que existen varias configuraciones con porcentajes de acierto similares, así que en esta ocasión se ha seleccionado la arquitectura que tiene 6 neuronas en la primera capa oculta y 4 en la segunda. La arquitectura seleccionada para la red puede verse en la Figura 1.

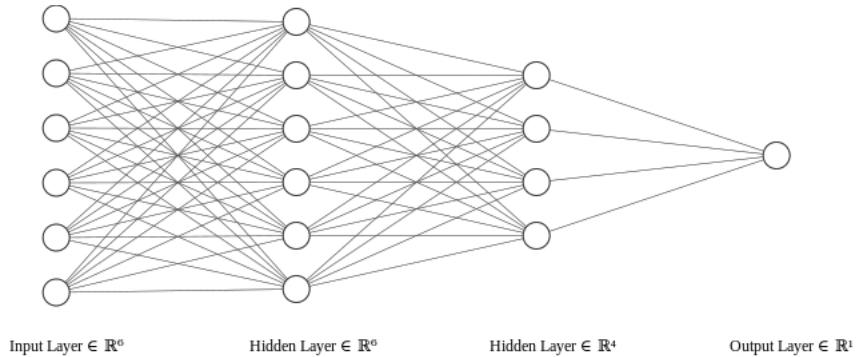


Figura 1: Arquitectura de la red

El modelo generado tiene un índice de aciertos del 81 % en el conjunto de validación y un 79 % en el de test con una desviación típica similar, como puede verse en el Cuadro 2. También hay que recordar que la base de datos que se está usando es bastante pequeña y las cifras obtenidas pueden sufrir variaciones de hasta un 5 % entre ejecuciones del entrenamiento.

	Precisión	Desviación típica
Entrenamiento	81 %	8.5
Validación	83 %	7.3
Test	79 %	7.5

Cuadro 2: Resultado del entrenamiento

A continuación hemos probado la red entrenada en diferentes imágenes, para esta etapa se ha intentado hacer recorriendo la imagen de forma iterativa repetidas veces con diferentes tamaños de ventana y con diferentes puntos de partida del iterador, de esta forma se ha podido marcar con recuadros de diferente tamaño la zonas que eran validadas por la red por encima de un umbral de confianza. En la Figura 2 podemos ver como las ventanas que indican una detección engloban solo parte del pez payaso que se sitúa de frente a la cámara. Aunque el sistema fue entrenado con patrones que solo mostraban a los peces de perfil el sistema detecta la presencia de este basándose en la distribución de colores del recuadro.



Figura 2: Detección sin máscara

El problema del método iterativo es que se precisa procesar gran cantidad de ventanas que sabemos seguro que no son pez payaso, como por ejemplo los fondos de las imágenes, por lo que intentamos procesar la imagen para que en base a la morfología y los colores seleccionara ventanas de regiones de interés. Para esto creamos una máscara binaria, que se muestra en la Figura 3, y es el resultado de filtrar solo el canal rojo de la imagen.

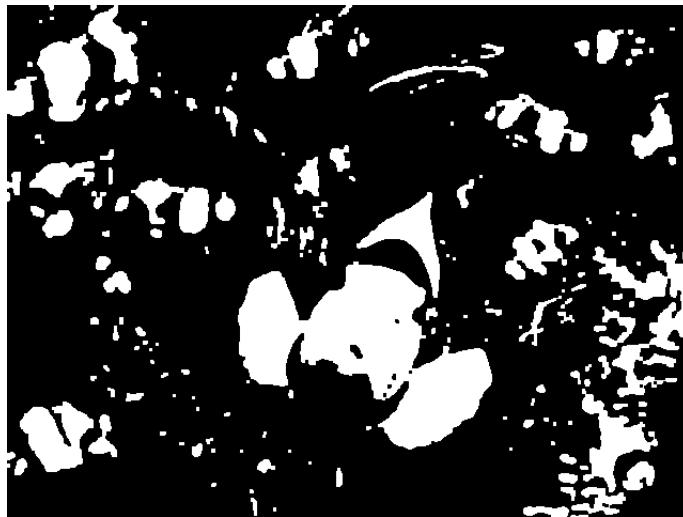


Figura 3: Máscara binaria

El resultado de usar la máscara, como se muestra en la Figura 4, son unas ventanas más certeras, que se centran en zonas más interesantes, y como se puede ver detecta parte de los peces payaso, especialmente aquellos que están más contrastados con el fondo.



Figura 4: Detección con máscara

#### 4.1.1. Discusión

El mejor modelo generado con esta aproximación da los resultados que se muestran en la matriz de confusión del Cuadro 2. Podemos ver que el índice de aciertos es aceptablemente bueno para tratarse de un primer intento. De esos datos también se extrae que este sistema tiende a generar mas falsos positivos que falsos negativos, debido a que los atributos basados en la distribución de color en la imagen no tienen en cuenta su distribución espacial y por ello tiende a producir detecciones positivas en otros peces de colores similares o incluso con el fondo.

		Resultado	
		Positivos	Negativos
Esperado	Positivos	6	2
	Negativos	4	8

Cuadro 3: Matriz de confusión

En la siguiente aproximación se intentará mejorar el resultado empleando como clasificador una MSV (Máquina de Soporte Vectorial).

#### 4.2. Aproximación 2

En esta segunda aproximación hemos usado la misma metodología para extraer las características de las imágenes, pero a diferencia de la anterior aproximación hemos usado una MSV que aborda el problema de clasificación de una forma un poco diferente que las redes neuronales artificiales.

El algoritmo de las máquinas de soporte vectorial tiene dos partes importantes que hemos tenido en cuenta para comparar las diferentes pruebas de entrenamiento, la primera es la tolerancia de error C. Cuando se están buscando los vectores de soporte esta variable permite que se ignore parte del ruido que pueda haber en los patrones de entrenamiento, permite que se generalice mejor y no se sobreajuste tanto. La otra parte importante del algoritmo es la función de kernel. Estas funciones permiten representar los patrones de las clases en un espacio de mayor dimensionalidad para de esta forma poder trazar hiperplanos que dividan las regiones adecuadamente.

Existen varias funciones que se pueden usar dependiendo del tipo de problema como linear, rbf y poly.

- Linear tiene un único parámetro C que ajusta los márgenes del hiperplano para clasificar correctamente la mayor cantidad de patrones posibles. Tras probar con varios valores se comprueba que con 14 se consigue un porcentaje de aciertos en test del 78 %.
- Poly emplea los parámetros C, que aumenta la sensibilidad del sistema y degree, que indica el grado del polinomio. Al probar con combinaciones de parámetros vemos que alcanza valores para test del 79 % con grado 8 y C=20.
- RBF se ajusta con gamma, que aumenta la sensibilidad del sistema y C. Este alcanza los mejores resultados con gamma=3 y C=8 con una tasa de acierto en test del 82 %.

Tras probar estos tres tipos para entrenar el modelo se ha comprobado que linear y poly tienen una precisión de aproximadamente un 79 % en test mientras que RBF muestra consistentemente una precisión del 82 %.

A partir de esta clasificación pasamos a comprobar el comportamiento del sistema en las imágenes reales.

En este primer ejemplo hemos usado kernel lineal con C = 14. En la Figura 5 observamos que está clasificando mal la anémona de la derecha y el pez de la izquierda. Aun teniendo algunos errores muestra una precisión bastante buena a la hora de localizar los peces payaso teniendo en cuenta que están sobre un fondo que se puede confundir con su piel.



Figura 5: Detección con kernel lineal

En el siguiente ejemplo se ha empleado el kernel polinómico, cuyo resultado se puede ver en la Figura 6. Vemos que el sistema funciona de una forma muy similar al kernel lineal, esto puede deberse a que los datos de entrada tienen una dimensionalidad muy baja y no hay una mejora apreciable en la segmentación.



Figura 6: Detección con kernel polinómico

En el último ejemplo se ha empleado un kernel RBF. En el ejemplo real clasifica bien casi todas las ventanas en las que se ha probado, en la Figura 7 podemos ver que el pez de la izquierda está correctamente clasificado, sin embargo, se ha clasificado incorrectamente la anémona de la derecha.



Figura 7: Detección con kernel RBF

En resumen, vemos que los 3 kernel probados tienen resultados muy parecidos. Si bien en otros problemas escoger uno u otro puede ser determinante para conseguir un buen resultado la dimensionalidad de los datos empleado provoca que los resultados sean muy parecidos.

#### 4.2.1. Discusión

Resulta difícil saber si el sistema está sobreajustando o no, dado que hay imágenes que tienen proporciones parecidas de canales RGB a las de los peces payaso. Además, interfieren como ruido características que no nos interesan, como son la posición del pez o la iluminación y se

ignoran otras como la morfología de la imagen, pero en este ejemplo la bondad del sistema ha mejorado a medida que incrementamos los valores de las variables gamma y C, que lo que hacen es definir un hiperplano más complejo y tener más en cuenta los patrones que podrían ser ruido. Por otro lado se puede observar que el método que se emplea para seleccionar los recuadros a comprobar de las fotos puede estar limitando las posibilidades del sistema. En las fotografías nos percatamos de que los recuadros a comprobar siempre son los mismos, puesto que este método no cambia con respecto al clasificador que se emplee. Aunque esa característica en parte permite que se puedan comparar mejor las clasificaciones entre distintas arquitecturas si el recuadro no es lo bastante preciso siempre se le estará pasando un marco inadecuado al sistema. Esta es una de las mayores dificultades que se han encontrado en el desarrollo y el método escogido como solución supone un compromiso que podemos considerar aceptable entre tiempo de ejecución y efectividad.

Una posible mejora a la hora de extraer las características de la imagen puede ser la técnica HOG (Histogram of Oriented Gradient), que permite obviar características que no nos interesan como son la posición, la iluminación, etc. Extraen características más relevantes. El problema que tenemos es que, para nuestro caso concreto, en el que tenemos una base de datos muy reducida, esta técnica extrae un número muy elevado de características que hace que nuestro sistema se sobreentrene y que dé resultados muy malos en el conjunto de Test.

A continuación, utilizaríamos una k-NN para intentar mejorar la clasificación.

### 4.3. Aproximación 3

Las primeras pruebas con k-NN consistieron en usar las características de la aproximaciones anteriores con el método de los K vecinos más cercanos. Se usó una función para encontrar el número de vecinos que conseguía un mejor porcentaje de aciertos en test. Sin embargo este método no daba buenos resultados, ya que o bien clasificaba todas las imágenes como negativos o como positivos según se modificaban parámetros del algoritmo.

Como el método para buscar los peces consiste en hallar ventanas con proporciones altas de rojos y k-NN guarda como ejemplos de peces payaso los que tienen estas características, se supuso que las características anteriores eran inadecuadas ya que k-NN no separa con hiperplanos, si no que busca por cercanía de vecinos. Los patrones de los positivos de peces payaso son mayoritariamente rojos, y dado que las ventanas que escogemos para la fase de identificación en las fotografías completas se basa en la cantidad de rojo que contiene la ventana que se considera más probable, el sistema tiende a clasificar todos los patrones como positivos.

De esta manera, se consideró necesario ser más específicos con el tono de rojos que representaba a la clase peces payaso. Durante la etapa de extracción de las características de la imagen se pasa el mismo filtro de rojos que usamos para hallar las ventanas susceptibles de contener peces payaso. A continuación, de las regiones que tienen mayores proporciones de rojos se obtuvo el valor más predominante mediante quartiles 0.75 y 0.95. Además, se realiza el mismo proceso para extraer los tonos blancos. De esta forma, ahora se trabaja con 12 características. Con las nuevas características ahora se consigue un porcentaje de acierto del 88 % en entrenamiento y un 82 % en test con 7 vecinos. En la figura 8 se puede ver la salida del detector con el nuevo método. En esta imagen identifica correctamente los 3 peces que aparecen.



Figura 8: Detección correcta

Pero en la figura 9 se puede ver que, aunque identifica correctamente el pez entre los demás, aparecen dos falsos positivos en un coral y un pez que comparten una tonalidad muy similar.



Figura 9: Detección incorrecta

En la figura 10 se puede ver que el sistema tiene una gran especificidad, ya que no produce falsos positivos en el caso de la anémona y el coral e identifica correctamente los dos peces payaso que aparecen a pesar de que uno de ellos esta de espaldas en la imagen.

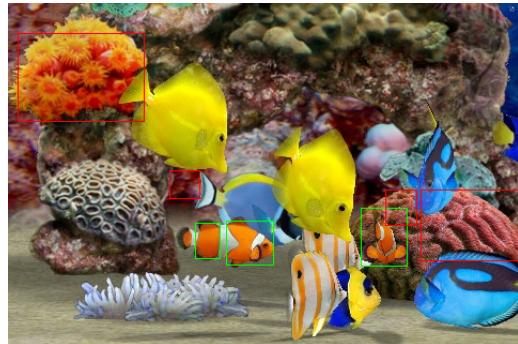


Figura 10: Detección correcta

Se ha intentado probar también otros algoritmos distintos a knn.

El radius clasifica según el resultado de la votación de los vecinos dentro de un radio. Sin embargo ese algoritmo da unos resultados bastante malos en el entrenamiento con un valor de 56 % en test.

### Redes Neuronales

Una vez comprobada la eficacia de las nuevas características para la técnica KNN se comprueba si también mejoraban de alguna manera la eficacia de las anteriores. Con RNA hubo que cambiar la arquitectura para adaptarse al nuevo número de entradas, pasando a una 12:7:5:1, con la cual obtuvimos como mejor promedio de porcentajes de validación y test de 80.7 % y 79.8 % respectivamente. Vemos que no supone una mejora significativa, si comparamos con los resultados de la imagen que usamos como referencia en la primera aproximación vemos que tampoco supone ninguna mejora.



Figura 11: Detección correcta

### SVM

Pasamos ahora a comprobar si con las SVM obtenemos mejores resultados. Volvemos a probar los mismos kernels que en la segunda aproximación. Con kernel lineal obtenemos un modelo con unos porcentajes de validación y test de 88.9 % y 84.5 % respectivamente, la imagen de referencia nos devuelve un resultado igual a la aproximación 2. Con el kernel polinómico obtenemos unos resultados de entrenamiento y test de 89.29 % y 86.16 %, sin embargo en la práctica ambas tienden a tener más sensibilidad que especificidad. Con el kernel gausiano los porcentajes de validación y test fueron 99.586 % y 82.166 % pero en las imágenes nuevas se apreció una muy ligera mejora en especificidad.

#### 4.3.1. Discusión

Los resultados en entrenamiento y test con k-NN son parecidos a los que se obtenían con la aproximación anterior en RNN y SVM, sin embargo la detección de peces en las imágenes empeora ligeramente.

Aún con la selección de nuevas características, que mejoran la eficacia en k-NN, los resultados no son mejores que con los anteriores métodos.

Los algoritmos como k-NN son más rápidos en la fase de entrenamiento ya que realizan el trabajo durante la clasificación de nuevos patrones. Sin embargo solo son efectivos si las clases están bien separadas y la dimensionalidad no es demasiado alta.

### 4.4. Aproximación 4

Para la última aproximación se ha empleado una técnica basada en redes neuronales profundas, una red neuronal convolucional. La principal diferencia con respecto al resto de técnicas es que con esta no es necesario extraer características manualmente, se pasan un conjunto significativo de los píxeles de la imagen a través de las capas de la red, que está formada por capas superpuestas de convoluciones y poolings, en las cuales se transforman y se extraen nuevas características hasta una capa final de neuronas que devuelven una probabilidad de que clase es el patrón en cada caso. En nuestro dominio hemos usado 3 convoluciones, 3 poolings y una última capa densa. En este caso los resultados no han sido tan satisfactorios como con las otras técnicas. Aunque en la fase de test se consiguen valores del 90 % a la hora de identificar los peces en las imágenes reales el sistema no es capaz de reconocer la mayoría de las imágenes que antes si podía. Sin embargo la diferencia más llamativa con respecto a anteriores resultados es que ya no se produce ningún falso positivo, el sistema solo identifica como peces payaso aquellos que realmente lo son, y coincide que aquellos que son mejor clasificados son los que tienen una ventana más acertada, que encuadra bien a pez payaso. La dificultad de probar este sistema con la imagen real es que mientras los demás se basaban en características de los colores este identifica también la morfología. Esto hace que si no se le pasan ventanas que contengan toda la figura del pez lo más probable es que no sea capaz de reconocerlo. En la figura 11 podemos ver el problema en la imagen más característica de la colección. Cuando el pez no está bien encuadrado o aparece en una posición distinta a como está en los recortes de entrenamiento no se detecta como positivo.



Figura 12: Detección por deep learning

También hemos intentado mejorar los resultados añadiendo los falsos positivos a la base de datos de negativos y los falsos negativos a la base de datos de positivos, pero nos hemos encontrado que los resultados empeoran.

#### 4.4.1. Discusión

La aplicación de técnicas de deep learning a este tipo de aplicaciones suele tener buenos resultados pero en este caso no han superado los de aproximaciones más clásicas. Si bien los valores de entrenamiento obtenidos han sido los mejores, el sistema no tiene un buen comportamiento con las imágenes reales debido a la dificultad para obtener ventanas que encuadren bien a los peces objetivo y debido a la falta de variedad de patrones para tratar las diferentes variaciones de luminosidad, tonalidad y posición de los peces.

## 5. Conclusiones

La primera dificultad surgida a la hora de identificar los peces es decidir que ventanas se le pasan al sistema entrenado. Este es un problema que ha tardado mucho tiempo en resolverse en el campo de la IA y uno de los mayores limitantes de su rendimiento. Las redes de tipo YoLo señalan directamente los objetos en la imagen sin recortar, pero en este caso se ha tenido que decidir que fragmentos probar. El resultado de esta heurística es aceptable en el marco de los sistemas más clásicos como son el perceptrón, SVM y kNN. En estos no es necesario que el pez esté bien encuadrado porque las características que extraemos son de composición de color y no morfológicas. Si bien es cierto que nada impide extraer características morfológicas los patrones que generan tienen una dimensionalidad que los hace imprácticos dado el reducido tamaño de la base de datos empleada. Dicho esto podemos concluir que, con respecto a la simplicidad de la técnica de extracción, esta resulta en una ventaja, ya que los resultados son aceptables incluso cuando el método para extraer los fragmentos no es el mejor.

Con respecto a kNN la técnica no parece la más adecuada para clasificar imágenes con el método que solo extrae características de color ya que los patrones se localizan en un espacio muy reducido. Por esto se ha tenido que afinar un poco más en la característica concreta que color a extraer.

En cuanto a las redes convolucionales los resultados no han sido los mejores por las limitaciones mencionadas anteriormente, aun así, se ha visto la alta especificidad que se obtiene siendo capaz de detectar algunos peces payaso y tiene un gran potencial si seguimos trabajando con ellas.

En el momento presente no se puede decir que con los resultados obtenidos este sistema pueda ser comercialmente viable ya su efectividad a la hora de reconocer las imágenes está muy condicionada a la iluminación, resolución, posición concretas en que se encuentra el pez y el encuadre del pez en la ventana. Sin embargo si ha servido para tener una perspectiva de como funcionan los sistemas de redes neuronales profundas.

## 6. Trabajo futuro

En este proyecto hemos podido comprobar las virtudes de algunas de las técnicas de aprendizaje automático más conocidas, pero nos hemos encontrado con ciertas limitaciones a la hora de llevar la calidad del proyecto más allá. Los dos problemas principales que hemos detectado son, la falta de patrones de ejemplo y la baja calidad de las ventanas que se obtienen con el filtro de color. El primero de los problemas tiene una fácil pero tediosa solución, que es dedicar tiempo en internet para completar la base de datos con suficiente variedad de ejemplos. El segundo de los problemas tiene diferentes soluciones, el primero consiste en utilizar algoritmos de extracción de segmentos de la imagen, de esta forma, podríamos obtener el contorno del pez y así lograr una mejor ventana. Ya se ha trabajado algo en esta dirección.



Figura 13: Segmentos de una imagen

En esta primera imagen se ha usado la función *felzenszwalb* de la librería cv de julia para poder obtener los segmentos de la imagen, luego se crean etiquetas para cada segmento y se eliminan los demasiado grandes y demasiado pequeños, aunque el comportamiento de esto ha resultado algo extraño. El resultado han sido las siguientes ventanas.

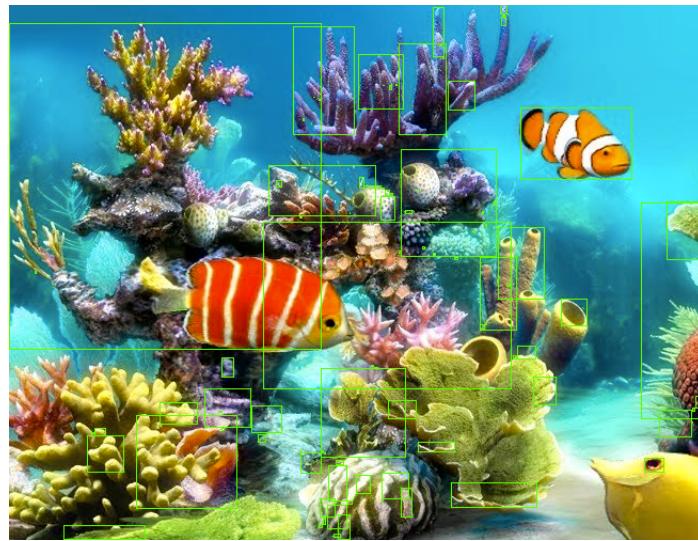


Figura 14: Detección por deep learning

Como se puede ver el pez payaso ha quedado perfectamente encuadrado. Algunas arquitecturas de deep learning logran crear un mapa semántico de la imagen que localiza cada objeto de la imagen y su posición en la misma, esto claramente daría solución a nuestro problema, nos permitiría dar con una solución viable y es una línea de trabajo para el futuro muy interesante.

## 7. Referencias

- [1] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE, 86(11):2278 - 2324. Recuperado de <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-3029>
- [2] Rizwan, M. (2018). *LeNet-5 – A Classic CNN Architecture*. engMRK Machine learning using python. Recuperado en 5 Mar 2020 de <https://engmrk.com/lenet-5-a-classic-cnn-architecture/>
- [3] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet Classification with Deep Convolutional-Neural Networks. NIPS 4828. Recuperado de <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, D. & Rabinovich, A. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*. Recuperado de [https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2015/papers/Szegedy\\_Going\\_Deeper\\_With\\_2015\\_CVPR.pdf](https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR.pdf)
- [5] Long, J., Shelhamer, E. & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA*, pp. 3431-3440. Recuperado de <https://arxiv.org/pdf/1411.4038.pdf>
- [6] Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH*, pp. 580-587. Recuperado de <https://arxiv.org/pdf/1311.2524.pdf>
- [7] Jie, H., Li, S., Albanie, S., Gang, S. & Enhua, W. (2018). Squeeze-and-Excitation Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT*, pp. 7132-7141. Recuperado de <https://arxiv.org/pdf/1709.01507.pdf>
- [8] Tsang, S. (2019). Review: SENet — Squeeze-and-Excitation Network, Winner of ILSVRC 2017 (Image Classification). *Medium*. Recuperado en 5 Mar 2020 de <https://towardsdatascience.com/review-senet-squeeze-and-excitation-network-winner-of-ilsvrc-2017-image-classification-a887b98b2883>
- [9] Redmon, J & Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV*, 2016, pp. 779-788. Recuperado de <https://arxiv.org/pdf/1506.02640.pdf>