# DETECCIÓN DE MATRÍCULAS EN VEHÍCULOS

**Nicolás Martínez González - n.gonzalezm**
**J.Iván Folgueira Cabado - ivan.folgueira**
**Alejandro Costa Suárez - alejandro.costa.suarez ***

**Aprendizaje Automático**
**Universidade da Coruña**
**Curso 2019-2020**

# Índice

# Índice de figuras

# Índice de cuadros

# 1. Introducción

Las matrículas llevan usándose para identificar vehículos mucho antes de la invención de los automóviles, se usaron por primera vez en el año 1884 en carruajes de caballos en Canadá. Unos años más tarde, Francia las empezaría a usar en vehículos a motor y después otros países continuarían la tendencia. En la actualidad el número de vehículos en circulación se ha incrementado mucho y con ello las infracciones y la complejidad de tratar esta enorme cantidad de información. (1, p 4,5)

Una solución a este problema, y a otros muchos problemas derivados del gran número de vehículos, es el reconocimiento automático de matrículas que permite enviar sanciones a los propietarios que se salten las normas de circulación sin la necesidad de tener personal que se dedique especialmente a ello. Esta no es la única aplicación posible del reconocimiento automático ya que puede usarse para controlar el pago en una estación de servicio, control del flujo de tráfico y estudio de rutas, sistemas de control de parking, elevación automática de barreras de circulación, control de acceso a zonas urbanas o privadas abriendo las puertas de forma automática, análisis de rutas de transporte, pago automático en cabinas de peaje, etc. (2)

El problema de reconocimiento de matrículas para estos usos es el tener que realizarlo en tiempo real y sobre todo con la precisión correcta para no confundir una matrícula con otra, durante este proceso pueden producirse otros errores intermedios como la posibilidad de no encontrar la matrícula en la imagen o encontrarla incompleta de forma que no sea posible obtener el identificador. Para llevar a cabo el objetivo general de reconocimiento e identificación de matrículas debemos superar una serie de objetivos intermedios. Dada una imagen con al menos una matrícula en ella, reconocer donde está la placa de matrícula, de este proceso saldrán varias posibles zonas y el siguiente paso será clasificarlas para identificar si en la imagen existe una matrícula o no. Por último, se identificarán los números y letras de la matrícula. Trataremos de resolver este problema en varios pasos generales que se tratarán mediante diferentes métodos, primero partiremos de una imagen que contenga una matrícula, sacaremos varios posibles trozos de la imagen en base al color que podrían ser matrículas, los llamaremos candidatos, de estos, sacaremos una serie de características como medias de color y tamaños, mediante RRNNAA (Redes de Neuronas Artificiales), máquinas de soporte vectorial y otras técnicas diferenciaremos las que son matrículas de las que no.

Conseguir este objetivo tiene múltiples ventajas como las mencionadas

anteriormente de no necesitar personal que realice la tarea y al mismo tiempo poder realizarla con rapidez, en múltiples sistemas al mismo tiempo, también permite reducir costes ya que una vez instalado un sistema que se apoye en esta tecnología no son necesarios métodos alternativos para la misma tarea.

Existen múltiples métodos automáticos que ya permiten hacer esto, entre estos métodos están algunos similares al los que usaremos nosotros, como RRNNAA y Deep Learning, pero también otros muy diferentes que no hacen uso de inteligencia artificial, como identificación exclusivamente mediante color y dimensiones de las placas.

Esta memoria consta de varios apartados con el siguiente contenido:

- Introducción: esta introducción, se da contexto al trabajo y se explican varios conceptos generales.
- Descripción del problema: se explicará este problema con más detalle, así como el caso concreto que trataremos y nuestras restricciones.
- Análisis bibliográfico: en esta sección se introducirán trabajos de otros autores que han abordado este problema, tanto desde una perspectiva similar como haciendo uso de otros métodos.
- Desarrollo: se explicarán y analizarán en detalle todas las aproximaciones del trabajo.
- Conclusiones: conclusiones finales tras completar el trabajo.
- Trabajo futuro: se explica brevemente como podría continuar el trabajo a partir de este punto.
- Referencias: se indican las referencias citadas a lo largo de toda la memoria.

# 2. Descripción del problema

El problema a resolver es la identificación de caracteres de las matrículas de vehículos en base a imágenes de estos. Algunas de las restricciones que debemos tener en cuenta son:

- Todas las imágenes utilizadas tienen una y solo una placa de matrícula válida.
- La placa de matrícula no tiene siempre las mismas dimensiones ni formato, puede ser cuadrada, rectangular, distinto color de fondo o número de caracteres.
- Siempre pueden identificarse claramente tanto la matrícula como sus caracteres en la imagen.
- La posición de la matrícula en la imagen puede variar, a veces estará en el centro de la imagen, otras en una posición inferior, superior o lateral.
- Los vehículos fotografiados pueden contener otros elementos con características similares a matrículas pero que no lo son, dentro de estos elementos se engloban pegatinas decorativas, señales o placas identificativas que indican su velocidad o carga máxima, emblemas o logotipos de marcas, nombres o modelos de los vehículos y rótulos de publicidad.

Haremos uso de una base de datos de imágenes de vehículos en distintas condiciones por lo que debemos buscar o construir dicha base de datos, en este caso no construiremos una ya que existen multitud de ellas en internet y se ajustan a nuestras necesidades. La base de datos elegida (3) contiene imágenes de distintos tipos de vehículos, mayormente coches y camiones, separadas en distintos tipos según sus características, que serán útiles en las diferentes iteraciones de nuestro problema, algunas de las características son:

- El color de la placa de matrícula.
- El ángulo desde el que está tomada la imagen, recto desde la parte frontal o trasera, desde delante o atrás, pero desde un lateral, desde arriba o abajo, etc.
- El tamaño de la placa.
- Su forma, esta puede ser rectangular, rectangular alargada o cuadrada dependiendo del tipo de vehículo o país de origen.

- Su ubicación en la imagen, centro, superior, inferior, etc.

Las imágenes están separadas en conjuntos en base a estas características y algunos conjuntos mezclan varias de estas imágenes, tenemos varios grupos: donde solo se aprecia la vista trasera del vehículo y hay buenas condiciones de iluminación, con la vista delantera y trasera mezclada en buenas condiciones de iluminación, imágenes de noche o con malas condiciones de iluminación mezcladas con imágenes diurnas, conjuntos con diferencias en el color de fondo de las placas, imágenes tomadas desde distintos ángulos.

El problema se tratará en iteraciones, se empieza con un conjunto de imágenes sencillo, solo desde atrás, con buena iluminación y color de placa blanco, para posteriormente complicarlo con mayor variedad de datos.

# 3. Análisis bibliográfico

Algunas de las aproximaciones recientes en la materia se enfocan de manera similar, concretamente las dos primeras partes de un proceso de cinco: imagen de entrada, preprocesado de imagen, ubicación de la matrícula, separación de caracteres, reconocimiento de caracteres. En las dos primeras partes, generalmente se utilizan algunos métodos comunes, como convertir la imagen a escala de grises o transformación de intensidad. La función de los dos primeros pasos es ayudar a buscar la matrícula de una manera más precisa, S. Li y col. (1) abordan el problema ajustando la distancia del vehículo y su orientación para que sea siempre el mismo, a continuación ajustan el brillo y contraste de la imagen en escala de grises. Reducen el ruido de la imagen resultante mediante un filtro medio y terminan el preprocesado transformando la imagen a un estado binario. Para localizar la placa de matrícula eliminan las zonas blancas que no tengan un tamaño superior a un umbral y procesan las restantes como una matriz, obteniendo de esta forma unas coordenadas X-Y donde se sitúa la matrícula. Este sistema basado en matemáticas y no en aprendizaje automático demuestra unos buenos resultados en cuanto a identificación de la placa de matrícula son de una precisión del 95 % y en cuanto a segmentación y reconocimiento de caracteres de 88 %, el total del sistema es del 80 %.

Otros enfoques distintos son el de A. Akoum y col., (4), y el de S. Vajpayee, (5), ambos proyectos hacen uso de Machine Learning para identificar las matrículas y los valores pero de maneras diferentes. Los primeros hacen un enfoque en tres fases: identificación de una placa de matrícula en una imagen, extracción los caracteres de la placa y después identificarlos. Para llevar a cabo la tarea hacen uso de dos redes de neuronas artificiales de Hopfield, la primera identifica la placa y la segunda los valores que ha aprendido de un conjunto de entrenamiento. Para identificar la placa de matrícula se basan en la variación de color en escala de grises entre los valores y el fondo. Este desarrollo resulta en unos buenos resultados del 87 % de probabilidad de reconocer correctamente los caracteres de una matrícula, muy similar al visto en el sistema anterior sin IA.

El trabajo de S. Vajpayee en identificación de matrículas de la India (5) también hace uso de 2 modelos de ML, pero para la localización de la matrícula toma modelos de Deep Learning preentrenados, concretamente “haar cascade”. Una vez detectada la placa de matrícula hace un preprocesado de la imagen resultante, solo de la placa, para facilitar la identificación de los valores: aumenta o reduce el tamaño de la placa para que los valores tengan

un tamaño adecuado, convierte la imagen a binario y elimina el ruido. Para identificar los valores vemos una nueva interpretación no vista anteriormente que consiste en resaltar los bordes de los caracteres y quedándose con los de mayor tamaño para posteriormente procesarlos. Por último, para identificar los valores hace uso de una red neuronal, al igual que en el trabajo anteriormente citado.

En otro trabajo reciente y similar a los citados también se hace uso de sistemas preentrenados, concretamente YOLO Darknet v3 en el proyecto de U. Upadhyay y col. (6). En este caso se usa YOLO para identificar la placa de matrícula, con buenos resultados, posteriormente haciendo uso de redes neuronales y el algoritmo SIFT, que detecta y describe características locales de imágenes y localiza ciertos puntos clave, se encuentran las placas de matrículas. Por último identifican los caracteres en el resultado. Este sistema propuesto tiene una precisión del 90 %. Es el más preciso de los citados pero también el más complejo.

En nuestro caso particular no usaremos ninguna de las aproximaciones concretas de estos trabajos pero sí usaremos ciertos aspectos que son comunes a muchos de ellos, como puede ser el caso de transformar la imagen a escala de grises en preprocesado. Nuestra propuesta por lo tanto, en una primera aproximación, es a partir de un conjunto de imágenes procesar cada una convirtiéndola a escala de grises, de esta imagen sacar las partes altos niveles de blancos, pues será más probable que exista la matrícula en ellas, de estos candidatos filtramos los primeros por tamaño eliminado los que no superen unas dimensiones mínimas. De los candidatos restantes extraeremos una serie de características, medias de color, simetrías, etc. Con estos datos crearemos manualmente un conjunto de entrenamiento para la RRNNAA. En otras aproximaciones se usarán máquinas de soporte vectorial, aprendizaje basado en instancias (kNN) y Deep Learning.

# 4. Desarrollo

Para este apartado la forma de proceder será incremental, es decir, empezamos por una aproximación sencilla creando la base de datos para los conjuntos de entrenamiento, validación y test y haciendo un primer entrenamiento sencillo solo con redes neuronales. En los sucesivos incrementos, también llamados aproximaciones, nos centraremos en refinar el diseño de forma que se obtengan mejores resultados en el proceso de identificación, también se añadirán funciones adicionales. Cuando un método muestre unos resultados aceptables se trabajará con otro diferente y se analizarán ambos, comparando sus ventajas y desventajas, se podrán tratar de mejorar métodos ya completados.

## 4.1. Preparación de Aproximaciones

Para las sucecivas aproximaciones del problema observamos que nuestro sistema precisa de un módulo que preprocese las imágenes a fin de extraer las características de las posibles correspondencias con matrículas.

Para ello, hemos diseñado un sistema que en base a una imagen de entrada extrae el canal blanco, consideramos que es el que mejor se corresponde con el color de la placa de la matrícula en este momento, pues en el conjunto de datos inicial que estamos usando para esta iteración todas las matrículas tienen fondo blanco. A continuación, nos quedaremos solo con los mejores candidatos de los cuales extraeremos una serie de características, medias de color de los distintos canales ya que las matrículas son todas similares en este aspecto, para la base de datos. Por último, probaremos la base con una RRNNAA y analizaremos los resultados. Comenzamos extrayendo los 3 canales de color (rojo, verde y azul) y los comparamos con un valor umbral, de 0.48, 0.4 y 0.48 respectivamente; nos quedamos con las zonas de la imagen que sean mayores que estos valores en los 3 canales, como se puede ver en la Fig. 1. Todas las zonas de la imagen que cumplan esta característica serán consideradas zonas potenciales para ser una matrícula. Hacemos esta comparación, a partir de un umbral, dado que en las imágenes reales es complicado que existan blancos puros, por lo que un 'semi blanco' es lo más adecuado. De esta forma en una imagen podrían ser candidatas aquellas zonas amarillentas, con un tono gris claro o que, por incidencia de la luz, se vean más brillantes aun no siendo de color blanco.


Figura 1: Canal blanco de una imagen.

Este proceso nos deja con multitud de objetos posibles, el número varía mucho en función de la imagen, algunas presentan más de 500 y otras no llegan a 100, nos quedamos solo los de mayores dimensiones descartando los que tengan un tamaño menor que un valor mínimo, este tamaño es un área en píxeles igual o mayor que 4000px dado que las matrículas cumplen esta característica.

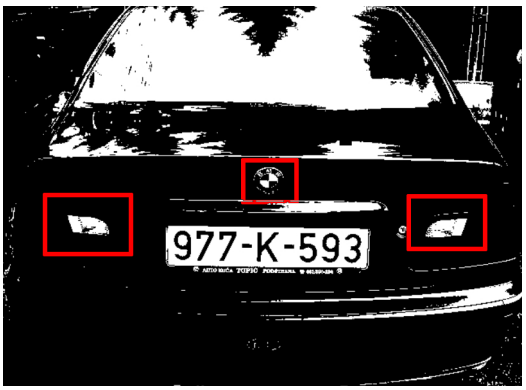


Figura 2: Canal blanco, algunos objetos a eliminar.

Algunos de los objetos que se eliminarían se han marcado manualmente en la Fig. 2. Esto ayuda a no tener que procesar objetos muy pequeños ya que una matrícula tiene unas dimensiones mínimas.


Figura 3: Imagen original, candidatos destacados.

Una vez identificados los objetos candidatos superponemos un recuadro sobre sus coordenadas en la imagen inicial, Fig. 3, en este punto, una vez eliminados los candidatos de menor tamaño, nos quedamos con entre 3 y 5 objetos por imagen de media llegando a 7 en casos extremos.
En este punto el programa muestra por pantalla cada posible matrícula, cada recuadro, y solicita una verificación de si lo es o no, la cual confirmamos manualmente mediante la terminal. Este proceso se repite para cada candidato en cada imagen y después con otra imagen del conjunto de la BBDD de imágenes. De esta manera, para cada objeto identificado guardamos en un fichero la media aritmética de sus valores RGB y un 1 si se trata de una matrícula o un 0 si no lo es, esta última será la salida deseada de nuestro sistema.

Una vez realizado este proceso con todos los objetos obtenemos nuestra base de datos, una matriz de cuatro columnas que procesaremos usando una red de neuronas artificiales. Para este proceso las tres primeras columnas de la matriz serán los valores de entrada (medias de los valores RGB) y la cuarta columna, las salidas deseadas. Mediante una adaptación del ejemplo de clasificación que se nos proporciona hemos diseñado una función que entrena las RNA durante varios ciclos y mide su precisión en los conjuntos de entrenamiento, validación y test y también su desviación típica en los mismos. De esta primer paso no obtenemos datos concretos, solo comprobamos que la base de datos y los atributos elegidos son adecuados para la resolución del problema y, que por lo tanto, vamos por buen camino.

Como conclusión, matizar que en este proceso de preparación la red identifica correctamente la mayoría de los patrones, pese a tener mucho margen de mejora. Este margen lo abordaremos en las sucesivas aproximaciones, la

mejora podría darse con una ampliación de los atributos de la base de datos para tener más información sobre el dominio, nuevas configuraciones de los distintos parámetros y nuevas arquitecturas para la red ya que en este punto usamos los valores por defecto. También se puede abordar el problema con otros métodos de IA.

## 4.2. Primera Aproximación

En esta aproximación mejoraremos la base de datos y reentrenaremos la RRNNAA, ya que los resultados previos pese a no ser definitivos tampoco eran ideales y podían ser mejorados añadiendo atributos a la base de datos y configurando mejor. Se añaden 2 atributos nuevos, media de color en simetrías y la relación entre el alto y ancho de las porciones candidatas. También usaremos esta nueva base de datos para entrenar una máquina de soporte vectorial. Por último, analizaremos los resultados obtenidos.

Para la creación de la nueva base de datos usamos el mismo método que antes pero ahora añadimos dos cambios. El primero consiste en añadir una función al código con la que, dado un posible candidato a matrícula, 'cortamos' el candidato en 2 mitades por el eje vertical, a partir de cada mitad sacamos la media de colores que componen el trozo, si ambas mitades tienen valores similares quiere decir que son simétricas. El nuevo atributo será el valor absoluto de la resta de los 2 valores. Cuanto más cercano a 0 sea mayor será la simetría.
Un ejemplo por el cual creemos que esto podría ser beneficioso viene dado por algunas de las observaciones al crear la primera base de datos, algunos de los candidatos detectados que no eran matrículas presentaban altos niveles de color blanco pero no en ambas mitades, concretamente las lunas traseras de los vehículos por los brillos del sol o sombras de árboles. En la bandeja trasera suelen presentarse objetos que no están en ambas partes, paraguas, chalecos reflectantes, etc.; modificando de esta forma la simetría, mientras que en las matrículas estos cambios son mucho menores, ya que solo hay ligeros cambios en el color negro por la anchura de unos números o letras frente a otros.
Para el segundo atributo, tomamos las medidas de ancho y alto para cada trozo, ya que son rectangulares. Dividimos los valores para obtener una relación que será el nuevo atributo. En este caso creemos que beneficiaría el aprendizaje ya que algunos de los casos considerados matrícula pero que no lo son, como el anteriormente citado de la luna trasera, tienen una relación muy diferente de la placa de matrícula la cual es siempre de dimensiones similares y, por lo tanto, la relación es similar en todos los casos que si sean matrícula.

Tras estos cambios a la base de datos, nos centramos en las modificaciones y entrenamientos a la RRNNAA y la creación de la SVM.

### 4.2.1. RRNNAA

Como entradas para la red de neuronas artificiales hemos empleado la base de datos de esta aproximación a excepción del último atributo de relación ancho-alto, dado que sin este la red ofrece resultados mejores. La división que hemos hecho de la base de datos para nuestro problema es de un 25 % para el conjunto de validación, un 25 % para el conjunto de test y la mitad restante para el conjunto de entrenamiento. Para cada iteración hemos establecido un límite máximo de 750 ciclos, y un límite de 250 ciclos sin mejorar como máximo dado que con esta configuración obtenemos mejores resultados.

| Arq. | Prec. Ent. | Desv. Ent. | Prec. Val. | Desv. Val | Prec. Test | Desv. Test |
|---|---|---|---|---|---|---|
| [4,2] | 82.885 | 2.45 | 81.959 | 4.54 | 83.311 | 4.16 |
| [5,3] | 82.919 | 2.41 | 82.972 | 3.29 | 82.094 | 4.42 |
| [5,2] | 82.709 | 1.51 | 81.891 | 4.41 | 84.067 | 4.01 |
| [6,2] | 85.461 | 2.25 | 81.554 | 3.34 | 85.202 | 3.25 |
| [6,3] | 82.752 | 3.12 | 80.405 | 3.31 | 82.229 | 4.68 |

Cuadro 1: RRNNAA, resultados entrenamiento, validación y test con 4 atributos

| Arq. | Prec. Ent. | Desv. Ent. | Prec. Val. | Desv. Val | Prec. Test | Desv. Test |
|---|---|---|---|---|---|---|
| [4,2] | 72.318 | 4.64 | 70.847 | 6.47 | 70.847 | 8.13 |
| [4,3] | 73.435 | 3.84 | 71.355 | 4.66 | 72.542 | 6.45 |
| [5,2] | 71.703 | 3.85 | 71.186 | 7.11 | 72.372 | 6.02 |
| [5,4] | 69.329 | 4.84 | 74.491 | 5.75 | 69.237 | 5.04 |
| [6,2] | 71.312 | 5.44 | 70.593 | 6.32 | 72.118 | 7.92 |

Cuadro 2: RRNNAA, resultados entrenamiento, validación y test con 5 atributos

Como se observa en las tablas anteriores, la red de neuronas artificiales alcanza mayor precisión y menor desviación cuando se le proporcionan 4 atributos, Fig. 1, que cuando se le proporcionan 5 , Fig. 2.
Por este motivo, la arquitectura de RRNNAA que hemos escogido es de 4 neuronas en la capa de entrada, 6 y 2 neuronas en las dos capas ocultas y 1 neurona en la capa de salida. El esquema final es el siguiente:

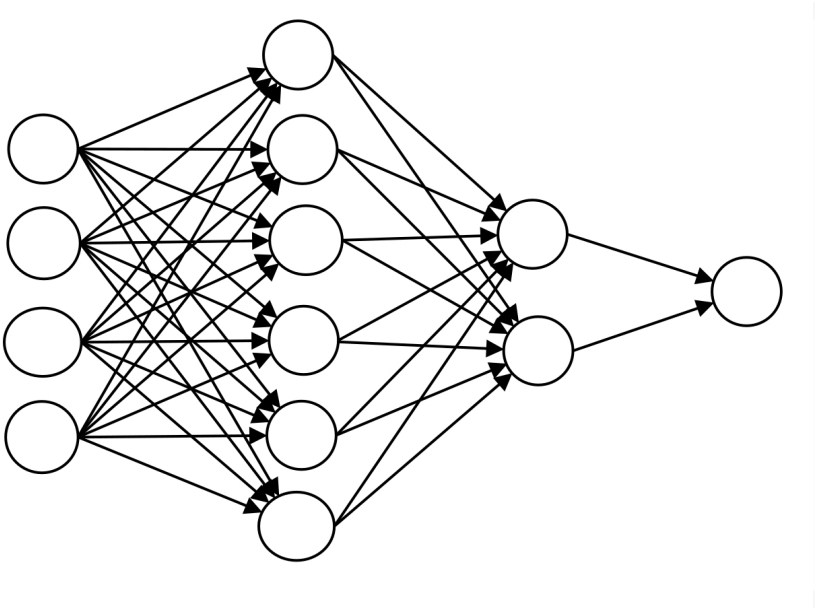

Figura 4: Esquema de la RNA.

### 4.2.2. SVM

Las entradas de la máquina de soporte vectorial son de un solo archivo que actuará como base de datos general para el problema, el archivo anteriormente generado tiene 5 entradas y una salida, las entradas son las medias de los 3 canales RGB por separado, la simetría de color y el ratio ancho-alto para cada candidato. No hacemos uso de técnicas como 'uno contra todos' ya que al tener solo dos clases posibles no es necesario, la configuración usada es la configuración estándar a la que iremos variando los parámetros, el kernel usado también es el por defecto que en este caso es 'rbf', ya que tras probar con otros es el que mejores resultados saca. Otros kernels no suben del 85 % de precisión mientras que con este conseguimos llegar al 91 %, como veremos más adelante.

Para los conjuntos de entrenamiento y test partimos la base de datos leída desde fichero en dos trozos, el 75 % de esta se asigna al conjunto de entrenamiento y el 25 % restante se asigna al de test. Las entradas se normalizan entre los valores máximos y mínimos. El número de ejecuciones que tomamos en este caso es de 60.

Ahora, realizamos varios entrenamientos cambiando los diferentes parámetros del sistema, modificaremos gamma y C, para controlar de esta forma la complejidad en la captura de los datos y evitar el sobrenetrenamiento no podemos tener gamma ni muy pequeño ni muy grande, la tolerancia a fallos la controlamos con C, con valores de C muy altos corremos el riesgo de sobreentrenar el sistema.

A continuación, se presentan en 2 tablas los 5 mejores resultados obtenidos de varias mediciones con modificaciones en algunos parámetros, se muestran las mejores combinaciones de parámetros con sus estadísticas. Tenemos dos tablas ya que en la primera de ellas, Fig. 3, hemos usado solo 4 atributos (media rojo, media azul, media verde, simetría) y en la segunda, Fig. 4, la base de datos completa con los 5 (media rojo, media azul, media verde, simetría, relación ancho-alto). Lo hicimos de esta manera para poder comparar con RRNNAA la cual presenta mejores resultados usando solo 4 atributos.

En lo que respecta a SVM, se obtienen mejores resultados con la base de datos completa, y también mejoran ligeramente con valores intermedios de ambos parámetros, siendo los extremos superiores testados 10 en ambos casos y los inferiores 1, de esta forma no sobreentrenamos el sistema, en cuanto a las desviaciones típicas siempre se encuentran en un margen de +-0.5.

En general, consideramos que los resultados obtenidos con máquinas de soporte vectorial son bastante buenos.

| | Gamma | C | Prec. Ent. | Prec. Test | Desv. Tip. Ent | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 84.88 | 83.8 | 1.36 | 3.17 |
| 2 | 4 | 10 | 93.09 | 90.09 | 1 | 3.19 |
| 3 | 4 | 4 | 85.8 | 84.7 | 1.3 | 3.43 |
| 4 | 4 | 10 | 87.3 | 84.86 | 1.4 | 3.16 |
| 5 | 6 | 2 | 85.7 | 84.7 | 1.4 | 3.04 |

Cuadro 3: SVM, resultados entrenamiento y test con los 4 primeros atributos.

| | Gamma | C | Prec. Ent. | Prec. Test | Desv. Tip. Ent | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 4 | 1 | 89.86 | 88 | 1.34 | 3.25 |
| 2 | 2 | 10 | 93.09 | 90.09 | 1 | 3.19 |
| 3 | 3 | 5 | 93.21 | 90.56 | 1 | 2.69 |
| 4 | 4 | 8 | 94 | 91.91 | 0.88 | 3.04 |
| 5 | 5 | 4 | 93.85 | 91.3 | 1.02 | 2.93 |

Cuadro 4: SVM, resultados entrenamiento y test con los 5 atributos.

### 4.2.3. Conclusiones de la aproximación

Como se puede a preciar en las tablas de resultados de máquinas de soporte vectorial, Fig. 4, y de redes de neuronas artificiales, Fig. 1, para este problema resulta más eficiente una máquina de soporte vectorial, incluso usando los mejores valores de la red. Creemos que la diferencia de eficacia entre ambos métodos radica en que las máquinas de soporte vectorial ofrecen, debido a su base matemática, una mayor exactitud y comportamiento similar lo que facilita entrenarlas, frente a la aleatoriedad de las RRNNAA y su inicialización aleatoria de los pesos. Usando los mismos parámetros y configuración en dos entrenamientos distintos, como hemos comprobado tanto para RRNNAA como para SVM, nada garantiza que las redes neuronales nos den resultados similares pero en SVM esto si es más común.

El trabajo futuro que planteamos para las siguiente aproximaciones será aplicar aprendizaje basado en instancias y Deep Learning a nuestro problema, así como intentar mejorar los resultados de las técnicas ya aplicadas.

## 4.3. Segunda Aproximación

Los resultados de la anterior aproximación pese a no ser perfectos los consideramos suficientemente aceptables como para no tratar de mejorar esos sistemas y, centrarnos en si haciendo uso de otros métodos obtenemos mejores resultados y también en ampliar el problema. En esta aproximación ampliaremos la base de datos con nuevas imágenes de matrículas con fondo azul, como las nuevas de taxis o VTCs, resolveremos el nuevo problema con los métodos vistos hasta ahora y también mediante aprendizaje basado en instancias (kNN) para el cual resolveremos de igual forma el problema antiguo (solo matrículas blancas, para comparar con los otros métodos en igualdad de condiciones), por último, discutiremos los resultados obtenidos por todos los métodos.

Primero implementaremos kNN con la base de datos sin ampliar, solo matrículas blancas, y compararemos como se comporta este sistema a la hora de resolver dicho problema con respecto a los vistos anteriormente.

### 4.3.1. kNN para matrículas blancas

Las entradas del clasificador son atributos leídos de un fichero, usaremos dos versiones del mismo: una con 4 entradas y otra con 5, para poder observar si su funcionamiento es el esperado y clasifica mejor con el atributo extra o como en RRNNAA ocurre al revés. Las entradas, al igual que en los casos anteriores son las medias de los 3 canales RGB por separado, la simetría de color y el ratio ancho-alto para cada candidato, siendo este último el que difiere de una base de datos a otra. En este caso al igual que en SVM no es necesario el uso de "uno-contra-todos" ya que tenemos solo dos clases posibles (matrícula o no matrícula), usaremos la configuración estándar proporcionada a la que haremos modificaciones en los valores de los parámetros y los algoritmos internos utilizados.

De esta base de datos se sacarán 2 conjuntos el 75 % es para entrenamiento y el 25 % restante se usará para test. Se entrenará durante 50 ciclos y se sacarán estadísticas de precisión de entrenamiento, test y desviación típica. Las entradas las normalizamos entre sus valores máximos y mínimos ya que de no hacerlo, debido a los rangos dispares de algunos atributos, unos podrían tener mucha mayor importancia que otros y esto no nos interesa.

Tras realizar varios entrenamientos tanto con 4 atributos como con los 5 variando tanto parámetros como algoritmos obtenemos las siguientes conclu-

siones:

- kNN con ponderación de vecinos (usando el inverso de la distancia) no mejora los resultados de kNN estándar, dónde la ponderación es uniforme para todos.
- El algoritmo a usar para calcular los vecinos cercanos, en este caso, no influye demasiado en el resultado, comparando 3 de ellos en múltiples ejecuciones (ball_tree, kd_tree y fuerza bruta), tanto fuerza bruta como ball tree muestran una precisión en test del 89-90 % mientras que kd tree nunca supera el 89 %, aún cambiando el tamaño. Al haber una diferencia tan pequeña dejamos este parámetro en automático.
- La métrica para calcular las distancias tampoco presenta grandes diferencias. Tras probar 3 métricas, minkowski, euclídea y la distancia de manhattan, tanto minkowski como manhattan alcanzan una media de 89-90 % mientras que la euclídea se queda, de media, un 0.6 % por debajo en precisión de test.
- Donde sí se observan la mayoría de cambios de precisión es cuando variamos el número de vecinos, como se puede ver en las tablas 5 y 6, cuyos valores se han obtenido con los mejores parámetros de entre los anteriormente mencionados (ponderación de vecinos uniforme, cálculo de vecinos cercanos "auto" y métrica de distancias minkowski), con un número bajo de vecinos se produce una clasificación peor y con un número alto, pese a no ser la peor, tampoco es óptima. Las mejores clasificaciones se producen cuándo el número de vecinos cercanos tomado es de entre 7-12 sin muchas diferencias.

Como se puede observar en las tablas los mejores resultados para kNN se dan con la base de datos al completo, y la mayoría de parámetros y algoritmos por defecto ya que cambiándolos no se presentan grandes mejoras y puede llegar a empeorar. Consideramos estos resultados bastante buenos aunque mejorables.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 78.02 | 3.98 |
| 2 | 3 | 88.35 | 1.63 | 81.67 | 4.15 |
| 3 | 5 | 87.79 | 1.41 | 82.59 | 4.01 |
| 4 | 7 | 87.07 | 1.31 | 83.35 | 3.83 |
| 5 | 10 | 85.92 | 1.49 | 84.43 | 3.93 |

Cuadro 5: kNN, resultados con diferentes números de vecinos para 4 entradas.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 87.66 | 4.56 |
| 2 | 3 | 93.26 | 1.06 | 88.47 | 3.623 |
| 3 | 5 | 92.66 | 1.17 | 89.83 | 3.52 |
| 4 | 7 | 92.18 | 1.02 | 90.0 | 3.54 |
| 5 | 10 | 91.61 | 0.91 | 89.79 | 3.34 |

Cuadro 6: kNN, resultados con diferentes números de vecinos para 5 entradas.

### 4.3.2. Conclusiones de los 3 métodos para matrículas blancas

Comparando estos resultados con los de las anterior aproximaciones podemos concluir que para este problema concreto el uso de RRNNA es innecesario ya que tanto SVM como kNN realizan el trabajo con mejores resultados y dadas sus características de forma más eficiente.

Como se puede ver en las tablas de los tres métodos con 4 atributos (el mejor caso usando RRNNAA) RRNNAA: 1, SVM: 3 y kNN: 5, los resultados aun así son muy similares para todos, lo que hace que nos decantemos más si cabe por SVM o kNN ya que su tiempo de ejecución y complejidad es menor.

Entre estos 2 y con la base de datos al completo, ya que en estos casos obtenemos mejores resultados con 5 atributos, nos quedaríamos con SVM ya que, si tenemos una base de datos muy grande, kNN requeriría de mucha más memoria al tener que almacenar los valores. Otro factor por el cual elegir SVM sobre el resto es el ruido, aunque en nuestra base de datos actual no exista apenas, si apareciese en el futuro kNN podría fallar en la clasificación.

### 4.3.3. Ampliación del problema: matrículas azules

Para esta ampliación del problema nos decantamos por tratar de reconocer matrículas con el fondo azul y los números en color blanco, aparte de las ya conocidas blancas y negras, ya que son otro tipo de matrículas comunes en otros países y, dada la nueva legislación de taxis y vehículos VTC, también en España.

La imágenes para este efecto no son de una base de datos ya construida sino que la construimos nosotros, todas las imágenes son lo más similares posibles a las usadas en el reconocimiento de las matrículas blancas, si alguna no cumplía estos criterios ha sido editada o descartada, esto es:

- Todas las imágenes son tomadas de día en buenas condiciones de iluminación.
- La fotografía ha sido tomada desde la parte de atrás de los vehículos, mayoritariamente, o desde la parte delantera, sin inclinaciones horizontales o verticales.
- En cada imagen existe una y sólo una placa de matrícula válida.
- Siempre pueden identificarse claramente tanto la matrícula como sus caracteres en la imagen.
- La posición de la matrícula en la imagen puede variar, a veces puede estar en el centro de la imagen, otras en la parte superior, inferior o lateral.
- Los vehículos fotografiados pueden contener otros elementos con características similares a matrículas pero que no lo sean, dentro de estos elementos se engloban pegatinas decorativas, logotipos de marcas, etc.
- Por último, todas las imágenes de las que se extraen las características tienen el mismo tamaño (640x480 px).

La extracción de características se realiza de igual manera que para las matrículas blancas y se extraen las mismas características, media de colores RGB por separado, simetría de color (cortando cada candidato en torno a su eje vertical) y ratio ancho-alto del candidato. Los umbrales para la extracción del color azul son (R<=0.4; G<=0.4; B>=0.4) mientras que el tamaño en px para considerarse candidato y eliminar de esta forma los más pequeños se mantiene en 4000 px. El resto del proceso es el mismo que para las matrículas blancas. Ver (4.1).

La nueva base de datos creada consta de todos los datos extraídos de las imágenes con matrículas blancas y, mezclados aleatoriamente, los datos recién extraídos de las azules. Cada línea del fichero es un candidato pudiendo ser matrícula (blanca o azul) o no matrícula.

### 4.3.4. Resultados RRNNAA

Como entradas para la red de neuronas artificiales hemos empleado la nueva base de datos que hemos confeccionado que incluye matrículas blancas y azules. La división que hemos hecho de la base de datos para nuestro problema es de un 25 % para el conjunto de validación, un 25 % para el conjunto de test y la mitad restante para el conjunto de entrenamiento.

Para cada iteración hemos establecido un límite máximo de 650 ciclos, y un límite de 100 ciclos sin mejorar como máximo dado que con esta configuración obtenemos mejores resultados.

Al aplicar esta técnica a la nueva base de datos los resultados son los siguientes:

| Arq. | Prec. Ent. | Desv. Ent. | Prec. Val. | Desv. Val | Prec. Test | Desv. Test |
|---|---|---|---|---|---|---|
| [3,5] | 73.77 | 3.6 | 75.03 | 7.65 | 75.25 | 3.87 |
| [4,4] | 74.55 | 4.39 | 73.6 | 7.31 | 73.31 | 5.90 |
| [5,3] | 75.33 | 2.54 | 74.82 | 5.73 | 72.30 | 4.87 |
| [6,2] | 75.65 | 2.68 | 75.39 | 4.42 | 74.53 | 4.06 |
| [7,5] | 74.34 | 2.76 | 74.03 | 6.56 | 76.61 | 4.97 |

Cuadro 7: RRNNAA, resultados entrenamiento, validación y test con 5 atributos

Como se aprecia en la tabla, la precisión del método alcanza su valor de precisión óptimo con una arquitectura de 7 y 5 neuronas en las dos capas ocultas. Si bien los resultados para esta base de datos no mejoran los de la original, suponen una cierta mejora con la base de datos anterior pese a tener dos tipos de matrículas en vez de uno sólo.

### 4.3.5. Resultados SVM

Las entradas continúan siendo los 5 atributos mencionados anteriormente, el conjunto de unas 300 entradas se divide en 2 conjuntos para entrenamiento y test, 75 % y 25 % respectivamente. Se entrena el sistema durante 50 ciclos haciendo variaciones en parámetros y kernels.

En cuanto a los kernels, usamos 4: RBF (radial basis function), lineal, polinomial con grado 3 y sigmoidal. Tanto RBF (tabla 8) como el polinomial (tabla 9) muestran buenos resultados aunque requiriendo el polinomial mayor tiempo de ejecución dada su complejidad, quedando lineal y sigmoidal notablemente por detrás de estos. Con kernel lineal nunca supera el 77 % de precisión dando igual gamma y/o C, para el sigmoidal es incluso peor, no llegando a superar el 69 %.

Las tablas presentadas a continuación muestran los resultados de los mejores kernels. Se indica como cambian al modificar los parámetros gamma y C. Los mejores resultados se dan con valores intermedios de estos, los rangos utilizados para gamma van desde 1 hasta 10, y lo mismo ocurre con el coeficiente de tolerancia a fallos, C. Los resultados haciendo uso de SVM se mantienen en un nivel aceptable pese a haber ampliado el espacio del problema.

| | Gamma | C | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 85.02 | 1.26 | 82.85 | 3.97 |
| 2 | 2 | 2 | 86.66 | 1.14 | 84.53 | 3.37 |
| 3 | 3 | 10 | 93.11 | 1.29 | 88.68 | 3.58 |
| 4 | 5 | 4 | 92.94 | 0.91 | 89.21 | 3.11 |
| 5 | 7 | 9 | 95.39 | 0.80 | 90.48 | 2.46 |
| 6 | 8 | 8 | 95.47 | 0.92 | 90.92 | 2.78 |
| 7 | 10 | 7 | 95.57 | 0.87 | 91.34 | 3.04 |

Cuadro 8: SVM con kernel RBF, resultados entrenamiento y test.

| | Gamma | C | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 3 | 10 | 89.75 | 1.23 | 86.70 | 3.26 |
| 2 | 4 | 1 | 88.20 | 1.26 | 85.43 | 3.82 |
| 3 | 6 | 9 | 95.08 | 0.83 | 90.87 | 3.15 |
| 4 | 7 | 7 | 93.00 | 1.06 | 87.87 | 3.75 |
| 5 | 8 | 2 | 92.04 | 1.04 | 89.14 | 3.63 |
| 6 | 9 | 9 | 93.6 | 1.08 | 90.14 | 2.9 |
| 7 | 10 | 3 | 93.01 | 0.99 | 88.85 | 2.93 |

Cuadro 9: SVM con kernel polinomial (con polinomio 3), resultados entrenamiento y test.

### 4.3.6. Resultados kNN

Las entradas en esta aplicación de kNN son los 5 atributos mencionados anteriormente, y de esta base de datos se sacarán 2 conjuntos: el 75 % es para entrenamiento y el 25 % restante se usará para test. Se entrenará durante 50 ciclos. Las entradas están normalizadas al igual que solo con matrículas blancas.

Tras realizar varios entrenamientos cambiando los parámetros llegamos a unas conclusiones similares a pruebas anteriores:

- kNN con ponderación de vecinos (usando el inverso de la distancia) produce un 100 % de precisión de entrenamiento no mejorando la precisión de test, dando igual el resto de variaciones (nº de vecinos, tipos de algoritmos, etc.).
- El algoritmo a usar para calcular los vecinos cercanos, al igual que en el caso anterior, no influye demasiado en el resultado, comparando 3 de ellos en múltiples ejecuciones (ball_tree, kd_tree y fuerza bruta), tanto fuerza bruta como ball tree muestran una precisión en test ligeramente superior a kd tree (1.2-1.4 % menos, de 88-89 % a 87 % en los mejores casos, mayor variación que solo con matrículas blancas). Dejamos dicho parámetro en automático ya que no elige el peor caso.
- La métrica para calcular las distancias no presenta diferencias. Tras probar 3 métricas(distancia Manhattan, Minkowski y euclídea) al contrario que el caso anterior, todas se quedan en 89-90 %.
- La mayor parte de los cambios de precisión se producen cuando variamos el número de vecinos, como se puede observar en la tabla 10, cuyos valores se han obtenido con los mejores parámetros de entre los previamente mencionados (ponderación de vecinos uniforme, cálculo de vecinos cercanos "auto" y métrica de distancias Minkowski), con un número bajo de vecinos se produce una clasificación peor y con un número alto también. El rango de vecinos que reportan las mejores clasificaciones se ha reducido con respecto al caso anterior de 7-12 a 6-9.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 87.73 | 3.69 |
| 2 | 3 | 93.37 | 1.17 | 87.17 | 3.15 |
| 3 | 5 | 92.09 | 0.96 | 89.12 | 2.69 |
| 4 | 7 | 91.42 | 1.16 | 89.51 | 3.21 |
| 5 | 10 | 90.77 | 0.89 | 87.97 | 2.94 |

Cuadro 10: kNN, resultados con diferentes números de vecinos para 5 entradas.

### 4.3.7. Conclusiones de la aproximación

Tras la ampliación del problema seguimos considerando poco adecuado el uso de RRNNAA, ya que tanto kNN como SVM logran mejores resultados, en especial SVM pese a ser ambos muy similares. En torno SVM es importante resaltar los resultados cuando se usa un kernel polinomial, puesto que las precisiones son las mismas pero el tiempo de computación para alcanzarlas es notablemente mayor. También es importante destacar el sobre entrenamiento que se produce cuando se usa ponderación con el inverso de la distancia en kNN.

El trabajo futuro partiendo de la situación actual podría ser intentar resolver el problema con Deep Learning, mejorar los resultados actuales o ampliar nuevamente el problema y pasar a reconocer matrículas en otras situaciones (con lluvia, niebla, de noche, etc.).

## 4.4. Tercera Aproximación

En las anteriores aproximaciones hemos mejorado los resultados y también ampliado el problema. Con la ampliación, mientras que los resultados de kNN y SVM se mantenían en buenos porcentajes, RRNNAA nunca llegaba a ser competitiva. Para esta aproximación intentaremos mejorar los resultados en general y especialmente RRNNAA. Para ello realizaremos cambios en como se preprocesan las imágenes ya que al introducir matrículas azules hemos observado que el procesamiento inicial para sacar características las perjudicaba. También ampliaremos la base de datos con más, aunque no muchas más, imágenes con matrículas azules siguiendo el criterio de selección de imágenes indicado en el apartado (4.3.3).

### 4.4.1. Revisión de la base de datos y nuevo atributo

Como mencionamos arriba, al introducir matrículas azules el proceso de selección de candidatos no era del todo adecuado, por lo que lo hemos mejorado. Hemos retocado los umbrales de color para las matrículas azules, no cambia la exclusión de los candidatos más pequeños cuyo umbral sigue en 4000px.

Con estos cambios el proceso para la selección de candidatos queda de la siguiente manera: a partir de una imagen se sacan diversos candidatos blancos (canalRojo>=0.48, canalVerde >= 0.48, canalAzul >=0.4) y azules (canalRojo <=0.48, canalVerde <= 0.48, canalAzul>=0.4), de los candidatos resultantes nos quedamos aquellos con un tamaño superior a 4000px. Para cada candidato sacamos diversas características que pasarán a formar la base de datos: media de color rojo, azul y verde, simetría de color en el eje vertical, relación ancho-alto y, a mayores, el nuevo atributo diferencia de color centro-borde. Cambian los valores umbrales el proceso se mantiene igual.

Este nuevo atributo consiste en, para cada candidato, "recortar" el borde del mismo y sacar su media de color, hacemos lo mismo con el centro y comparamos ambas medias. Esto se basa en que las matrículas no tienen sus números o letras pegados al borde de la placa, hay un espacio pequeño del color de fondo entre medias, algo que no ocurre con otro tipo de elementos. En nuestro caso el borde seleccionado es el 15 % exterior del candidato mientras que consideramos centro todo lo demás. Una vez hechas las medias el nuevo atributo será el valor de la diferencia entre ambas medias.

### 4.4.2. Resultados RRNNAA

Como entradas para la red de neuronas artificiales hemos empleado la base de datos actualizada y con el nuevo atributo. La división de la base de datos para nuestro problema es de un 20 % para el conjunto de validación, un 20 % para el conjunto de test y el resto para el conjunto de entrenamiento.

Para cada iteración hemos establecido un límite máximo de 1000 ciclos de entrenamiento, y un límite de 100 ciclos sin mejorar validación.

Al aplicar esta técnica a la nueva base de datos los resultados son los siguientes, los resultados que se muestran son los medios para 50 ejecuciones:

| Arq. | Prec. Ent. | Desv. Ent. | Prec. Val. | Desv. Val | Prec. Test | Desv. Test |
|---|---|---|---|---|---|---|
| [3,5] | 86.65 | 1.5 | 86.45 | 2.88 | 86.18 | 3.25 |
| [4,4] | 86.69 | 1.56 | 86.40 | 3.01 | 86.22 | 3.21 |
| [5,3] | 86.59 | 1.64 | 86.37 | 3.20 | 86.17 | 3.13 |
| [6,3] | 86.57 | 1.63 | 86.43 | 3.25 | 86.10 | 3.16 |
| [7,5] | 86.55 | 1.58 | 86.32 | 3.16 | 86.18 | 3.15 |

Cuadro 11: RRNNAA, resultados entrenamiento, validación y test con 5 atributos

Como se aprecia en la tabla, la precisión del método no cambia mucho entre arquitecturas pero si con respecto a la aproximación anterior que mejoran los resultados un 10 %. Es importante matizar que son los mejores resultados de RRNNAA hasta este momento pese a estar clasificando 2 tipos de matrículas en vez de uno solo.

### 4.4.3. Resultados SVM

Las entradas son las mismas que para RRNNAA, aquí la división entre entrenamiento y test es de 75 % y 25 % respectivamente. Se entrena el sistema durante 50 ciclos haciendo variaciones en parámetros y kernels.Las entradas están normalizadas.

En cuanto a los kernels, usamos 4: RBF (radial basis function), lineal, polinomial (con grados 2 y 3) y sigmoidal. Tanto RBF (tabla 13) como el polinomial (tabla 12) muestran buenos resultados, aunque requiriendo el polinomial de grado 3 mayor tiempo de ejecución dada su complejidad. En cuanto al polinomial de grado 2, lo mencionamos en esta aproximación dado que en la anterior sí se quedaba muy por detrás del de grado 3, los resultados en este caso son relativamente similares a los observados en la tabla 12 pero bajando de media un 1 % la precisión y con mucho menor tiempo de ejecución. Quedando lineal y sigmoidal notablemente por detrás del resto, otra vez. Con kernel lineal el pico de precisión esta en el 85 %, para el sigmoidal es incluso peor, no llegando a superar el 80 % en ninguna ocasión. Aun así, son unos resultados mejores que los de otras aproximaciones en estos dos casos.

Las tablas presentadas a continuación muestran los resultados de los mejores kernels. Se indica como cambian al modificar los parámetros gamma y C. En este caso SVM se mantiene en un nivel aceptable, no empeora pero tampoco se ve beneficiado por el nuevo atributo.

| | Gamma | C | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 87.73 | 0.89 | 87.49 | 2.40 |
| 2 | 2 | 2 | 91.03 | 1.77 | 89.34 | 2.49 |
| 3 | 3 | 7 | 91.70 | 1.85 | 89.76 | 2.49 |
| 4 | 5 | 8 | 92.98 | 1.93 | 90.45 | 2.43 |
| 5 | 6 | 5 | 93.27 | 1.92 | 90.63 | 2.42 |
| 6 | 7 | 10 | 93.85 | 1.88 | 90.90 | 2.40 |
| 7 | 10 | 3 | 94.17 | 1.86 | 90.96 | 2.36 |

Cuadro 12: SVM con kernel polinomial (con polinomio 3), resultados entrenamiento y test.

| | Gamma | C | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 3 | 4 | 87.51 | 0.85 | 87.43 | 2.37 |
| 2 | 4 | 1 | 89.83 | 1.64 | 88.62 | 2.63 |
| 3 | 6 | 10 | 92.10 | 2.14 | 90.34 | 2.72 |
| 4 | 7 | 5 | 93.07 | 2.22 | 90.85 | 2.60 |
| 5 | 8 | 2 | 94.11 | 2.26 | 91.19 | 2.46 |
| 6 | 9 | 9 | 94.60 | 2.31 | 91.30 | 2.40 |
| 7 | 10 | 3 | 94.85 | 2.32 | 91.35 | 2.38 |

Cuadro 13: SVM con kernel RBF, resultados entrenamiento y test.

### 4.4.4. Resultados kNN

Las entradas en esta aplicación de kNN son las mismas que en los 2 métodos anteriores, extraemos 2 conjuntos: el 75 % es para entrenamiento y el 25 % restante se usará para test. Se entrenará durante 50 ciclos. Las entradas están normalizadas.

Tras realizar varios entrenamientos cambiando los parámetros llegamos a unas conclusiones similares a pruebas anteriores:

- kNN con ponderación de vecinos (usando el inverso de la distancia) produce un 100 % de precisión de entrenamiento no mejorando la precisión de test, dando igual el resto de variaciones (nº de vecinos, tipos de algoritmos, etc.).
- El algoritmo a usar para calcular los vecinos cercanos no influye en el resultado, se han comparando 3 de ellos en múltiples ejecuciones (ball_tree, kd_tree y fuerza bruta). En la anterior aproximación había diferencias mínimas, en esta no son apreciables. Dejamos dicho parámetro en automático.
- Al igual que en la anterior aproximación la métrica para calcular las distancias no presenta grandes diferencias. Tras probar 3 métricas(distancia Manhattan, Minkowski y euclídea), todas se quedan en 89-90 %.
- Al contrario que en la anterior aproximación, no se producen tampoco grandes diferencias al variar el número de vecinos, como se puede ver en las tablas a continuación.

En la tabla se muestran los mejores resultados con los valores de kernel RBF con distancia de Minkowski.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 89.26 | 2.61 |
| 2 | 3 | 96.45 | 2.57 | 90.14 | 2.57 |
| 3 | 5 | 95.25 | 2.52 | 90.51 | 2.50 |
| 4 | 9 | 94.05 | 2.39 | 90.51 | 2.60 |
| 5 | 13 | 93.17 | 2.43 | 90.38 | 2.58 |

Cuadro 14: kNN, resultados con diferentes números de vecinos para 6 entradas.

### 4.4.5. Conclusiones de la aproximación

Con el nuevo atributo de la base de datos RRNNAA sufre una muy buena mejora en los resultados pero se mantiene por debajo del resto de métodos, ya que tanto kNN como SVM logran mejores resultados. Aun así no usaríamos RRNNAA porque su precisión es menor, de entre los otros dos métodos en este caso no hay grandes diferencias y cualquiera de ellos cumple la función.

Para el trabajo futuro partiendo de la situación actual, se podría intentar resolver el problema con Deep Learning, tratar de mejorar nuevamente los resultados con nuevos atributos para la base de datos o ampliar nuevamente el problema y pasar a reconocer más tipos de matrículas o en otras situaciones (con lluvia, niebla, de noche, etc.).

## 4.5. Cuarta Aproximación

En las anteriores aproximaciones hemos mejorado los resultados y también ampliado el problema. En la anterior aproximación, mientras que los resultados de kNN y SVM se mantenían en buenos porcentajes, RRNNAA sufría una importante mejora. En esta aproximación ampliaremos el problema ligeramente (no hemos encontrado muchas imágenes válidas) con fotos con peor visibilidad, nocturnas principalmente, algunas con lluvia o donde el vehículo está mojado, etc.

Compararemos nuevamente los resultados de los tres métodos.

### 4.5.1. Ampliación del problema

Para esta aproximación del problema hemos incorporado imágenes tomadas con peor iluminación y con menor calidad. En este caso, los atributos que usaremos serán los anteriores (media de colores, grado de simetría, relación entre alto y ancho) junto con la diferencia entre la media del borde del candidato y su interior que añadimos en la aproximación anterior.

Para extraer las características de estas nuevas imágenes al ser, en general más oscuras hemos añadido otros umbrales, a mayores de los que ya teníamos para blancas y azules. Estos umbrales nos permiten extraer candidatos dada la "oscuridad" o "amarillez" de algunas imágenes en las que los vehículos están iluminados por farolas o la luz de los semáforos, entre otros. Estos nuevos umbrales son los siguientes: (canalRojo >=0.48, canalVerde >= 0.55, canalAzul<=0.4)

### 4.5.2. Resultados RRNNAA

Las entradas para la red de neuronas artificiales en esta aproximación son la base de datos confeccionada anteriormente. Para ello hemos designado un 20 % de la base de datos para el conjunto de validación, un 20 % para el de test y el 60 % restante para el de entrenamiento. El límite de ciclos de entrenamiento que hemos determinado es 1000, además de un límite de 100 para el número de ciclos sin mejorar.

Los resultados son los siguientes, son los medios para 50 ejecuciones:

| Arq. | Prec. Ent. | Desv. Ent. | Prec. Val. | Desv. Val | Prec. Test | Desv. Test |
|---|---|---|---|---|---|---|
| [3,5] | 86.15 | 1.66 | 86.25 | 3.42 | 85.85 | 3.30 |
| [4,4] | 86.10 | 1.69 | 86.16 | 3.35 | 85.70 | 3.34 |
| [5,3] | 86.14 | 1.79 | 86.12 | 3.34 | 85.64 | 3.26 |
| [6,3] | 86.15 | 1.80 | 86.13 | 3.27 | 85.68 | 3.27 |
| [7,5] | 86.18 | 1.77 | 86.07 | 3.22 | 85.63 | 3.30 |

Cuadro 15: RRNNAA, resultados entrenamiento, validación y test con 5 atributos

Tal y como se observa en la tabla anterior los resultados de esta aproximación son ligeramente más bajos en relación con la anterior. Cabe destacar que en este caso el tipo de imágenes añadidas a la base de datos poseen peor iluminación y cierta distorsión.

### 4.5.3. Resultados SVM

Igualmente que con la red de neuronas artificiales las entradas son la misma base de datos y también se normalizan. En este caso la segmentación de las entradas es de un 75 % para el conjunto de entrenamiento y un 25 % para el conjunto de test. El sistema se entrena durante 50 ciclos y se producen modificaciones en los parámetros y kernels.

En esta aproximación continuamos comparando los 4 kernels: RBF, lineal, polinomial (con grados 2 y 3) y sigmoidal. En lineas generales la precisión de este método apenas se ve afectada por el aumento de la base de datos tal y como se muestra en la tabla 16 la nueva aproximación solo disminuye un 1 % la precisión, siendo los resultados más notables los alcanzados con el kernel RBF. En cuanto al resto de kernels, como era de esperar, el polinomial da unos resultados similares pero requiriendo de mucho mayor tiempo de ejecución. Los kernels sigmoidal y lineal se quedan muy por detrás de sus competidores al igual que en las aproximaciones anteriores (10-15 % peor).

La siguiente tabla presenta los valores de SVM con un kernel RBF para cada valor de gamma y C. En esta situación se cumple que SVM apenas se ve influido significativamente por la nueva aproximación.

| | Gamma | C | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|---|
| 1 | 3 | 3 | 90.42 | 2.00 | 88.56 | 2.81 |
| 2 | 4 | 2 | 91.20 | 2.22 | 89.17 | 2.89 |
| 3 | 6 | 4 | 92.39 | 2.37 | 89.94 | 2.81 |
| 4 | 7 | 8 | 92.88 | 2.40 | 90.22 | 2.75 |
| 5 | 8 | 7 | 93.18 | 2.40 | 90.36 | 2.70 |
| 6 | 9 | 5 | 93.42 | 2.41 | 90.48 | 2.66 |
| 7 | 10 | 9 | 93.74 | 2.42 | 90.59 | 2.61 |

Cuadro 16: SVM con kernel RBF, resultados entrenamiento y test.

### 4.5.4. Resultados kNN

Las entradas son iguales que para el resto de técnicas de esta aproximación, siendo la segmentación de las mismas de un 75 % para el conjunto de entrenamiento y un 25 % para el conjunto de test. Las entradas del sistema están normalizadas y el número de ciclos de entrenamiento en este caso es de 50.

Tras realizar varios entrenamientos cambiando los parámetros llegamos a unas conclusiones similares a pruebas anteriores:

- kNN con ponderación de vecinos (usando el inverso de la distancia) produce un 100 % de precisión de entrenamiento no mejorando la precisión de test.
- El algoritmo a usar para calcular los vecinos cercanos no influye en el resultado, hemos comparado los 3 de siempre (ball_tree, kd_tree y fuerza bruta) y continuamos a no ver diferencias significativas.
- Lo mismo ocurre con la métrica para calcular las distancias. Tras probar las 3 métricas(distancia Manhattan, Minkowski y euclídea), todas se quedan en 89 %.
- Una vez alcanzamos un número de vecinos medio tampoco se producen mejoras.

En la tabla se muestran los mejores resultados con los valores de kernel RBF con distancia de Minkowski y distancia euclídea.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 88.52 | 2.29 |
| 2 | 3 | 96.25 | 2.71 | 89.47 | 2.20 |
| 3 | 5 | 94.84 | 2.75 | 89.88 | 2.28 |
| 4 | 9 | 93.46 | 2.64 | 89.87 | 2.30 |
| 5 | 13 | 92.51 | 2.65 | 89.61 | 2.37 |

Cuadro 17: kNN, resultados con diferentes números de vecinos para 6 entradas, Minkowski.

| | Nº vecinos | Prec. Ent. | Desv. Tip. Ent | Prec. Test | Desv. Tip. Test |
|---|---|---|---|---|---|
| 1 | 1 | 100 | 0.0 | 89.17 | 1.98 |
| 2 | 4 | 95.52 | 2.72 | 89.79 | 1.96 |
| 3 | 6 | 94.57 | 2.63 | 89.87 | 1.91 |
| 4 | 8 | 93.97 | 2.53 | 89.92 | 2.07 |
| 5 | 12 | 93.07 | 2.48 | 89.88 | 2.15 |

Cuadro 18: kNN, resultados con diferentes números de vecinos para 6 entradas, euclídea.

### 4.5.5. Conclusiones de la aproximación

Aun añadiendo una cierta cantidad de nuevos candidatos todos los sistemas aguantan el tipo perfectamente, dicho esto RRNNAA sigue sin parecernos una mejor opción, nos quedaríamos con kNN o SVM.

Para el trabajo futuro partiendo de la situación actual, se podría intentar resolver el problema con Deep Learning, tratar de mejorar nuevamente los resultados con nuevos atributos para la base de datos o ampliar nuevamente el problema y pasar a reconocer más tipos de matrículas.

## 4.6. Quinta Aproximación

En las anteriores aproximaciones hemos trabajado con los métodos RRNNAA, SVM y kNN, haciendo uso de un preprocesado para extraer características y usar estas como entrada para los distintos métodos. En esta aproximación usaremos métodos basados en Deep Learning para procesar los candidatos en vez de extraer características. Los métodos que usaremos son RRNNAA, formadas por perceptrones multicapa, y redes de neuronas convolucionales (CNN).

Compararemos los resultados de ambos métodos.

### 4.6.1. Preparación Deep Learning

Para la preparación del problema, puesto que usaremos las entradas directamente sin realizar extracción de características, necesitamos guardar las imágenes candidatas. Guardamos las imágenes y, en un fichero de texto, las salidas del sistema que serán booleanas para indicar si el trozo candidato correspondiente es o no matrícula.

El procesado de las imágenes es el mismo, con los mismos umbrales y métodos pero en lugar de separar características exportamos el "trozo" como tal a un fichero.

A continuación adaptamos los diferentes códigos propuestos para crear ambos sistemas.

### 4.6.2. Resultados CNN

Para CNN las entradas del sistema serán las imágenes candidatas, a estas se les cambia el tamaño para que sea cada píxel una entrada de la red, en este caso las dimensiones elegidas son 75x100px. Optamos por estas dimensiones ya que de esta manera algunos candidatos escalan hacia arriba y otros hacia abajo.

A continuación, transformamos las entradas, ya con los nuevos tamaños, a una matriz HWCN con la función proporcionada. Al tener un dataset para entrenamiento de unas 420 imágenes las dividimos en batches de tamaño 128. Para test no lo hacemos puesto que tenemos un conjunto de unas 170 imágenes y no lo consideramos necesario.

Una vez procesadas las entradas creamos el modelo, este alternará capas convolucionales y de maxpool, al contrario que el código proporcionado y dado que nuestra salida es categórica no usaremos una capa de softmax.

Por último, ejecutamos el sistema para ver los resultados. Estos han sido singularmente buenos, en los primeros ciclos de entrenamiento la precisión era relativamente baja, del 50 %, pero cuanto más se entrenaba mejores resultados obtenemos. El pico del entrenamiento y test se alcanza en el ciclo 43 con una precisión en entrenamiento del 99,41 % y una precisión en el conjunto de test del 97,96 %.

A falta de ver como se comporta el perceptrón multicapa, esta es una muy buena opción para resolver este problema. Mejora en bastante a los modelos no profundos analizados anteriormente en un 7 %, lo cual consideramos bastante.

### 4.6.3. Resultados Perceptrón Multicapa

En este caso las entradas y las salidas son las mismas que para Deep Learnig pero con un matriz en el procesamiento. Este matiz es que en vez de transformar la imagen haciendo uso de la función HWCN, convertimos la imagen en un vector que la represente estando la imagen en escala de grises, hacemos esto último por simplicidad.

Los conjuntos en este caso son 3 entrenamiento validación y test al contrario que antes, y la arquitectura del sistema es una red neuronal de 3 capas ocultas (15,10,5 neuronas respectivamente). Las entradas son una por cada píxel y una única salida que indica si el candidato es o no matrícula. Para la separación de las entradas en los 3 conjuntos hacemos uso del mismo sistema que en el caso de RRNNAA estándar, holdout.
El número de ciclos que hemos designado a entrenamiento es de 1000 y hemos establecido un máximo de 100 ciclos sin mejorar validación.
Para la división de la entrada, hemos asignado un 20 % al conjunto de validación, un 20 % al conjunto de test y el 60 % restante al conjunto de entrenamiento. El número de ejecuciones para este caso es de 50.

El resultado obtenido en cuanto a precisión de entrenamiento es 79.69 %, en validación, 80.62 % y en test 79.15 %. Las desviaciones típicas son de 1.51, 3.99 y 4.43 para entrenamiento, validación y test respectivamente.

Estos resultados son peores que con el modelo de CNN pero puede deberse a que la arquitectura no es la más adecuada. Aun así no hacemos pruebas con otras en este caso dado que el tiempo de ejecución es muy elevado. (no nos funcionaba usar la gráfica para el procesado)

### 4.6.4. Conclusiones de la aproximación

CNN presenta mejores resultados que el perceptrón multicapa pero esto creemos que se debe a la configuración de este último, en cuanto a la comparación con sistemas no profundos como los vistos anteriormente, está claro que los profundos ofrecen mucho mejores resultados pero también tienen mayor complejidad y un coste computacional mayor.

# 5. Conclusiones Finales

En líneas generales los resultados del trabajo son favorables, y creemos que para problemas como el que nos ocupa podría suplir con creces su resolución. De entre todos los sistemas analizados nos decantamos por destacar el método de Deep Learning (CNN) dada su alta precisión.

De la misma forma cabe mencionar que kNN ofrece resultados óptimos requiriendo poco tiempo de cómputo. Debido a la propia naturaleza de los sistemas RRNNAA y SVM conllevan más tiempo de cómputo sin embargo ofrecen buenos resultados, particularmente SVM.

Durante la realización del trabajo afrontamos varias dificultades como ciertos problemas de coordinación o el hecho de buscar modelos que se adapten a nuestro problema; mas de entre todos ellos el que denotamos como más crucial fue la familiarización con el lenguaje de programación.

# 6. Trabajo Futuro

Considerando el trabajo actual, el trabajo futuro puede enfocarse en mejorar los resultados del sistema, añadir más casos de uso con imágenes más borrosas, imágenes donde la matrícula no pueda verse correctamente, o donde esté inclinada o tomada la foto desde otros ángulos. También podrían procesarse imágenes de otros tipos de vehículos, o intentar que el sistema localice en tiempo real para usarlo en sistemas de control de aparcamiento, autovías para la monitorización de vehículos, etc.

Otra opción sería quedarnos con los datos funcionales actuales y tratar de identificar los elementos de la matrícula, números y letras.

En cualquier caso este problema dista mucho de estar completado.

# Referencias


[1] Li S, Chen Y. License Plate Recognition [Generic]; 2011. Available from: http://www.diva-portal.org/smash/get/diva2:422755/FULLTEXT02.pdf.

[2] Automatic number-plate recognition - Wikipedia. 2020 [Generic]; 2020. Available from: https://en.wikipedia.org/wiki/Automatic_number-plate_recognition.

[3] Ribarić S, Kalafatić Z, Hrkać T, Srebrić V, Sokol D, Zrno D, et al.. License Plate Detection, Recognition and Automated Storage [Generic]; 2020. Available from: http://www.zemris.fer.hr/projects/LicensePlates.

[4] Akoum A, Daya B, Chauvet P. Two Neural Networks for License Number Plates Recognition [Journal Article]. 2010;12.

[5] Vajpayee S. AI-based Indian license plate detector [Journal Article]. Medium. 2020;p. 1. Available from: https://towardsdatascience.com/ai-based-indian-license-plate-detector-de9d48ca8951.

[6] Upadhyay U, Mehfuz F, Mediratta A, Aijaz A. Analysis and Architecture for the deployment of Dynamic License Plate Recognition Using YOLO Darknet. In: 2019 International Conference on Power Electronics, Control and Automation (ICPECA); 2019. p. 1–6.