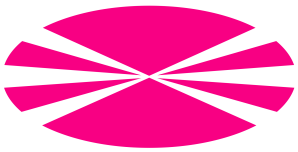


UNIVERSIDADE
DA CORUÑA

Aprendizaje Automático

Memoria de la práctica

Sistema de detección de caras de Papá Noel

Autores:

Óscar Alejandro Manteiga Seoane

Breogán Fernández Moreira

Martín Castro Fernández

Nicolás Vázquez Cancela

Arturo Ramos Rey

Índice

Índice de figuras

Índice de Tablas

1. Resumen.

En esta Memoria se describirán los pasos para la resolución de un problema de detección de la cara de Papá Noel en fotos. En una primera instancia se hará una introducción, con consideraciones y definiciones previas que ayudarán a entender el documento. Posteriormente se explicará el problema con más detalle, separando por un lado las cuestiones asociadas y por otro la tarea a resolver. También mostraremos ventajas y desventajas de la resolución del problema y los objetivos que se esperan alcanzar.

(a) Imagen con Papá Noel

(b) Imagen sin Papá Noel

Figura 1: Imágenes que puede tratar el sistema

Después de esto se relatará una descripción detallada del problema, explicando cada detalle de la Base de datos, las restricciones incorporadas, como serán las fotos que pueden introducirse al sistema, origen de los datos y su descripción...

Sumado a esto, realizaremos un análisis bibliográfico en donde haremos un recorrido a través de los artículos relacionados con la tarea. Buscaremos explicarlos de forma resumida, relacionarlos con nuestro problema y obtener información que nos pueda resultar de ayuda para la conquista de nuestro objetivo.

A continuación narraremos todo el proceso de desarrollo del sistema, explicando las iteraciones por las que ha pasado, cómo funciona por dentro, las evoluciones entre cada uno y los resultados obtenidos. Haremos también una discusión de los obtenido a modo de resumen y de resaltar lo más importante.

A mayores tendremos un apartado de conclusiones, donde vamos a especificar tanto lo que hemos aprendido a lo largo del proceso, como lo que creemos que se puede mejorar o no, los resultados y posibles extrapolaciones a otras tareas en las que pueda ser de utilidad.

Finalmente hablaremos de trabajo futuro que pueda quedar por hacer en el proyecto, como mejoras y nuevas incorporaciones, mostraremos la bibliografía y un glosario tanto de términos como de acrónimos utilizados (con el fin de que todo se entienda sin un conocimiento elevado del tema).

2. Introducción.

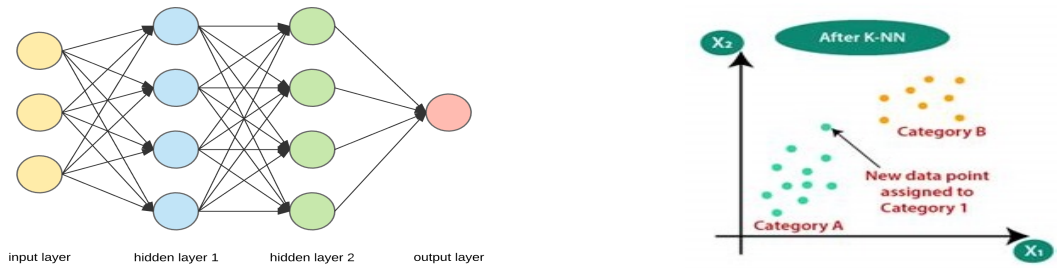
2.1. Consideraciones previas.

Antes de poder introducir el problema a resolver, es necesario entender unos conceptos claves. El aprendizaje automático (AA), que en resumidas cuentas es una subcategoría de la inteligencia artificial (IA), es un proceso de creación de modelos analíticos que permiten que las máquinas se adapten a nuevas situaciones. Es decir, analiza los datos y crea patrones para realizar una tarea.

Dentro de ella existen varios tipos de aprendizaje. En primer lugar tenemos al aprendizaje supervisado, en el que el comportamiento deseado lo representamos por un conjunto de ejemplos que servirán a mayores para evaluar el sistema. El siguiente tipo será el aprendizaje no supervisado, en el que no tenemos ningún tipo de señal que indique un comportamiento deseado para el sistema y su evaluación se basa en la regularidad de grupos de datos identificados. Finalmente tenemos el aprendizaje por refuerzo, en el que el comportamiento deseado no se representa mediante ejemplos sino mediante una cierta evaluación sobre los resultados que genera el sistema en su entorno.

A la hora de representar el conocimiento en estos tipos de aprendizaje automático, es importante establecer el lenguaje para codificar dicho conocimiento en ellos. Para esto disponemos de redes semánticas, árboles y/o reglas, sistemas difusos, redes de neuronas artificiales, máquinas de soporte vectorial... Estos 2 últimos son vitales, por lo que es necesario entenderlos. Las Redes de Neuronas Artificiales, o ANN, por sus siglas en inglés, son modelos computacionales que buscan obtener el conocimiento de forma similar a las células cerebrales humanas. Dentro de ellas tendremos varias capas interconectadas (2a) y, usándolas, podemos hacer que aprendan observando data sets de entrenamiento para que en un futuro puedan realizar la tarea asociada con datos que nunca han visto. Las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés) son modelos de aprendizaje supervisado con algoritmos de aprendizaje asociados que analizan datos para clasificación y análisis de regresión.

A mayores también es importante entender KNN, que es una técnica y algoritmo de aprendizaje automático que se puede utilizar tanto para tareas de regresión como de clasificación. De forma resumida, este algoritmo clasifica puntos por cercanía/similaridad en un plano (2b).



(a) ANN layers (b) kNN example

Figura 2: Capas de una RNA y ejemplo de kNN

2.2. Problemática asociada.

El detectar el rostro de Papá Noel, nos ofrece el reto de distinguir en un primer paso lo que es o no una cara, para a continuación determinar si se trata o no de este personaje navideño. En una primera instancia, no puede parecer un problema de gran importancia, pero las soluciones de este tipo de problemas se pueden usar para otros ámbitos más relevantes.

Con otras Bases de Datos, en las que contemos con fotos de personas con y sin mascarilla, por poner un ejemplo, podríamos detectar si se lleva puesta o no para vigilar que todo el mundo cumple la normativa en un lugar concreto. Como se puede ver, detectar rostros y/o elementos en ellos, es un problema muy útil en muchos aspectos de la vida. Con la detección de la cara de Santa Claus buscamos solucionarlo con ciertos aspectos divertidos que puedan atraer la atención de personas ajenas al ámbito del aprendizaje automático.

2.3. Problema a resolver.

El problema a resolver será la detección de la cara de Papá Noel en imágenes. Como hemos mencionado, puede parecer un problema absurdo, pero recordemos que con cambiar la Base de Datos podemos hacer sistemas de gran ayuda. Las fotos tienen que contener una persona para que sean aceptadas, si así es el caso, en base a como sea su cara será un caso positivo o negativo en la clasificación. Usaremos como características de este personaje su gran barba blanca y larga y su gorro rojo con un pompón blanco. Para eso sacaremos las características de la imagen por zonas obteniendo los colores de las mismas, datos de los que el sistema clasificará el input.

2.4. Ventajas y desventajas.

Como ventajas de solucionar el problema tenemos en primer lugar, el traslado de la misma a ámbitos más relevantes como el mencionado en la sección anterior. Detectar caras es algo importante, pero muchas veces saber las características de las mismas o lo que llevan puesto es algo que nos abre muchos campos de aplicación. Otro elemento a tener en cuenta sería el acercamiento del Aprendizaje Automático a personas ajenas al campo e incluso, enseñar a niños y niñas cómo funciona para que puedan en un futuro llegar a ser los investigadores del mañana.

2.5. Objetivos del trabajo.

Los objetivos que tendremos serán reconocer de forma efectiva el rostro de Papá Noel, no dar como positivos imágenes en las que no aparezcan personas parecidas a este personaje y evitar que el error sea elevado. Esto sobre todo se puede dar en fotos donde aparezcan rostros con solo algún elemento característico del personaje u otras personas parecidas, como los Reyes Magos. Buscaremos en un principio que detecte a Papá Noel de forma más precisa pese a que pueda dar falsos positivos, para en un futuro intentar ajustarlo lo máximo posible.

3. Descripción del problema.

3.1. Descripción concreta del problema.

Para establecer y concretar el problema a solucionar, se tiene que aclarar que lo que se quiere hacer es: dada una imagen, al pasar por el sistema realizado, que este nos proporcione un resultado positivo si aparece la cara de Papá Noel; o negativo si no se encuentra en la foto. A la hora de establecer qué métrica usaremos para fijar si un resultado es bueno o para comparar entre varios modelos o algoritmos, primero debemos conocer los que valoramos como posibles candidatos:

- Precisión: mide la cantidad de veces que dado un positivo este es cierto.
- Tasa de errores: relación de la cantidad de elementos clasificados erróneamente con respecto a los elementos totales
- Sensibilidad: probabilidad de clasificar como positivos los elementos que efectivamente son positivos
- Especificidad: probabilidad de clasificar como negativo los elementos que efectivamente son negativos
- F1: es el valor de la media harmónica entre sensibilidad y especificidad

Nosotros nos quedamos con la precisión y el f1 score, ya que nos interesa que nuestro sistema obtenga todos los positivos, pero minimizando la cantidad de falsos positivos así como la de falsos negativos, y al combinar estas dos métricas podemos determinar si se comporta de esta manera.

3.2. Restricciones que se aplican al problema.

A la hora de ingresar una foto en el sistema, esta tendrá que estar recortada conteniendo únicamente la cara, gorro y barba en vista frontal del sujeto a probarse si es o no Papá Noel, y con un fondo que evite los colores rojos y blancos. Tenga o no la imagen a este personaje, es necesario que si se trate de un rostro de una persona de frente. A mayores, las imágenes no pueden ser en blanco y negro, ya que nuestro sistema lo que obtiene son los colores de la misma por zonas. El formato puede ser PNG o JPEG. La Base de datos que tenemos, como explicaremos en la siguiente sección, no contiene las imágenes ya recortadas. Esto implica que por cada imagen tendremos que recortar la forma de la cara de la persona y establecerla en un fondo que no sea ni blanco ni rojo a ser posible. Finalmente, todas las imágenes que no cumplan con las restricciones, serán eliminadas de la misma.

(a) Imagen original (b) Cara recortada (c) Cambio de fondo

Figura 3: Modificaciones de las imágenes de la BBDD

3.3. Descripción de la Base de Datos.

3.3.1. Origen.

Encontramos la Base de Datos en la página Kaggle (`https://www.kaggle.com/deepcontractor/is-that-santa-image-classification`), aunque debido a la mala calidad de algunas imágenes o por no cumplir las restricciones del problema mencionados, se ha tenido que filtrar o editar cada imagen de la Base de Datos para poder usarla.

Nuestra BBDD se compone de **392** imágenes de entrenamiento, de las cuales **85** son caras de Santa Claus y **307** son caras de personas aleatorias. Para *testear* nuestro modelo tenemos **507** imágenes, **201** de Papá Noel y **306** de otras personas. A la hora de incorporarlas al sistema, las juntaremos todas (una vez estén filtradas y editadas según las restricciones), ya que en el propio programa establecemos la división para entrenamiento y test.

3.3.2. Descripción de los datos.

Las imágenes que contiene nuestra Base de datos son de muy distintos tamaños y orientación. Casi la totalidad de las mismas son a color, pero existen excepciones a blanco y negro. A mayores no en todas aparece un rostro o uno que se sitúe mirando al frente. Para toda esta cantidad de datos obviamente hemos realizado el filtrado de las restricciones descritas con anterioridad y el posterior tratamiento para resaltar el rostro de la persona en la imagen. En las imágenes positivas de la Base de datos se puede extraer los siguientes patrones:

- Gorro mayormente blanco y rojo sin dorado (para no confundir con el rey mago Gaspar).
- Rostro humano con dos ojos y piel rosada.
- Gran barba blanca.

Estos patrones de colores son los que usaremos para que nuestro sistema analice las zonas de las imágenes y así determinar si se trata de Papá Noel o no.

4. Análisis bibliográfico.

Para poder realizar el sistema de la manera más formada posible, es necesario que busquemos trabajos relacionados con nuestra problemática para tener un ejemplo o bases de las que partir. Por ese motivo, en esta sección realizaremos un viaje por varios trabajos relacionados con aspectos de nuestro sistema.

Actualmente, el campo del reconocimiento facial está muy estudiado y cuenta con una tecnología muy avanzada. Existen numerosas técnicas para su empleo, como las geométricas (en las que distintas partes de las imágenes se transforman en primitivas geométricas), las de análisis de la textura de la piel (basadas en la apariencia y análisis del espacio), las basadas en vídeos (que permiten construir un modelo en 3D y analizar también características y cambios temporales) y las que usamos en nuestro trabajo; las holísticas (que consisten en extraer los rasgos más característicos y representarlos como un vector de pesos). Todas estas técnicas necesitan un proceso de extracción de patrones característicos de imágenes, la cual se puede conseguir mediante la ayuda de redes de neuronas artificiales como explica J.A. López Orozco en su artículo [5].

Como hemos mencionado, este tipo de problemas y estudios están a la orden del día en el mundo de la informática. Sin embargo, resulta complicado encontrar trabajos que empleen la misma técnica que nosotros, ya que al estar tan establecida, los investigadores buscan desarrollar nuevos métodos o mejorar los ya instaurados. Un trabajo muy interesante sería el de Néstor Llamas Llopis [8], en el cual se busca determinar de forma automática la identidad y la localización temporal de las personas que aparecen en un programa de televisión utilizando técnicas de video tracking y de reconocimiento facial. Otro más cercano al nuestro, sería el de Luis Antonio González Hernández y María Guadalupe López Medina [6]. Este, está centrado en el reconocimiento facial en un ámbito más general y utilizando algoritmos de extracción de características, así como redes neuronales para realizar la clasificación. Otro trabajo centrado en reconocimiento facial es el de Cépeda Neira, Lourdes del Pilar y Roncal Lázaro, Selwyn [1]; que emplean redes Hopfield y el algoritmo de Viola-Jones a través de Matlab entre otras cosas, para resolver el problema. Otro ejemplo sería el de Cadena Moreano JA, Montaluisa Pulloquinga RH, Flores Lagla GA, Chancúsig Chisag JC, Guaypatín Pico OA [2] en el cual se abordan los diferentes procesos, etapas y métodos de extracción de características que operan los sistemas de reconocimiento facial.

Por otro lado, Marcos del Pozo Baños [3], en su trabajo de fin de máster, propone un sistema biométrico de detección facial sobre vídeo basado en patrones binarios locales aplicados sobre el color. Un trabajo interesante, pero no tan complejo como el de Antonio Marín Hernández, José Alberto Medina Covarrubias y Dino Alejandro Pardo Guzmán [7]. En él, se presentan los resultados y avances de un sistema de detección de fatiga y distracción en conductores de camiones en minería de cielo abierto, con una unidad de censado de imagen en infrarrojo, capaz de capturar imágenes de ojos cubiertos con lentes de sol; un sistema de iluminación con ajuste del eje de polarización y filtro espectral, para reducción de ruido lumínico proveniente de la radiación solar. También se presentan algoritmos de detección de rasgos faciales para localización y monitores de contornos y rasgos faciales.

Otro trabajo similar al nuestro es el de Pedro Pablo García García [5], que describe

el proceso de extracción de patrones característicos de imágenes, mediante la ayuda de Redes Neuronales Artificiales, al igual que hacemos nosotros. Por último, tenemos el trabajo de Dilan Andrés Caro Barranco y Alexa Carolina López Roncancio [4], que desarrollaron un sistema de registro de asistencia basado en técnicas de reconocimiento facial y procesamiento digital de imágenes. Para llevar a cabo dicho proyecto, identificaron y caracterizaron las variables relevantes para el reconocimiento facial, para luego, diseñar e implementar un sistema inteligente para la toma de asistencia a un encuentro de personas en una interfaz web, utilizando redes neuronales, histogramas de gradientes orientados y las variables anteriormente caracterizadas.

5. Desarrollo.

5.1. Primera iteración.

5.1.1. Descripción.

La solución propuesta consiste en el análisis de las imágenes en las que aparece Papá Noel a través de un sistema inteligente. Como se explica en la sección 2.3. hemos tenido que filtrar estas imágenes para quedarnos con las que sean analizables, y a partir de esas, hemos recortado las caras para obtener una colección de datos que sirva para entrenar el sistema.

Para procesar estas imágenes, las dividiremos en una rejilla de siete filas y tres columnas, haciendo la media de los colores de los píxeles de las celdas de la imagen. Para hacerlo, partiremos de la base de que cada fila tiene 3 valores de color (rojo, verde y azul). Para esta primera iteración haremos la media y desviación típica de los 3 colores de las 3 celdas. Estas celdas coinciden con la ubicación de los rasgos más característicos de Papá Noel descritos en la sección 2.3.2. Luego, compararemos la media anteriormente obtenida con el valor de los colores que debería tener una foto de Santa Claus, y si coinciden, el sistema lo notificará.

Figura 4: División de imágenes

A la hora de recoger las imágenes de la base de datos para proceder al entrenamiento y validación, usaremos la técnica de cross validation (CV). Esta consiste en dividir el conjunto de entrada en subconjuntos de k elementos y establecer cuales servirán para entrenamiento y cuales para test. Esto se realizará para evaluar el rendimiento de los modelos de aprendizaje usados en nuestro sistema.

En la siguiente iteración, podríamos reconocer además si se trata de un rostro humano, ya que con el proceso de análisis de colores por áreas anteriormente explicado, no garantiza que realmente se trate de Papá Noel porque podría dar positivo con cualquier otro objeto que coincida con esos colores en esas zonas de la imagen.

Al procesar una imagen, obtenemos la media de los tres colores principales (rojo, verde, azul) de las tres áreas más características de las imágenes de Santa Claus como se indica en la descripción, por lo que en total tenemos 9 atributos por cada patrón. Al trabajar con las medias de los colores de ciertas áreas de las imágenes, no es necesario normalizar la base de datos ya que los colores están acotados entre 0x0 y 0xFFFFFF, de todos modos por simplicidad, pasamos a acotarlos de 0 a 1 usando números racionales. En las siguientes tablas (tabla 1, tabla 2) se puede apreciar la media y la desviación típica de los patrones positivos y negativos. Es curioso como algunos patrones por mucho que se entrene, el modelo siguen produciendo un falso positivo (figura 5b) o un falso negativo (figura 5a).

Imágenes con Santa	Rojo 1ª celda	Verde 1ª celda	Azul 1ª celda	Rojo 2ª celda	Verde 2ª celda	Azul 2ª celda	Rojo 3ª celda	Verde 3ª celda	Azul 3ª celda
Media	0.86	0.86	0.81	0.70	0.52	0.44	0.79	0.73	0.70
Desviación típica	0.09	0.10	0.12	0.12	0.13	0.13	0.11	0.13	0.15

Tabla 1: Tabla con media y std. de los 9 atributos (poca variabilidad de imágenes)

Imágenes sin Santa	Rojo 1ª celda	Verde 1ª celda	Azul 1ª celda	Rojo 2ª celda	Verde 2ª celda	Azul 2ª celda	Rojo 3ª celda	Verde 3ª celda	Azul 3ª celda
Media	0.57	0.44	0.37	0.62	0.46	0.39	0.62	0.45	0.39
Desviación típica	0.22	0.18	0.13	0.13	0.14	0.14	0.14	0.14	0.15

Tabla 2: Tabla con media y std. de los 9 atributos (mayor variabilidad de imágenes)

Con estos datos, podemos pensar que se puede obtener buenos resultados (que los veremos en la siguiente sección) sin mucho miedo a falsos positivos. El problema principal que podremos tener serán imágenes con un alto contenido de color blanco en los extremos superior e inferior y algo de rosado en el centro. Como podemos ver en las siguientes imágenes, esto pasa y podrá ser un punto de enfoque para la siguiente iteración evitando que un gato salgo como positivo.

(a) Imagen que el modelo clasifica negativa pero es positiva

(b) Imagen que el modelo clasifica positiva pero es negativa

5.1.2. Resultados.

Con la base de datos ya disponible, comenzamos entrenado un modelo de R.N.A. durante cien ciclos y un ratio de aprendizaje del 0.01 con unos errores bastante próximos entre sí representado en la gráfica (figura 6), llegamos a la conclusión experimentalmente y por la simplicidad del la clasificación como la planteamos que la mejor arquitectura eran dos capas ocultas de dieciséis y ocho neuronas respectivamente, con nueve de entrada y una de salida al solo haber dos clases, usando en todas ellas funciones de transferencia sigmoides, por esto tenemos a mayores un parámetro de umbral para separar la clase de la neurona de salida en positivo o negativo, por lo que recurrimos a la curva ROC para encontrar el mejor valor que aproximadamente es 0.5 como se puede ver en la gráfica (figura 7).

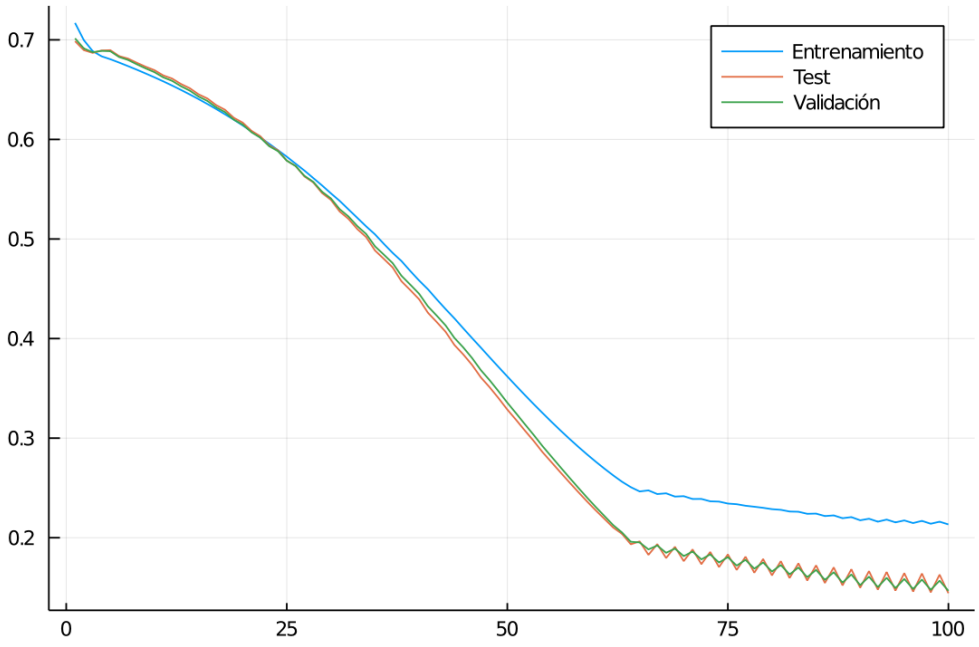


Figura 6: Errores de entrenamiento, test y validación (iteración 1)

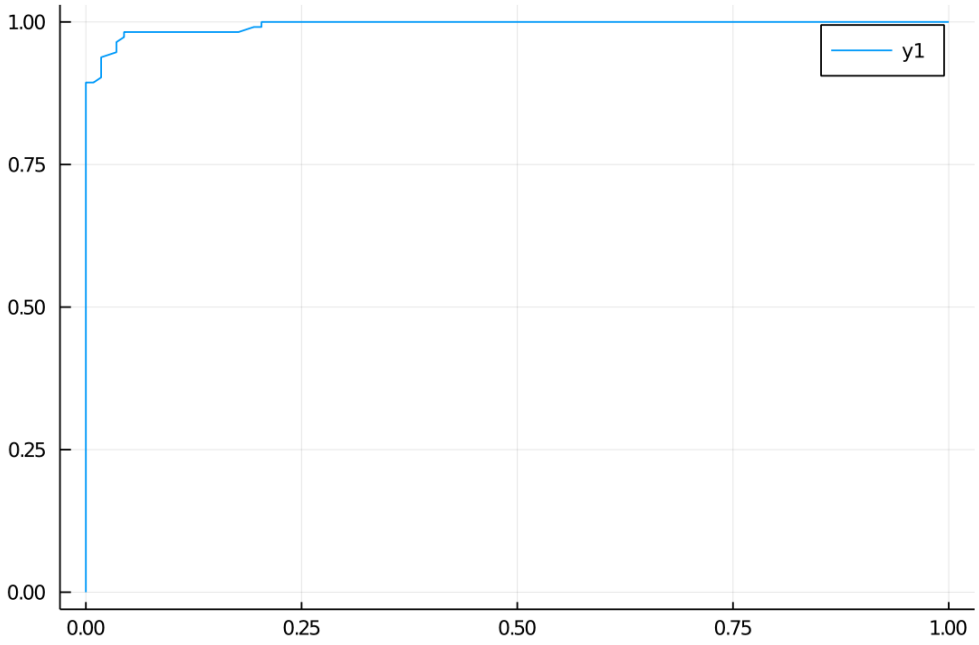


Figura 7: Curva ROC obtenida sobre la mejor R.N.A (iteración 1)

Una vez encontrado el umbral óptimo, volvimos a probar la red entrenada con este y todos los patrones de la base de datos, obteniendo los resultados de la tabla 3 y la matriz de confusión se puede observar en la tabla 4.

Umbral	0.5
Precisión	92.5 %
Tasa de error	7.4 %
Sensibilidad	96.5 %
Especificidad	88.5 %
Valor predicativo positivo	89.4 %
Valor predicativo negativo	96.2 %
F1-score	92.8 %

Tabla 3: Tabla con las estadísticas resultantes de la R.N.A. con todos los patrones

Como hemos mencionado, estos resultados son con el umbral óptimo (que en nuestro caso es 0.5). En ellos podemos ver como tenemos un buen número de precisión (recordemos que es aún la primera iteración) y sensibilidad pese a algún que otro error. Si nos fijamos en la tabla de confusión vemos como estos errores tienden a ser sobre los falsos positivos, lo que indica que el sistema es más favorable a producir un positivo en una imagen sin Papá Noel, que un negativo en una imagen con el personaje. Esto se ajusta a lo que queremos, ya que preferimos que ante una imagen de Santa Claus intente siempre acertar pese a fallar en

imágenes donde no está. Esto nos produce un buen Recall, que sumado a la ya mencionada precisión, nos proporciona un buen F1-score (que no deja de ser una media harmónica entre estos valores).

		Predicción	
		Negativo	Positivo
Real	Negativo	155	20
	Positivo	6	169

Tabla 4: Tabla de confusión resultante de probar la R.N.A. con todos los patrones

Después de conseguir experimentalmente una arquitectura adecuada de R.N.A., nos dispusimos a probarla con validación cruzada con 'k' igual a diez junto con otras buenas arquitecturas y tres algoritmos (SVM, kNN y árboles de decisión) de la librería de Python 'scikit-learn' con bindings a Julia. En las tablas 5, 9, 8, y 7 se muestran la media y desviación típica de la precisión y F1 de cada modelo de cada algoritmo.

RNA	**1º modelo**	**2º modelo**	**3º modelo**	**4º modelo**
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32,16]	[16,8]	[8,4]	[4,2]
Ciclos máximos	200	100	100	100
Ratio de aprendizaje	0.01	0.01	0.02	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	90.35	95.17	94.99	78.99
Desviación típica	7.40	5.05	5.39	10.49
F1 media	89.09	95.70	95.68	71.81
Desviación F1	8.45	4.25	4.38	14.85

Tabla 5: Tabla con resultados de la RNA para distintos modelos

RNA	5º modelo	6º modelo	7º modelo	8º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32]	[16]	[8]	[4]
Ciclos máximos	100	100	100	100
Ratio de aprendizaje	0.01	0.01	0.01	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	94.93	95.45	94.21	89.06
Desviación típica	5.85	4.92	5.67	6.09
F1 media	95.46	95.82	94.93	87.62
Desviación F1	5.01	4.11	4.84	7.35

Tabla 6: Tabla con resultados de la RNA para distintos modelos

Árboles de decisión	Profundidad máxima del árbol	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	92.90	6.88	92.47	7.28
2º modelo	4	92.45	7.17	92.31	7.33
3º modelo	8	91.12	9.41	91.35	8.97
4º modelo	16	91.1	9.41	91.35	8.97
5º modelo	32	91.12	9.41	91.35	8.97
6º modelo	64	91.12	9.41	91.35	8.97

Tabla 7: Tabla con resultados de los árboles de decisión para distintos modelos

Knn	Nº de vecinos	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	95.55	6.33	95.71	5.98
2º modelo	4	95.11	7.09	95.20	6.95
3º modelo	8	95.13	6.65	95.15	6.57
4º modelo	16	94.70	6.43	94.55	6.36
5º modelo	32	95.13	5.23	95.32	4.94
6º modelo	64	91.60	7.51	91.54	7.42

Tabla 8: Tabla con resultados de kNN para distintos modelos

SVM	1º modelo	2º modelo	3º modelo	4º modelo	5º modelo	6º modelo	7º modelo	8º modelo
Kernel	rbf"	rbf"	rbf"	"linear"	"poly"	"poly"	"poly"	"sigmoid"
Grado del Kernel	-	-	-	-	3	3	3	-
Gamma del Kernel	2	"scale"	.ªuto"	-	2	"scale"	.ªuto"	2
'C'	1	1	1	1	1	1	1	1
Precisión media	95.59	95.59	96.02	95.57	95.55	95.55	91.58	38.87
Desviación típica	6.82	6.82	5.63	6.24	5.45	5.45	7.06	5.26
F1 media	95.74	95.74	96.25	95.91	95.47	95.47	91.11	23.48
Desviación F1	6.73	6.73	5.31	5.69	5.58	5.58	7.11	30.46

Tabla 9: Tabla con resultados de las SVM para distintos modelos

5.1.3. Discusión.

Como se puede comprobar, este problema es perfectamente abordable con cualquiera de los cuatro algoritmos, obteniendo unos resultados más que aceptables en los ejemplos de nuestra base de datos, aunque bien es cierto que esta ha tenido que ser filtrada manualmente varias veces para favorecer la distinción por colores en las áreas indicadas, siendo esto sin duda la parte más tediosa de la iteración, aunque también la función de validación cruzada de los cuatro modelos se complicó bastante. Sobre el mejor algoritmo, si nos fijamos en las tablas 5, 9, 8, y 7 veremos que el modelo de SVM es ligeramente mejor que la R.N.A., pero esta última ofrece resultados más estables respecto a su desviación típica, aunque kNN también ofrece resultados bastante buenos siendo los árboles de decisión peor algoritmo para este problema. Por otro lado, el algoritmo R.N.A. es bastante más costoso de entrenar que los otros algoritmos con diferencia, pero a pesar de ello por sus buenos resultados seguiremos usando preferentemente este algoritmo.

5.2. Segunda iteración.

5.2.1. Descripción

Siguiendo la primera iteración, lo que buscaremos con la segunda es reducir el número de atributos necesarios para que el sistema tenga un mejor rendimiento y pueda darnos mejores resultados al replantearnos los datos que le pasamos. Para eso lo que haremos será pasar de 9 atributos (3 por cada celda) que teníamos antes a 7 (2 en la primera y la tercera y 3 en la segunda). Esto lo hacemos así ya que Papá Noel tiene un gorro y barba blanca, por lo que los colores de la primera y tercera fila se pueden ignorar, fijándonos solo en la desviación típica y media de ellos para ver si se trata de blanco o no.

Imágenes con Santa	1º Celda	2º Celda Rojo	2º Celda Verde	2º Celda Azul	3º Celda
Media	0.87	0.72	0.54	0.46	0.77
Desviación Típica	0.04	0.11	0.12	0.12	0.04

Tabla 10: Datos de entrada para imágenes positivas

Imágenes sin Santa	1º Celda	2º Celda Rojo	2º Celda Verde	2º Celda Azul	3º Celda
Media	0.44	0.61	0.45	0.38	0.48
Desviación Típica	0.11	0.14	0.13	0.14	0.11

Tabla 11: Datos de entrada para imágenes negativas

5.2.2. Resultados.

Para esta iteración se continúa con la misma arquitectura de 100 ciclos, ratio de aprendizaje de 0.01 y dos capas ocultas de 16 y 8 neuronas y manteniendo el umbral en aproximadamente 0.5, para esta iteración obtenemos una precisión del 94 una tasa de fallos del 6 una sensibilidad del 93.7 una especificidad del 94.2 un valor predictivo positivo de 94.2 un valor predictivo negativo del 93.7 y un F1 de 93.9, siendo esta iteración mas precisa, mas especifica y con un mayor valor predictivo positivo y F1 que la primera, pero teniendo menos sensibilidad y valor predictivo negativo. Esto se puede observar mejor en la tabla que contiene la matriz de confusión 17

RNA	**1º modelo**	**2º modelo**	**3º modelo**	**4º modelo**
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32,16]	[16,8]	[8,4]	[4,2]
Ciclos máximos	200	100	100	100
Ratio de aprendizaje	0.01	0.01	0.02	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	69.65	90.99	88.41	73.66
Desviación típica	9.83	7.06	6.93	4.24
F1 media	61.04	89.37	85.61	67.82
Desviación F1	7.82	9.59	9.57	7.54

Tabla 12: Tabla con resultados de la R.N.A. para distintos modelos

RNA	5º modelo	6º modelo	7º modelo	8º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32]	[16]	[8]	[4]
Ciclos máximos	100	100	100	100
Ratio de aprendizaje	0.01	0.01	0.01	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	86.87	93.60	90.59	84.64
Desviación típica	3.96	3.45	5.70	3.86
F1 media	84.86	92.75	89.69	83.98
Desviación F1	5.49	4.49	8.41	5.15

Tabla 13: Tabla con resultados de la R.N.A. para distintos modelos

Árboles de decisión	Profundidad máxima del árbol	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	94.68	4.64	94.71	4.85
2º modelo	4	93.77	4.32	93.38	4.15
3º modelo	8	93.77	3.80	93.34	3.90
4º modelo	16	93.77	3.80	93.34	3.90
5º modelo	32	93.77	3.80	93.34	3.90
6º modelo	64	93.34	3.80	93.34	3.90

Tabla 14: Tabla con resultados de los árboles de decisión para distintos modelos

kNN	Nº de vecinos	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	93.77	4.86	93.25	4.89
2º modelo	4	96.00	3.28	95.57	3.53
3º modelo	8	96.00	2.56	95.56	3.24
4º modelo	16	95.13	3.25	94.72	3.64
5º modelo	32	94.70	2.79	94.32	3.19
6º modelo	64	91.66	4.75	91.76	4.14

Tabla 15: Tabla con resultados de kNN para distintos modelos

SVM	1º modelo	2º modelo	3º modelo	4º modelo	5º modelo	6º modelo	7º modelo	8º modelo
Kernel	rbf"	rbf"	rbf"	"linear"	"poly"	"poly"	"poly"	"sigmoid"
Grado del Kernel	-	-	-	-	3	3	3	-
Gamma del Kernel	2	"scale"	.ªuto"	-	2	"scale"	.ªuto"	2
'C'	1	1	1	1	1	1	1	1
Precisión media	96.89	96.89	94.70	95.57	96.46	96.46	86.81	18.51
Desviación típica	2.14	2.14	2.79	2.14	4.05	3.44	5.70	8.92
F1 media	96.40	96.40	94.45	95.19	96.10	96.13	85.86	11.13
Desviación F1	2.89	2.89	3.10	2.87	4.27	3.70	5.58	7.76

Tabla 16: Tabla con resultados de las SVM para distintos modelos

		predicción	
		negativo	positivo
real	negativo	164	10
	positivo	11	165

Tabla 17: Matriz de confusión para la segunda iteración

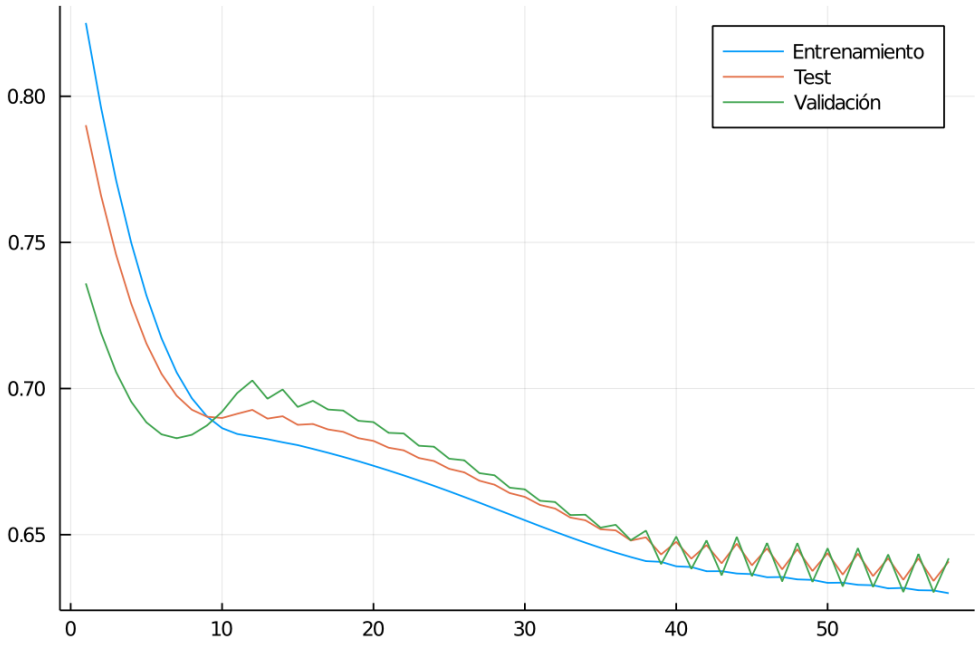


Figura 8: Errores de entrenamiento, test y validación (iteración 2)

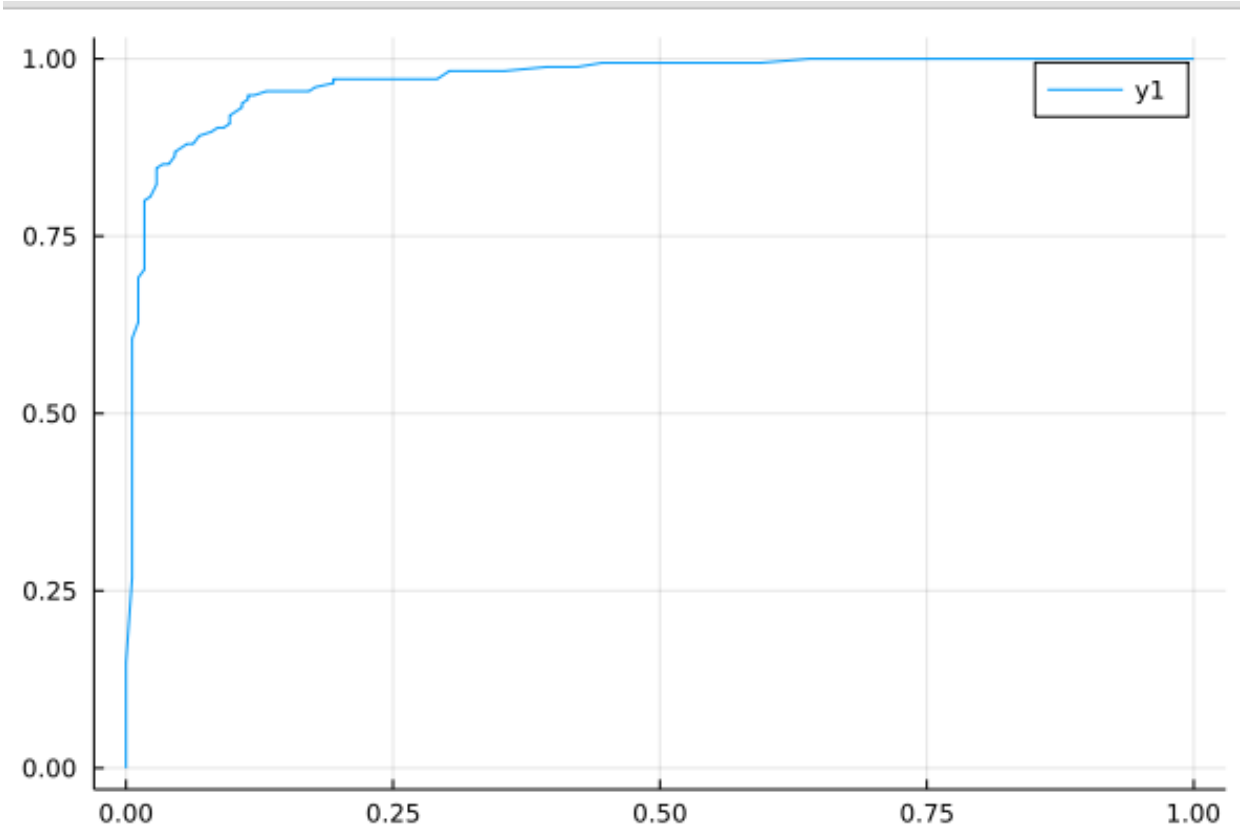


Figura 9: Curva ROC obtenida sobre la mejor R.N.A. (iteración 2)

5.2.3. Discusión.

Una de las cosas que se buscaba con esta iteración era clasificar correctamente las dos imágenes de ejemplo en la anterior, sin embargo estas resultan muy difíciles de clasificar para nuestro sistema, ya que en el caso del falso negativo, la imagen es muy oscura y de un tono azul, por lo que no detecta el blanco. En el caso de la falsa positiva, aunque no se trate de un rostro humano, tiene las zonas donde irían el gorro y la barba completamente blancas, y la zona de la cara coincide con la nariz del gasto, que es de un color parecido. Este segundo caso se resolverá mas tarde con la tercera iteración, en la que se añadirá que la imagen contenga una cara humana.

5.3. Tercera iteración.

5.3.1. Descripción.

En esta nueva iteración lo que haremos será buscar una nueva característica en las imágenes. Como Papá Noel no deja de ser una persona, intentaremos detectar ojos de un rostro humano para así mejorar la precisión del sistema buscando que no nos proporcione falsos positivos en objetos que son de los colores del personaje navideño.

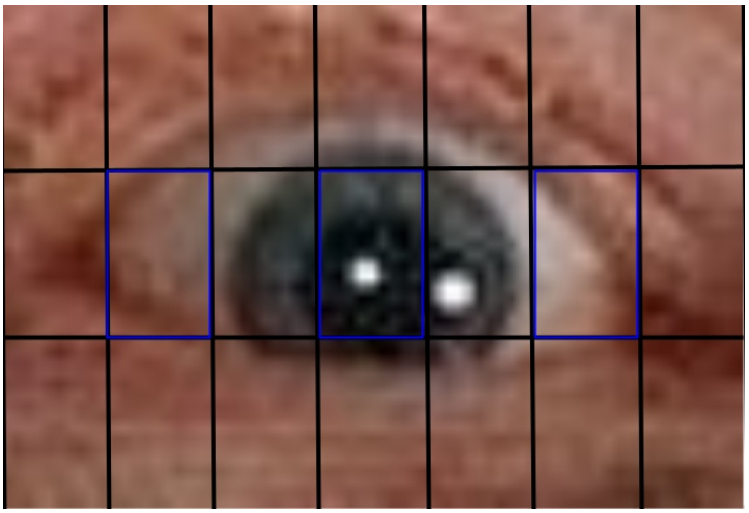


Figura 10: Imagen de entrenamiento de ojo (con las divisiones en celdas)

Figura 11: Imagen de entrenamiento de no-ojo

Para poder hacerlo, primero necesitamos extraer un conjunto de entrenamiento de nuestra base de datos original. Para eso recortamos a mano imágenes de rostros humanos (algunas mostrando los ojos y otras mostrando partes de la cara sin ellos). La red se entrena primero con las que contienen ojos dividiéndola en 3 filas y 7 columnas. De esas recogemos la media de las celdas (2,2), (2,4) y (2, 6). Cabe destacar que para hacerlo se obvian los colores de la imagen, ya que no son representativos, por lo que se pasa todo a blanco y negro.

Una vez tenemos esta nueva red, las imágenes que le pasamos de la base de datos original para comprobar si tienen o no ojos, son divididas en una cuadrícula de 60 filas y 64 columnas como aparece en la figura 12 y los ojos se extraen en rectángulos de 3 filas y 4 columnas. Es importante resaltar que no se procesa toda la imagen **para que el sistema sea más eficiente**, ya que solo necesitamos lo siguiente: para el ojo izquierdo, necesitamos el área situada entre las filas de la 20 a la 39 y de las columnas de la 16 a la 31; para el derecho, usaremos las mismas filas pero cogiendo está vez las columnas de la 32 a la 47. En cada rectángulo de esa área, será donde la nueva red buscará si existen ojos o no. Para que

esto no nos suponga ningún problema el recorte de las imágenes tiene que hacer que los ojos (si tiene) de lo que aparezca en la imagen, se establezcan en esta cuadrícula específica. Una vez realizado esto, esa iteración recuperará de la iteración 2 el procesado de los colores.

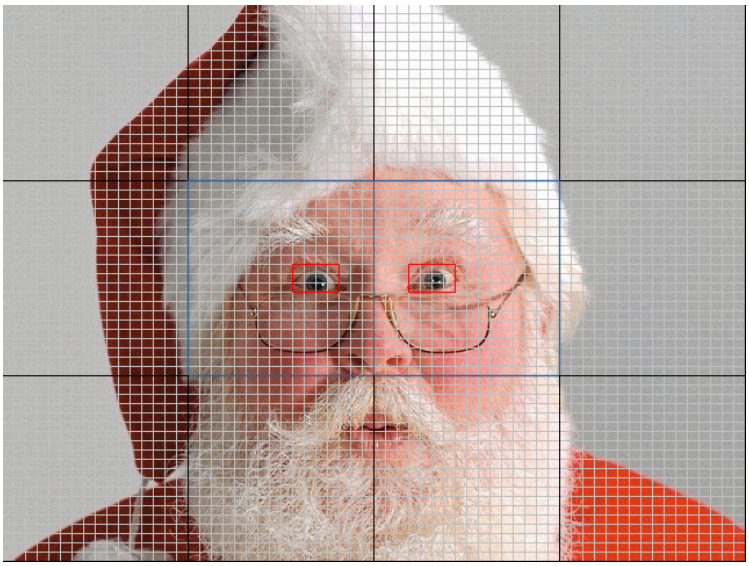


Figura 12: Imagen con la cuadricula superpuesta.

En las tablas 18 y 19 se puede apreciar las medias y desviaciones típicas de la base de datos creada con ojos y no ojos recortados de las imágenes de la de Papá Noel para entrenar la red.

Imágenes con un ojo	Gris en la 1ª celda	Gris en la 2ª celda	Gris en la 3ª celda
Media	0.49	0.27	0.42
Std.	0.19	0.15	0.18

Tabla 18: Tabla con media y std. de las celdas en grises para detectar ojos (positivo)

Imágenes sin un ojo	Gris en la 1ª celda	Gris en la 2ª celda	Gris en la 3ª celda
Media	0.63	0.62	0.62
Std.	0.15	0.14	0.15

Tabla 19: Tabla con media y std. de las celdas en grises para detectar ojos (negativo)

5.3.2. Resultados.

Como en anteriores iteraciones, para discernir entre ojo y no ojo comenzamos entrenando la arquitectura R.N.A. que usamos ya anteriormente al tratarse de un problema muy similar al de detectar los colores de Santa en una imagen, obteniendo errores bastante bajos como se puede ver en la figura 13. Es curioso como los errores del conjunto de entrenamiento y validación son notablemente más bajos que los de test en este problema.

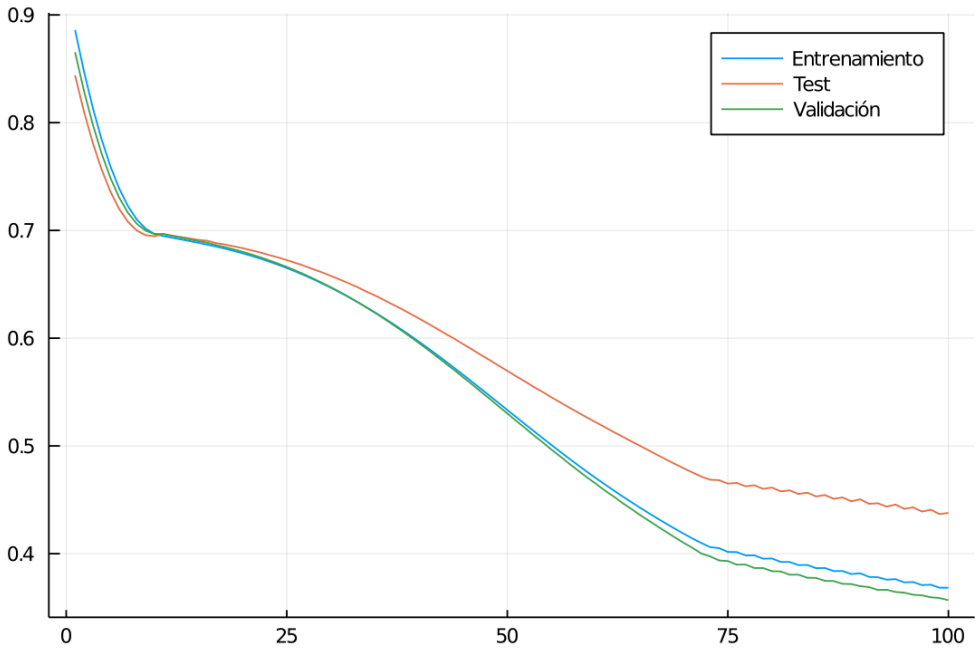


Figura 13: Errores de entrenamiento, test y validación (iteración 3)

Después de validar la arquitectura propuesta, buscamos el umbral de clasificación para separar la salida de la R.N.A. en positivo o negativo con la curva ROC, que como se puede ver en la figura 14, un buen umbral sería 0.5 aproximadamente.

RNA	1º modelo	2º modelo	3º modelo	4º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32,16]	[16,8]	[8,4]	[4,2]
Ciclos máximos	200	100	100	100
Ratio de aprendizaje	0.01	0.01	0.02	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	58.56	73.05	75.21	64.99
Desviación típica	6.24	8.62	8.93	4.34
F1 media	47.39	68.71	70.55	59.89
Desviación F1	14.6	11.67	10.41	7.37

Tabla 20: Resultados de Cross Validation para diferentes modelos de RNA

RNA	5º modelo	6º modelo	7º modelo	8º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32]	[16]	[8]	[4]
Ciclos máximos	100	100	100	100
Ratio de aprendizaje	0.01	0.01	0.01	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	70.79	78.33	77.73	70.53
Desviación típica	6.01	8.10	6.44	4.81
F1 media	66.51	77.35	76.39	65.92
Desviación F1	8.22	7.54	5.37	8.31

Tabla 21: Resultados de Cross Validation para diferentes modelos de RNA

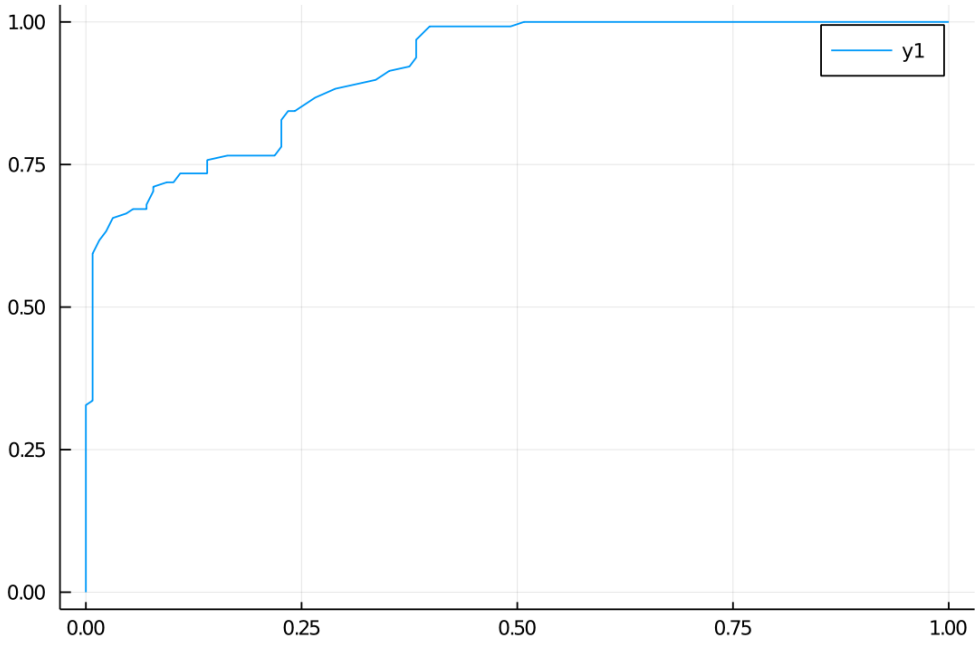


Figura 14: Curva ROC obtenida sobre la mejor R.N.A. (iteración 3)

Luego probamos la arquitectura y el umbral con todos los patrones para hacer la matriz de confusión (tabla 22 y extraer de esta todas las estadísticas (tabla 23.

		Predicción	
		Negativo	Positivo
Real	Negativo	110	18
	Positivo	30	98

Tabla 22: Tabla de confusión resultante de probar la R.N.A. con todos los patrones

Umbral	0.5
Precisión	81.5 %
Tasa de error	1.8 %
Sensibilidad	76.5 %
Especificidad	85.9 %
Valor predicativo positivo	84.4 %
Valor predicativo negativo	78.5 %
F1-score	80.3 %

Tabla 23: Tabla con las estadísticas resultantes de la R.N.A. con todos los patrones

Con esta arquitectura de R.N.A y otras dos, las comparamos con los tres algoritmos (arboles de decisión, kNN, SVM) con Cross Validation k=10 para decidir cual usar finalmente. En las tablas 20, 24, 25 y 26 se puede ver los parámetros y la media y desviación típica de la precisión y F1 de cada modelo para cada algoritmo.

Árboles de decisión	**Profundidad máxima del árbol**	**Precisión media**	**Desviación típica**	**F1 media**	**Desviación F1**
1º modelo	2	85.49	7.02	84.74	7.33
2º modelo	4	91.41	6.25	91.86	4.58
3º modelo	8	92.60	4.21	92.84	3.61
4º modelo	16	92.60	4.21	92.84	3.61
5º modelo	32	92.60	4.21	92.84	3.61
6º modelo	64	92.60	4.21	92.84	3.61

Tabla 24: Tabla con resultados de árboles de decisión para distintos modelos

kNN	**Nº de vecinos**	**Precisión media**	**Desviación típica**	**F1 media**	**Desviación F1**
1º modelo	2	90.18	6.06	89.29	7.92
2º modelo	4	90.99	4.96	90.60	4.69
3º modelo	8	90.21	6.71	90.03	6.86
4º modelo	16	91.01	5.20	90.77	4.77
5º modelo	32	90.58	5.94	90.34	5.97
6º modelo	64	84.36	7.57	82.37	8.73

Tabla 25: Tabla con resultados del kNN para distintos modelos

SVM	**1º modelo**	**2º modelo**	**3º modelo**	**4º modelo**	**5º modelo**	**6º modelo**	**7º modelo**	**8º modelo**
Kernel	rbf"	rbf"	rbf"	"linear"	"poly"	"poly"	"poly"	"sigmoid"
Grado del Kernel	-	-	-	-	3	3	3	-
Gamma del Kernel	2	"scale"	.ªuto"	-	2	"scale"	.ªuto"	2
'C'	1	1	1	1	1	1	1	1
Precisión media	89.04	91.36	86.69	87.09	88.18	92.52	77.69	29.32
Desviación típica	5.48	4.12	4.27	5.25	7.36	4.78	5.78	6.28
F1 media	88.73	91.22	85.40	85.94	88.71	92.33	80.78	27.33
Desviación F1	5.18	3.78	6.75	6.81	6.32	4.90	3.54	9.39

Tabla 26: Tabla con resultados de la SVM para distintos modelos

Como podemos ver en las tablas 20, 24, 25 y 26, la R.N.A. en general muestra peores resultados con diferencia que los demás algoritmos, siendo los árboles de decisión y kNN los mejores en este caso y SVM ligeramente inferior. Pero a pesar de esto, escogeremos la R.N.A. ya que como se demostrará experimentalmente más adelante, tiene una mayor generalización cuando se escanea una imagen de una cara en búsqueda de ojos. Una vez que tenemos un algoritmo para clasificar si hay ojo o no en una imagen, solo resta diseñar el procedimiento para escanear una imagen de una cara entera y como dijimos en la descripción de esta iteración, se agruparán los píxeles de esta en una cuadricula para extraer rectángulos primero por el lado izquierdo y luego el derecho de tres filas y cuatro columnas con los que, tras codificarlos, se evalúan dos veces con la red, una con la imagen extraída normal y otra

con efecto espejo vertical ya que los ojos son simétricos, y si se encontrara uno, ambos resultados deberían ser positivos. En este proceso, se lleva la cuenta de las coordenadas donde se encuentran ojos, y cada vez que se encuentra uno nuevo, se comprueba la distancia entre este y la media de los ya encontrados, considerándose un fallo si supera un umbral que se le pasa por parámetro, porque un rostro frontal de un humano debería concentrar los casos positivos en ojo izquierdo y luego en el derecho, y si supera un numeró de fallos que se le pasa también por parámetro, el procedimiento devuelve negativo, si no y si al menos se ha encontrado un ojo por cada lado devuelve positivo. Para encontrar el valor óptimo de estos parámetros, se a vuelto a usar la curva ROC pero varias veces para cada umbral de clasificación y para un mismo número de fallos máximo. Como se puede ver el la figura 15, aproximadamente el valor óptimo para el umbral de clasificación es del 0.8, el de la distancia máxima es 0.3 y el de número de fallos máximo es 5.

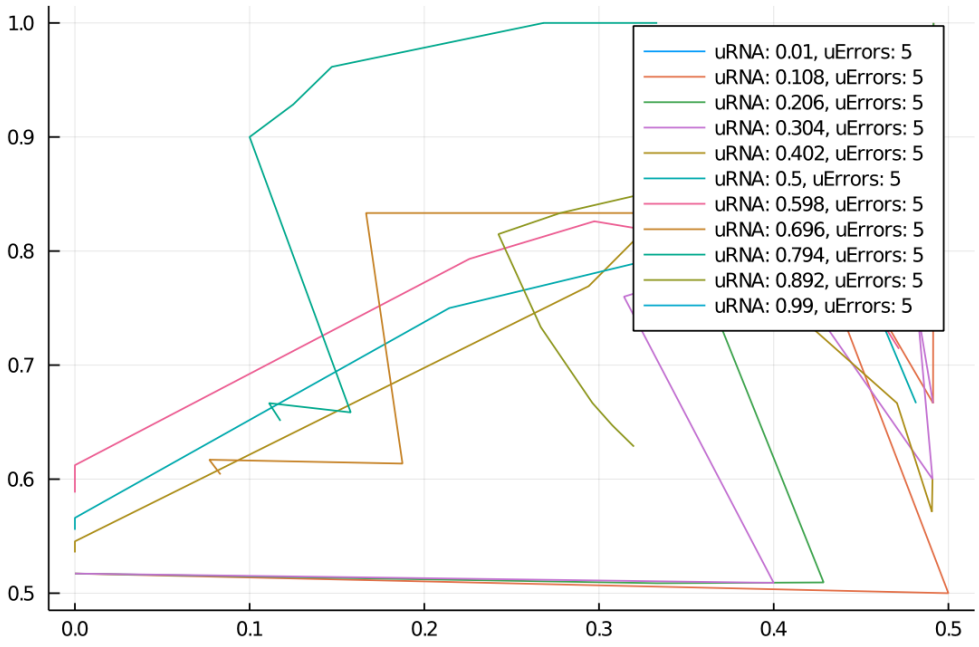
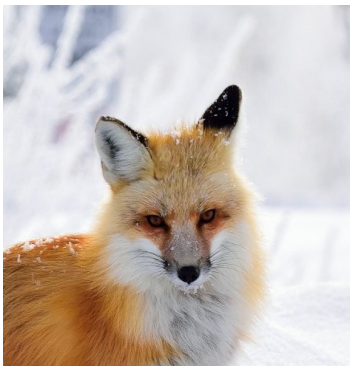


Figura 15: Curva ROC combinando umbral de clasificación y distancia máxima

Tras conseguir los parámetros óptimos para el procedimiento detector de rostros, lo probamos con la base de datos de casos positivos de santa y una nueva con imágenes que no son rostros para los negativos, esta segunda ha sido creada de forma que al ser evaluada con la red de la anterior iteración, ofrece una gran cantidad de falsos positivos como se puede ver en las figuras de ejemplo 16a y 16b, y en la tabla de resultados 27 y la de confusión 28.

(a) Imagen que el modelo clasifica positiva correctamente



(b) Imagen que el modelo clasifica positiva pero es negativa

Umbral	0.5
Precisión	50 %
Tasa de error	50 %
Sensibilidad	50 %
Especificidad	50 %
Valor predicativo positivo	93.3 %
Valor predicativo negativo	6.6 %
F1-score	6.5 %

Tabla 27: Tabla con las estadísticas resultantes de la R.N.A. con todos los patrones

		Predicción	
		Negativo	Positivo
Real	Negativo	2	2
	Positivo	18	18

Tabla 28: Tabla de confusión resultante de probar la R.N.A. con todos los patrones

En cambio, al probar las bases de datos anteriormente mencionada combinando la salida de la red de la anterior iteración con el procedimiento detector de rostros de esta con una conjunción lógica, muestra unos resultados bastante buenos como se puede ver en la tabla de resultados 29 y la de confusión 30 al no solo basarse en los colores del gorro, cara y barba para clasificar entre Santa y no Santa.

Umbral de clasificación	0.8
Distancia máxima	0.3
Número máximo de fallos	5
Precisión	86.6 %
Tasa de error	13.3 %
Sensibilidad	92.3 %
Especificidad	82.3 %
Valor predicativo positivo	80.0 %
Valor predicativo negativo	9.3 %
F1-score	8.5 %

Tabla 29: Tabla con las estadísticas resultantes de la R.N.A. con todos los patrones

		Predicción	
		Negativo	Positivo
Real	Negativo	28	6
	Positivo	2	24

Tabla 30: Tabla de confusión resultante de probar la R.N.A. con todos los patrones

5.3.3. Discusión.

Concluimos exitosamente la combinación de los dos sistemas de clasificación para detectar a Papá Noel en imágenes, aunque no se ha podido probar de forma más extendida con un mayor número de patrones debido al alto coste temporal de buscar y recortar más imágenes que confundan al sistema de la anterior iteración. Por lo que en las siguiente iteraciones seguiremos usando este sistema junto con las optimizaciones que haremos en la codificación de las imágenes.

5.4. Cuarta iteración.

5.4.1. Descripción.

Como penúltima iteración, lo que haremos será reducir aun más el número de atributos necesarios para que el sistema tenga un mejor rendimiento y pueda darnos mejores resultados al replantearnos los datos que le pasamos. Para eso lo que haremos será pasar de 7 atributos que teníamos antes (recordemos que de la primera iteración a la segunda pasamos de 9 a estos 7) a solo 5. Esto lo conseguiremos sacando la media y la desviación típica de la barba.

En resumen: tendremos 3 atributos en la fila del medio de la imagen (1 por cada color) y dos en las filas adyacentes (1 para la desviación típica y otro para la media de los colores). Esto lo podemos hacer ya que el gorro y la barba realmente son muy parecidos,

por lo que haciendo lo mismo para estas últimas características podremos en un principio conseguir resultados muy parecidos pero de forma más eficiente y rápida.

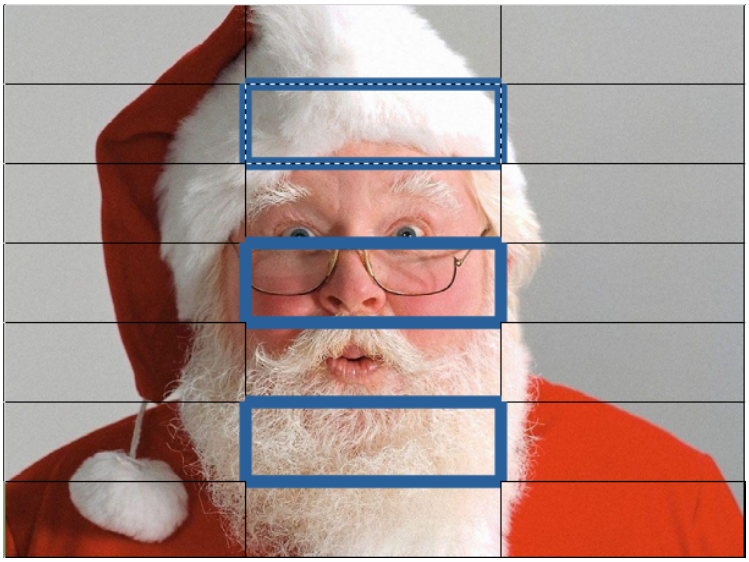


Figura 17: División de imágenes

Si volvemos a ver la imagen que nos muestra como dividimos las fotos de Papá Noel, podemos observar como la zona del gorro y de la barba son muy parecidas. Esto es lo que usamos como premisa a la hora de llevar a cabo esta nueva iteración. La zona crítica sigue siendo la del medio, por lo que ahí seguiremos analizando los 3 colores primarios y usando también la detección del rostro mediante los ojos realizada en la tercera iteración.

Imágenes con Santa	1º y 3º Celda	2º Celda Rojo	2º Celda Verde	2º Celda Azul
Media	0.82	0.72	0.54	0.46
Desviación Típica	0.07	0.11	0.12	0.12

Tabla 31: Valores de entrada para imágenes positivas

Imágenes sin Santa	1º y 3º Celda	2º Celda Rojo	2º Celda Verde	2º Celda Azul
Media	0.46	0.61	0.45	0.38
Desviación Típica	0.14	0.14	0.13	0.14

Tabla 32: Valores de entrada para imágenes negativas

5.4.2. Resultados.

Al ejecutar esta iteración usamos el mismo modelo de R.N.A que en el resto y se obtienen las siguientes gráficas, de errores en las etapas de entrenamiento test y validación 18 y la curva ROC 19

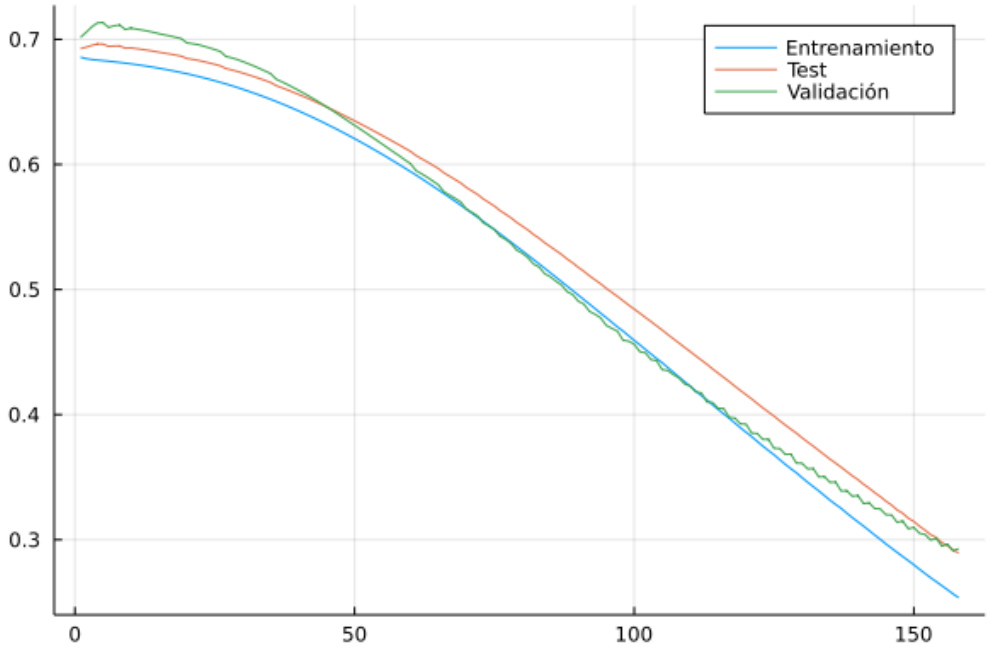


Figura 18: Errores de entrenamiento, test y validación

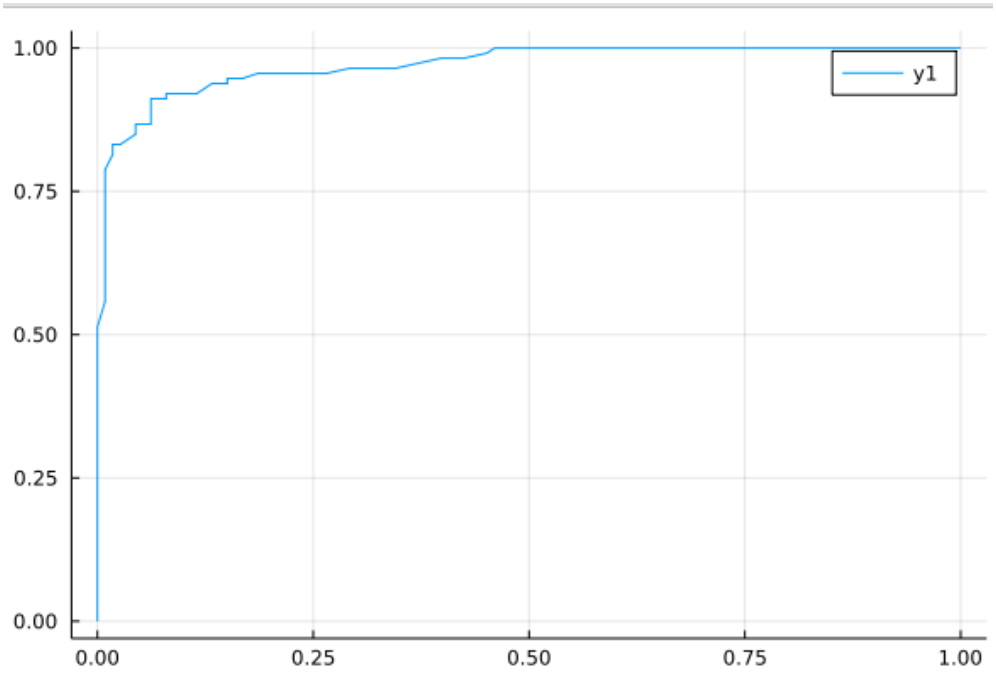


Figura 19: Curva ROC obtenida con la red

También obtenemos la matriz de confusión 33 así como la tabla donde se reflejan todas las métricas 34

		Predicción	
		Negativo	Positivo
Real	Negativo	97	16
	Positivo	17	96

Tabla 33: Matriz de confusión para la iteración 4

Umbral	0.5
Precisión	0.854
Tasa de error	0.146
Sensibilidad	0.849
Especificidad	0.858
Valor Predictivo positivo	0.857
Valor Predictivo negativo	0.850
F1-score	0.853

Tabla 34: Tabla de resultados de la iteración 4

Una vez tenemos estos resultados obtenemos la media y desviación típica de la precisión y la F1-score de diferentes algoritmos (R.N.A, SVM, kNN, Árboles de decisión) con diferentes modelos mediante validación cruzada, lo que resulta en las siguientes tablas.

RNA	1º modelo	2º modelo	3º modelo	4º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32,16]	[16,8]	[8,4]	[4,2]
Ciclos máximos	200	100	100	100
Ratio de aprendizaje	0.01	0.01	0.02	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	52.08	77.71	81.71	62.08
Desviación típica	10.81	10.85	7.89	8.30
F1 media	39.71	76.50	78.48	55.79
Desviación F1	14.70	11.58	11.21	8.86

Tabla 35: Resultados de Cross Validation para diferentes modelos de RNA

RNA	5º modelo	6º modelo	7º modelo	8º modelo
Nº de ejecuciones por k	10	10	10	10
Ratio del conjunto de validación	0.25	0.25	0.25	0.25
Topología de la red	[32]	[16]	[8]	[4]
Ciclos máximos	100	100	100	100
Ratio de aprendizaje	0.01	0.01	0.01	0.01
Error mínimo	0	0	0	0
Nº de ciclos sin mejorar máximos	10	10	10	10
Normalización de base de datos	true	true	true	true
Umbral de clasificación de la neurona de salida	0.5	0.5	0.5	0.5
Precisión media	70.88	83.59	82.78	76.08
Desviación típica	5.82	7.50	5.78	8.22
F1 media	67.58	80.56	79.82	75.20
Desviación F1	6.13	9.34	11.57	10.12

Tabla 36: Resultados de Cross Validation para diferentes modelos de RNA

Árboles de decisión	Profundidad máxima del árbol	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	92.43	4.26	91.55	5.82
2º modelo	4	92.84	5.74	91.62	7.54
3º modelo	8	93.28	5.77	92.18	7.53
4º modelo	16	93.28	5.77	92.18	7.53
5º modelo	32	93.28	5.77	92.18	7.53
6º modelo	64	93.28	5.77	92.18	7.53

Tabla 37: Resultados de Cross Validation para diferentes modelos de Arboles de decisión

kNN	Nº de vecinos	Precisión media	Desviación típica	F1 media	Desviación F1
1º modelo	2	92.88	3.80	91.99	5.52
2º modelo	4	93.73	3.88	93.19	5.57
3º modelo	8	95.05	3.98	94.59	4.94
4º modelo	16	93.32	3.85	92.64	5.41
5º modelo	32	91.97	3.65	91.32	5.32
6º modelo	64	91.12	5.55	89.76	8.20

Tabla 38: Resultados de Cross Validation para diferentes modelos de kNN

SVM	1º modelo	2º modelo	3º modelo	4º modelo	5º modelo	6º modelo	7º modelo	8º modelo
Kernel	rbf"	rbf"	rbf"	"linear"	"poly"	"poly"	"poly"	"sigmoid"
Grado del Kernel	-	-	-	-	3	3	3	-
Gamma del Kernel	2	"scale"	"auto"	-	2	"scale"	.ªuto"	2
'C'	1	1	1	1	1	1	1	1
Precisión media	94.64	94.64	92.45	93.73	95.53	95.53	83.24	23.35
Desviación típica	3.56	3.56	4.23	4.39	3.03	3.65	4.79	6.87
F1 media	94.06	94.06	91.80	93.14	94.96	94.71	80.58	19.02
Desviación F1	4.73	4.73	5.60	5.94	4.33	4.86	6.55	9.25

Tabla 39: Resultados de Cross Validation para diferentes modelos de SVM

5.4.3. Discusión.

A partir de los resultados obtenidos se puede apreciar que el mejor algoritmo para esta iteración sería utilizar una máquina de soporte vectorial con un kernel tipo 'poly', otros buenos resultados serían de SVM con un kernel 'rbf' o un kNN con 5 vecinos.

Comparando esta iteración con la iteración 2, ya que la meta de esta iteración 4 era obtener una mejora con respecto a la 2, se observa que los resultados de esta iteración son generalmente peores que los de la segunda bajando un poco en todas las métricas, mostrando que, mientras generalizar la primera y tercera celda haciendo la media y desviación típica de los tres valores r g y b de estas reportó una mejora, generalizar esos valores aun más y mezclarlos es contraproducente, esto puede deberse a varios factores como la iluminación de las imágenes o cierto ruido que puede ser causado por pequeñas partes de otros colores, sobre todo en la celda correspondiente al gorro, ya que su parte blanca es solo una pequeña franja y , por la posición de la celda 1, es posible que se cuele un poco del rojo del resto del gorro, lo que empeora los resultados.

5.5. Quinta iteración.

5.5.1. Descripción.

En esta última iteración aplicaremos las técnicas de Deep Learning con nuestro problema de detectar si en una imagen de una cara es la de Papá Noel o no, aprovechando su potencial de reconocimiento automático de las característica en imágenes con sus redes neuronales convolucionales, las cuales nos fueron proporcionadas directamente en el lenguaje Julia, con una arquitectura de nueve capas como se puede ver en la figura 20, una primera de convolución, la segunda es una función de compresión, y así alternando entre estas dos hasta una antepenúltima capa de redimensionación, luego una capa de neuronas totalmente conectadas y finalmente se aplica la función Softmax para proporcionar una salida acotada entre cero y uno para cada clase, que en nuestro caso solo serán dos, por lo que podemos omitir esta última capa y obtener la salida directamente de la neurona de salida de la penúltima capa, esta salida se separará en dos clases con un umbral que se obtendrá experimentalmente como se verá en los resultados.

Uno de los procesos más distintivos de estas redes son las convoluciones. El cual consiste en tomar un grupo de píxeles de la imagen de entrada e ir realizando un producto escalar con un kernel. El kernel recorrerá todas las neuronas de entrada y obtendremos una nueva matriz, la cual será una de las hidden layers.

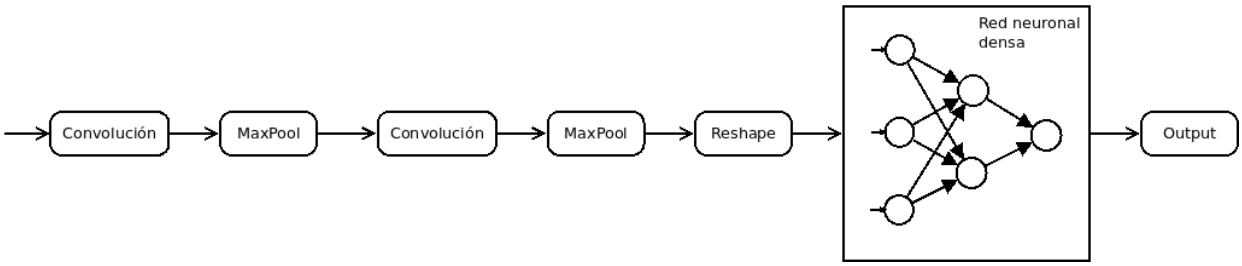


Figura 20: Arquitectura R.N.A. para nuestro problema.

La base de datos usada fue la misma que usamos en anteriores iteraciones sin ninguna modificación dado a flexibilidad de las redes convolucionales que no requiere de una extracción de características manual, por lo que no se puede mostrar la media y desviación típica de las áreas en una imagen como hicimos en anteriores iteraciones y hacerlo de cada píxel es inviable, por lo que la mejor forma de visualizar la media de la base de datos es con la siguiente figura 21

Figura 21: Imagen creada con la media de los colores de la base de datos.

5.5.2. Resultados.

Los resultados no han sido muy buenos a pesar de probar con varios modelos como se puede ver en las tablas 40, 41, 42, 43, 44, 45, 46, 47, 48, y 49, y comprobar el correcto funcionamiento de nuestro código, lo que nos llevó a pensar que se debe a un fallo interno de la librería de Julia Flux sobre R.N.A. que usamos porque al usar su función de entrenamiento, la nueva R.N.A permanece dando exactamente los mismos resultados erróneos sin que podamos hacer nada como se muestra en la figura 22.

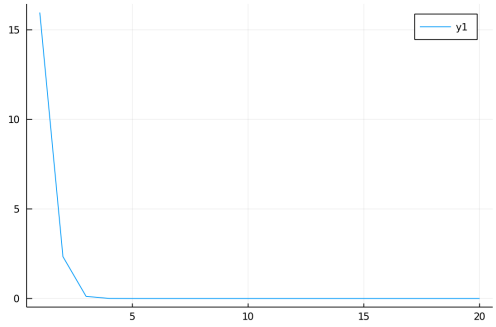


Figura 22: Errores R.N.A.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
1º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=16*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=16*3, canal salida=32*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=32*3, canal salida=32*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=1536, salida=1, función transferencia=sigma		

Tabla 40: 1º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
2º Modelo	Convolución: filtro=(5,5), canal entrada=3, canal salida=16*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(5,5), canal entrada=16*3, canal salida=32*3		
	MaxPool=(2,2)		
	Convolución: filtro=(5,5), canal entrada=32*3, canal salida=32*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=384, salida=1, función transferencia=sigma		

Tabla 41: 2º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
3º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=32*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(5,5), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(5,5), canal entrada=64*3, canal salida=64*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=3074, salida=1, función transferencia=sigma		

Tabla 42: 3º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
4º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=16*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=16*3, canal salida=32*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=6144, salida=1, función transferencia=sigma		

Tabla 43: 4º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
5º Modelo	Convolución: filtro=(5,5), canal entrada=3, canal salida=16*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(5,5), canal entrada=16*3, canal salida=32*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=3456, salida=1, función transferencia=sigma		

Tabla 44: 5º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
6º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=32*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=12288, salida=1, función transferencia=sigma		

Tabla 45: 6º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
7º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=16*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=16*3, canal salida=32*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=64*3, canal salida=64*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=768, salida=1, función transferencia=sigma		

Tabla 46: 7º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
8º Modelo	Convolución: filtro=(4,4), canal entrada=3, canal salida=32*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(4,4), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(4,4), canal entrada=64*3, canal salida=128*3		
	MaxPool=(2,2)		
	Convolución: filtro=(4,4), canal entrada=128*3, canal salida=128*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=192, salida=1, función transferencia=sigma		

Tabla 47: 8º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
9º Modelo	Convolución: filtro=(3,3), canal entrada=3, canal salida=32*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=64*3, canal salida=64*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=1536, salida=1, función transferencia=sigma		

Tabla 48: 9º modelo de R.N.A. convolucionales usado y sus resultados.

R.N.A. Convolucionales	Capas	Precisión máxima entrenamiento	Precisión máxima test
10º Modelo	Convolución: filtro=(4,4), canal entrada=3, canal salida=32*3	50.04	42.42
	MaxPool=(2,2)		
	Convolución: filtro=(4,4), canal entrada=32*3, canal salida=64*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=64*3, canal salida=128*3		
	MaxPool=(2,2)		
	Convolución: filtro=(3,3), canal entrada=128*3, canal salida=128*3		
	MaxPool=(2,2)		
	reshape: 3D -¿2D		
	Dense: Neuronas entrada=384, salida=1, función transferencia=sigma		

Tabla 49: 10º modelo de R.N.A. convolucionales usado y sus resultados.

5.5.3. Discusión.

Tras ejecutar esta aproximación observamos que los resultados son peores comparados con el resto de las aproximaciones y como comentamos en el apartado anterior esto puede deberse a la librería Flux de Julia. Debido a esto también obtenemos los mismos valores de precisión máxima de test y entrenamiento para las distintas iteraciones.

Para una siguiente iteración podríamos probar con otra librería que nos diera unos mejores resultados.

6. Conclusiones.

Después de las distintas iteraciones vistas anteriormente, podemos sacar diferentes puntos clave del trabajo realizado. En primer lugar, como se puede comprobar en apartados anteriores, hemos tenido resultados aceptables, tal y como se tenía previsto en los objetivos

principales. Hemos visto que el problema principal ha sido solucionado correctamente e incluso tras añadir complejidad mediante la detección de ojos los resultados continúan siendo buenos. Por ello podemos concluir que este tipo de sistema es bastante viable a la hora de resolver este problema y, por tanto, problemas parecidos a este, ya sea detección de caras o de patrones por color sobre una imagen.

Para extraer las características nuestra primera idea fue utilizar los colores, ya que Papa Noel siempre se representa con un característico traje rojo, barba blanca y gorro rojo con adornos blancos, después de esto hay que dividir la imagen por regiones y de esas tomas las mas relevantes, ara las regiones optamos por ua cuadrícula de 7 filas por 3 columnas, de la que escogemos 3 celdas, una correspondiente a la barba, otra a la base del gorro y otra que incluye parte de la cara, dadas estas celdas, establecemos como valores las medias de los valores de rojo verde y azul de cada celda.

Para la segunda aproximación decidimos que, dado que tanto el gorro como la barba son blancos, no importa mucho la distribucion de rojo verde y azul si no que la media fuese alta (colores claros), así que pasamos de las tres medias a una media global y la desviácion tipica para cada una de esas dos celdas.

Para la tercera iteración decidimos ir más allá e intentar detectar que la imagen era un rostro humano, para esto pensamos en detectar ojos, ara lo cual necesitamos partir en muchas mas celdas las imágenes de modo que quede una distribución de celdas como en la anterior iteración par la cara pero en este caso para los ojos, de donde volveremos a sacar valores de 3 celdas en particular

En la cuarta iteración buscamos acelerar el proceso haciendo la media de los valores presentes en las celdas 1 y 3 de la iteración dos, de esta forma son menos datos a procesar, sin embargo observamos que al hacer esto los resultados, en general, empeoraban con respecto a iteraciones anteriores.

Los resultados parecen indicar que la mejor iteración fue la segunda iteración, con el modelo de 8 vecinos y modelo de 4 vecinos de kNN dando ambos resultados bastante similares en cuanto a las medias de precisión y F1 score, pero siendo la desviación típica de estos valores mas baja para 8 vecinos , sin embargo, a falta de un test estadístico no podemos afirmas con seguridad que estos resultados son efectivamente los mejores.

Como quinta y ultima iteración usamos una aproximación de deep-learning, la principal diferencia de esta iteración con respecto al resto es el uso de redes de neuronas convolucionales, esto es, a las imagenes se les aplica una serie de convoluciones y compresiones. Con respecto a los resultados de esta iteracion observamos que son peores que los de las anteriores, esto puede deberse a la arquitectura elegida o a los datos de entrada.

Por otro lado, una de las mayores dificultades a la hora de desarrollar el sistema ha sido la escasez de imágenes de calidad, ya que para que el sistema funcione perfectamente las fotos a analizar deben de ser de una cara de frente y ajustada a los bordes. A la hora de reunir este material había una gran cantidad de imágenes en las que las caras estaban de lado, y por lo tanto no eran válidas para el sistema y también había muchas otras en las que la cara era solo una pequeña parte y, al recortarla se perdía tanta resolución que se volvían

inservibles, es por esta razón que la BBDD que utilizamos cuenta con relativamente pocas imágenes, o al menos menos de las que nos habría gustado tener para entrenar y validar el sistema lo mejor posible.

7. Trabajo futuro.

A lo largo del desarrollo de esta práctica hemos pasado por diferentes iteraciones en las que resolvíamos el problema de forma distinta. Siempre que avanzamos una, intentábamos o bien mejorar el rendimiento del sistema y sus resultados (generalmente reduciendo el número de parámetros de entrada), o bien generando nuevas características (como detectar ojos en un rostro) para evitar falsos positivos que nada tenían que ver con Papá Noel.

A partir de este trabajo se puede reutilizar esta misma última característica para, por ejemplo, verificar en otros sistema que requieran de imágenes de personas si se trata de un rostro o no. Otro uso que pensamos que podría funcionar bien sería utilizarlo como primera capa de verificación en desbloqueo por detección facial en teléfono inteligentes. A mayores podríamos mejorar este subsistema de la práctica para que funcionase con vídeo y detectar seres vivos que tengan ojos en una grabación.

Si pensamos en esta aplicación a vídeo, podríamos usar el sistema para que por cada frame de un vídeo nos produzca un resultado positivo o negativo a un determinado problema. Aplicaciones a esto podrían ser su implementación a un sistema de CCTV para detectar personas encapuchadas o que lleven bien puesta la mascarilla o incluso para detectar el número de Santa Claus en un centro comercial en Navidad. Todo esto podríamos hacerlo mejorando la rapidez del sistema para que funcione a una velocidad lo suficientemente rápida para aplicarlo a un vídeo. En cada frame podríamos hacer un recorrido de la imagen buscando lo que queramos detectar de la misma forma que lo hacemos para detectar ojos en la tercera y posteriores iteraciones.

Si pensamos en la práctica completa, la detección de patrones en una imagen sirve para infinidad de cosas. Lo primero que se nos viene a la mente después de estos años de pandemia sería usarla para detectar si una persona lleva o no mascarilla. En este caso no usaríamos los colores (las mascarillas pueden ser de una infinidad de ellos), pero si podríamos recoger la media y desviación típica en esa región como usamos con la barba y el gorro de Santa Claus en las iteraciones 2 y 4 del sistema. Con esto podríamos permitir, por ejemplo, el acceso a una instalación de una persona con una foto de su rostro, hacer estadísticas en espacios cerrados de cuánta gente la lleva puesta... Si a mayores añadimos con mencionamos en el anterior párrafo que funcionase con vídeo, podríamos incluso hacer una cámara de vigilancia para comprobar el correcto uso de la mascarilla.

Otra mejora que podríamos implementar en un futuro al sistema sería la detección de Papá Noel en una imagen sin recortar. De esta forma tendríamos que pasar de decir si una foto es o no Santa Claus, a decir si está o no en ella. Esto sería algo más complejo, pero posiblemente con recorrer la imagen de la misma forma que hacemos para la detección de ojos podría ser suficiente para que funcionase (al menos en una primera iteración claro está). Esto lo podemos aplicar al mundo empresarial convirtiendo este sistema en una aplicación

móvil que lo detecte a partir de una foto.

Siguiendo con la aplicación al mundo real de este sistema, este puede ser modificado fácilmente cambiando los inputs del sistema (la Base de Datos) y las celdas de las que recogemos los datos (recordemos que no analizamos toda la imagen para ser más eficientes). Si cambiamos el dataset por fotos de objetos de una cierta fábrica válidos y otro de objetos que presentan algún problema de fabricación, podríamos hacer un sistema de detección de errores en cadena de montaje. Esto, sin duda, puede ser de gran ayuda en el control de calidad de una empresa y que no conlleve una gran cantidad de operarios revisando manual y repetitivamente la gran cantidad de productos que puede generar una cinta de producción.

Otro ejemplo relacionado con esto, sería su implementación a una panadería. Los cliente no quieren que el pan se encuentre o muy tostado o demasiado crudo. Esto se puede ver muy bien de forma visual, con lo que podríamos hacer lo mismo y detectar si algo se está saliendo del rango de aceptación para evitar que se nos pase el pan en el horno o sitios similares.

Finalmente, y de forma más imaginativa, pensamos en la incorporación de detección de patrones en imágenes al ámbito médico. Si pensamos en la dificultad de identificar la rotura de un hueso en una radiografía, no parece muy complicada. Ahora, si pensamos en ecografías o en detectar células cancerosas en una imagen, la cosa se complica en gran medida. Con el uso de estas herramientas de aprendizaje automático y, de forma parecida a lo que hemos hecho, podríamos tener un sistema que detectase ciertas patologías en imágenes médicas.

8. Bibliografía.

Referencias

[1] L. d. P. Cépeda Neira and S. Roncal Lázaro. Desarrollo de un sistema inteligente usando redes hopfield para el reconocimiento de rostros. 2017.

[2] J. C. C. Chisag, G. A. F. Lagla, G. S. V. Alvarez, J. A. C. Moreano, O. A. G. Pico, and E. M. I. Chicaiza. Utilización de recursos didácticos interactivos a través de las tic´s en el proceso de enseñanza aprendizaje en el área de matemática. 2017.

[3] M. del Pozo Baños. Sistema biométrico de detección facial sobre vídeo basado en patrones binarios locales aplicados sobre el color. 2009.

[4] A. C. L. R. Dilan Andrés Caro Barranco. Sistema inteligente para el registro de asistencia basado en procesamiento digital de imágenes y redes neuronales convolucionales. 2018.

[5] P. P. García García. Reconocimiento de imágenes utilizando redes neuronales artificiales. 2013.

[6] L. A. González Hernández and M. G. López Medina. Sistema inteligente para reconocimiento de rostros. 2016.

[7] A. M. Hernández, J. A. M. Covarrubias, and D. A. P. Guzmán. Sistema inteligente para detección de fatiga y distracción en conductores de camión de acarreo pesado en minería de cielo abierto. 2016.

[8] N. Llamas Llopis. Reconocimiento facial no supervisado en programas de televisión. 2017.

9. Glosario de términos y acrónimos.

Términos

aprendizaje por refuerzo

Técnica para determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de recompensa. 2

aprendizaje no supervisado

Técnica para deducir una función a partir de datos de entrenamiento. 2

aprendizaje supervisado

Técnica donde un modelo se ajusta a las observaciones sin conocimiento previo. 2

aprendizaje automático

Proceso mediante el cual se usan modelos matemáticos de datos para ayudar a un equipo a aprender sin instrucciones directas. 2, 3

cross validation

La validación cruzada (cross validation en inglés) es un procedimiento para evaluar el rendimiento de los modelos de aprendizaje. Los conjuntos de datos generalmente se dividen en una estrategia aleatoria o estratificada. La técnica de división puede variar y elegir según el tamaño de los datos y el objetivo final.. 8

data sets

Colección de registros de datos para el procesamiento informático. 2

inteligencia artificial

Capacidad de una computadora u otra máquina para realizar aquellas actividades que normalmente se cree que requieren inteligencia. 2

máquinas de soporte vectorial

Algoritmo que construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. 2

redes de neuronas artificiales

Representación del conocimiento mediante un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. 2, 6

redes semánticas

Representación del conocimiento lingüístico en la que los conceptos y sus interrelaciones se representan mediante un grafo. 2

Acrónimos

AA

Aprendizaje Automático. 2

ANN

Artificial Neural Network. 2

CV

Cross Validation. 8

IA

Inteligencia Artificial. 2

KNN

K-Nearest Neighbors. 2

MSV

Máquinas de Soporte Vectorial. 2

RNA

Redes de Neuronas Artificiales. 2

SVM

Support Vectorial Machine. 2