

# **Memoria - AA**

## **Práctica 1 - Detección de Carreteras**



***Elaborado por:***

*Iago Pérez Díaz*

*Iker Calvo Gómez*

*Ángel Álvarez Rey*

*Carlos Álvarez Nieto*

# Índice

<b>1 - Introducción</b>	<b>3</b>
<b>2 - Descripción del problema</b>	<b>4</b>
● Descripción del problema:	4
● Restricciones:	4
● Descripción de la base de datos:	4
● Propiedades de los datos:	6
<b>3 - Análisis bibliográfico</b>	<b>7</b>
● Trabajos recientes e importantes en el ámbito tratado	7
● Referencias bibliográficas a trabajos donde se resuelve el mismo tipo de problema	8
<b>4 - Desarrollo</b>	<b>9</b>
<b>5 - Conclusiones</b>	<b>9</b>
<b>6 - Trabajo futuro</b>	<b>10</b>
<b>7 - Bibliografía</b>	<b>10</b>

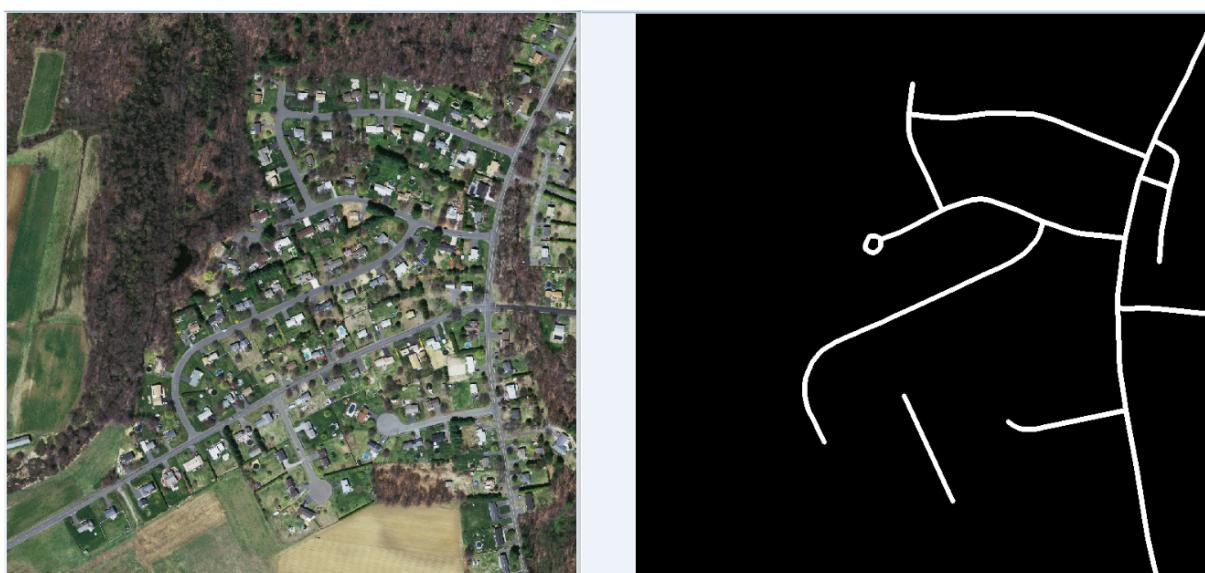
# 1 - Introducción

El uso de imágenes por satélite se ha vuelto cada vez más común en diferentes campos, desde la cartografía hasta la agricultura, la gestión de recursos naturales y la planificación urbana. Estas imágenes proporcionan información detallada y actualizada sobre la superficie terrestre, lo que ha llevado a un aumento en la demanda de métodos automáticos y precisos para analizarlas.

Las carreteras son infraestructuras fundamentales para el transporte de personas y mercancías, lo que las convierte en una parte importante de cualquier planificación urbana o rural. La capacidad de detectar y mapear carreteras a partir de imágenes por satélite podría tener una amplia variedad de aplicaciones, desde la planificación de rutas de transporte y la monitorización de la infraestructura de transporte hasta la evaluación de la accesibilidad a los servicios. Es por eso que hemos decidido orientar nuestro trabajo hacia la detección de carreteras.

El objetivo de este trabajo es desarrollar un método automático y preciso para detectar carreteras a partir de imágenes por satélite. Al conseguirlo, se podrían, por ejemplo, trazar las rutas más óptimas para diferentes propósitos, lo que a su vez puede mejorar la eficiencia en el transporte y la reducción de costos.

La salida que daría nuestro sistema sería la imagen de entrada con los píxeles detectados como carretera de color blanco y todo lo demás de color negro, como podemos ver en la *figura 1.1*:



**(fig 1.1) Imagen de ejemplo de entrada/salida de nuestro sistema**

## **2 - Descripción del problema**

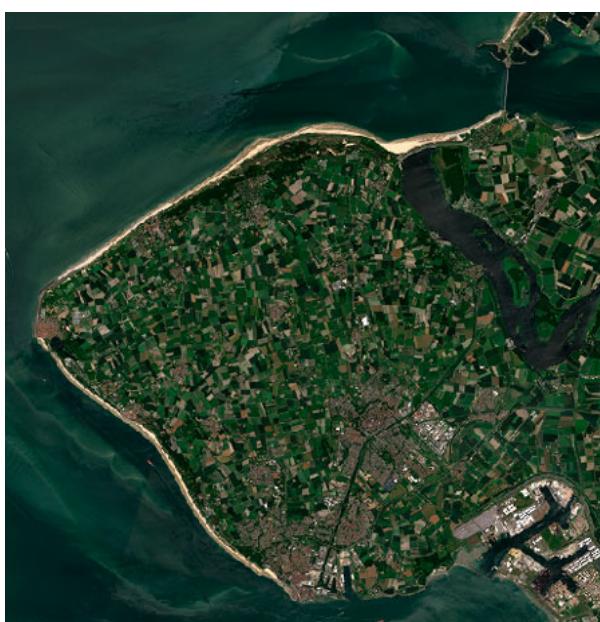
- **Descripción del problema:**

El problema de la detección de carreteras en imágenes vía satélite consiste en clasificar correctamente los píxeles que pertenecen a la carretera y aquellos que no lo hacen. Esta segmentación se realiza en base al color de los píxeles y su diferencia con los píxeles vecinos. El objetivo es obtener una imagen segmentada donde los píxeles de la carretera estén resaltados de color blanco y los demás píxeles estén en negro.

- **Restricciones:**

Para poder realizar una correcta segmentación, las imágenes de entrada deben cumplir ciertas restricciones. Estas incluyen que las imágenes sean a color, tomadas durante el día y con buena iluminación, siempre con cielo despejado, que sean perpendiculares al suelo y sacadas desde la misma altura.

En las siguientes figuras, que no pertenecen a nuestra base de datos (ya que en nuestra BBDD todas las imágenes fueron tomadas con buenas condiciones), podemos ver dos imágenes de la misma zona, una que satisface todas nuestras restricciones (*figura 2.1*) y otra que no satisface la del cielo despejado (*figura 2.2*). Por tanto la *figura 2.2* sería inapropiada para nuestro tratamiento.



**(fig 2.1) Buenas condiciones**

**(fig 2.2) Malas condiciones**

- **Descripción de la base de datos:**

La base de datos utilizada en este trabajo ha sido creada por la Universidad de Toronto y contiene imágenes satelitales de las carreteras de Massachusetts. Las imágenes tienen un tamaño de 1500x1500 píxeles, pero se han dividido en cuatro partes iguales de 750x750 píxeles para facilitar el entrenamiento y la extracción de patrones. Se han seleccionado 200 imágenes para la base de datos, de las cuales 5 se utilizarán para el test final y las 195 restantes se emplearán en el proceso de entrenamiento.

Es importante mencionar que la base de datos en sí, ha sido obtenida de la página web de la Universidad de Toronto, por lo que dejamos la referencia bibliográfica [aquí](#).

Por último, en las siguientes figuras podemos ver el resultado de la división de una de las imágenes de la base de datos en 4 subimágenes del mismo tamaño. Esta división la hacemos porque las imágenes obtenidas de la BBDD son de gran tamaño y resolución, lo que implicaría más tiempo para tratar la misma cantidad de imágenes y más recursos necesarios para almacenarlas (ya que estas están en formato TIFF).



(fig 2.3) Imagen en resolución completa (1500x1500)

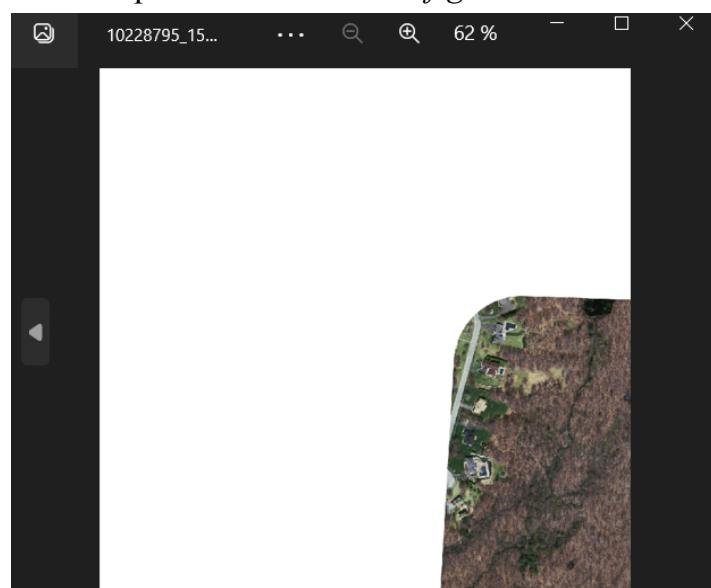




**(fig 2.4) Imágenes fragmentadas en 750x750**

- **Propiedades de los datos:**

Las imágenes utilizadas en este trabajo tienen un tamaño de 750x750 píxeles y toman valores entre 0 y 255. Pueden contener tanto carreteras como otros elementos de la imagen, como por ejemplo árboles o edificios, pero nunca contendrán cuerpos de agua. Asimismo, algunas imágenes pueden contener zonas vacías, que representan áreas sin información, es decir, sin carreteras ni ningún otro elemento de la imagen satelital. En este caso, se incluirá un valor de negro en la imagen segmentada para llenar estas zonas vacías. Por ejemplo, la imagen puede estar vacía en su totalidad o solo en algunas partes de la imagen, como se puede observar en la *figura 2.5*.



**(fig 2.5) Imagen con un fragmento sin información de mapa**

## **3 - Análisis bibliográfico**

En cuanto a los trabajos recientes e importantes en el ámbito tratado, se han identificado diversas técnicas y métodos de clasificación y detección de terreno mediante el uso de técnicas de aprendizaje automático. En particular, se han encontrado varios trabajos que utilizan el modelo Bayesiano, como la tesis de **V. Mnih "Machine Learning for Aerial Image Labeling" [2013]**, y el trabajo de **O. Burton y T. Lockers "A Multi-spectral Image Classification Approach for Terrain Identification" [2018]**. Ambos trabajos utilizan una aproximación basada en las reglas de Bayes para obtener las probabilidades de las clases de salida. El primer trabajo mencionado se centra en la clasificación de terreno a partir de imágenes aéreas, mientras que el segundo se enfoca en la identificación de la composición de un terreno a partir de una imagen multiespectral.

Por otro lado, también se han encontrado trabajos que utilizan redes neuronales convolucionales, como el trabajo de **Y. Wei, K. Zhang y S. Ji "A CNN-based approach for SAR image ship detection using path tracing and edge detection" [2019]**. En este trabajo, los autores proponen la utilización de múltiples semillas de inicialización para beneficiar tanto la segmentación como el trazado de rutas en la detección de barcos en imágenes SAR.

Además, se ha encontrado un trabajo que utiliza redes neuronales artificiales para la clasificación de terreno, el trabajo de **S. Yu "Artificial neural networks for land cover classification" [2002]**. En este trabajo, se propone una técnica de clasificación de terreno utilizando una red neuronal artificial, capaz de clasificar los datos de la imagen sin tener que definir una regla de clasificación.

En cuanto a los trabajos donde se resuelve el mismo tipo de problema, se han encontrado varios trabajos que se centran en la detección y extracción de carreteras a partir de imágenes satelitales. Algunos de los trabajos más relevantes son **"Road Extraction by Deep Residual U-Net" de H. Wang et al. [2021]**, **"Road Detection using Multi-Scale Feature Learning and DeepLabv3+ Network" de M. Zhang et al. [2021]**, **"Multi-Scale and Multi-Task Learning for Road Detection in Satellite Images" de T. Lin et al. [2020]**, y **"Road Network Extraction and Intersection Detection from Aerial Imagery using Deep Convolutional Neural Networks" de S. Milz et al. [2018]**.

Cada uno de estos trabajos propone un enfoque diferente para la detección de carreteras, utilizando técnicas de aprendizaje profundo como redes neuronales convolucionales, y combinando múltiples tareas para mejorar la precisión de la

detección de carreteras en imágenes satelitales. En particular, se destaca el trabajo de **S. Milz et al. [2018]**, ya que propone una técnica de agrupación jerárquica para unir las secciones de carreteras detectadas, lo que podría mejorar la eficacia de la detección de redes de carreteras en imágenes satelitales.

## **4 - Desarrollo**

### **● Descripción:**

En cuanto a la base de datos, como se ha dicho anteriormente, se compone de un conjunto de imágenes, de las cuales se extraen ciertas características. Se ha optado por tomar “ventanas” de cada una de ellas, que son divisiones o recortes de las mismas.

Usamos estas ventanas en la obtención de características de las imágenes, se calculan seis características de cada ventana, que son la media y la desviación típica de cada canal RGB. Por ende obtenemos 6 inputs para cada target. Estos valores proporcionan información valiosa acerca de la distribución de colores en la ventana, que ayudan a distinguir los tramos que son carretera y los que no. Como primera aproximación hemos decidido normalizar los datos a través del cálculo de la media y desviación típica de los canales RGB. Creemos que es la mejor para este caso debido a que permite estandarizar los valores de intensidad de los colores de cada imagen de modo. Además, es una técnica fácil de implementar, y ayuda a reducir la influencia de valores externos en la imagen.

Estas ventanas en los ground truth se asocian con los valores positivo y negativo, siendo una imagen “positiva” cuando su píxel central se corresponde con carretera, y “negativa” en caso contrario. Este píxel central será el target de los input correspondientes para esa ventana.

Estos valores input y target están forzados a tener una relación ( $\text{carretera} * 1.2 > \text{noCarretera}$ ), sino habría demasiados valores no carretera que sobreentrenarían a nuestra red con un 96-99% de valores no carretera, dando como resultado que la red aprenda a responder que todos los píxeles centrales son no carretera y obteniendo buenos resultados por ello.

A la hora de entrenar los algoritmos los inputs se dividen en un 85% entrenamiento y un 15% para test. En el algoritmo RRNNA de ese 85% de entrenamiento se extrae otro 15% para validación quedando: 72.25% entrenamiento, 15% test y 12.75% validación.

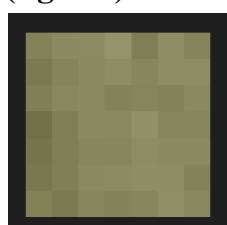
## ● 1<sup>a</sup>-Aproximación

A la hora de definir el tamaño de la ventana y el salto entre ventanas elegiremos un tamaño pequeño y un salto también pequeño( Usaremos un salto de ventana de 3 y tamaño de ventana de 7). Con el tamaño pequeño de ventanas conseguimos que solo tomen peso en el píxel carretera los valores muy cercanos al centro y evitar que se diluya la información RGB de media y desviación típica de los píxeles cercanos. Con un salto pequeño de ventana conseguimos poder obtener un mayor número de píxeles carretera, poniendo un salto más grande correríamos el riesgo de pasar de largo demasiadas carreteras y no conseguir una muestra representativa de estas, podría subirse el salto de ventana si lo que se desea es bajar la muestra de carreteras y no carreteras por un sobreentrenamiento.

### -Ejemplo ventana 7x7:



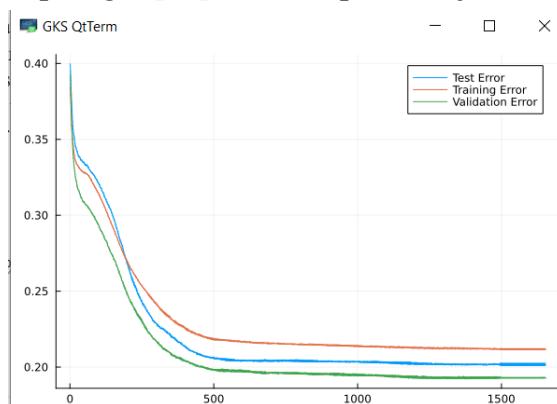
(fig. 4.1) Pixel central Carretera



(fig. 4.2) Pixel central No carretera

En el **Algoritmo RRNNAA** reamos redes neuronales con topología [12] [12 12] y [12 6] y a su vez usamos un bitrate de aprendizaje entre 0.001(bajo) y 0.1(alto). Con un aprendizaje bajo, al no generar grandes saltos en el aprendizaje, se impide saltar los valores de salida óptimos con facilidad. Como error mínimo definimos 0.20 para que la red no decida por poner todos los valores como positivos o negativos y conseguir buen rendimiento por ello y añadimos 200 iteraciones por la complejidad a la hora de clasificar y poder encontrar una solución aceptable. A la hora de nombrar las topologías obviamos las capas de entrada,6 , y de salida, 1, mencionado sólo las capas ocultas.

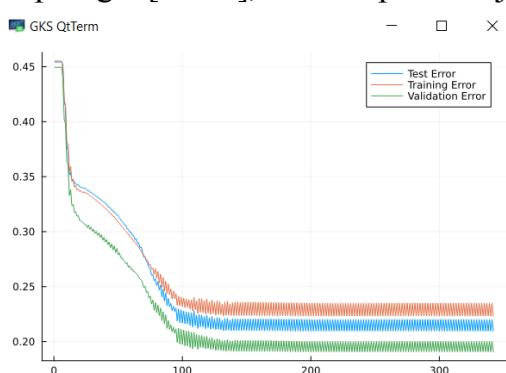
### 1. Topología [12], bitrate aprendizaje 0.0025, error mínimo 20%



(fig 4.3)

Error Mínimo: **0.2011783514921838**

### 2. Topología [12 12], bitrate aprendizaje 0.01, error mínimo 20%

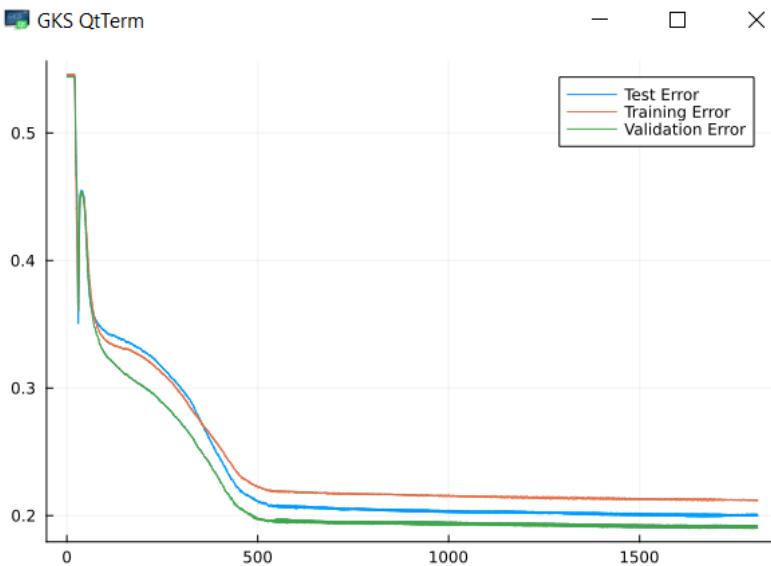


(fig 4.4)

Error Mínimo: **0.2094090478446234**

Puede observarse que al elegir un aprendizaje mayor se dan muchos más saltos durante el entrenamiento impidiendo una solución óptima aun añadiendo más capas con más neuronas.

### 3. Topología [12 12], bitrate aprendizaje 0.0015, error mínimo 20%



**(fig 4.5)**

Esta última topología será la elegida para la RRNNAA por ende será la que mostremos su tabla de resultados extendida:

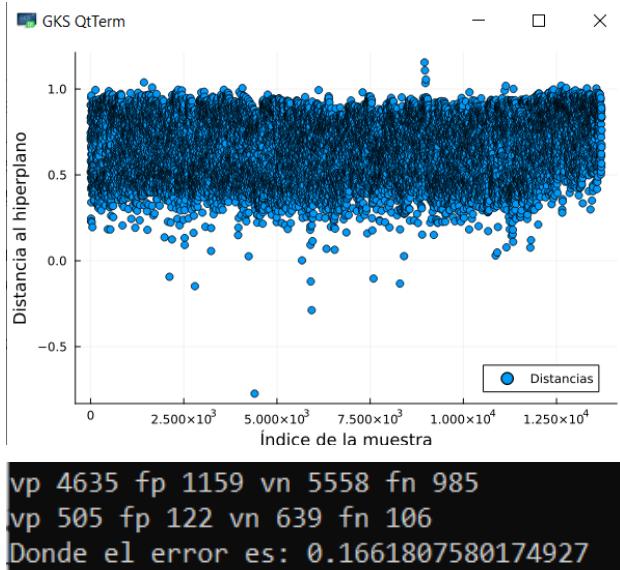
```
Entrenamiento- Error Minimo: 0.2124353467970392 | Sensibilidad: 0.7875646532029608
Test- Error Minimo: 0.19931312174324964 | Sensibilidad: 0.7875646532029608
Validacion- Error Minimo: 0.1901769853279821 | Sensibilidad: 0.7875646532029608
```

Para el **Algoritmo SVM** decidimos usar una única iteración de entrenamiento debido a que cada vez que se entrena el modelo, se ajustan los parámetros de la función de decisión para maximizar el margen entre las clases y minimizar el error de clasificación. Si se entrena el modelo de manera iterativa, los parámetros del modelo pueden cambiar en cada iteración y esto podría dar lugar a resultados impredecibles e inestables.

Usaremos la función SVC() con distintas gamma para ajustar nuestro espacio de búsqueda de una solución óptima y conseguir una clara diferenciación entre clases. También disminuimos el número de datos de entrenamiento para encontrar una solución óptima en un tiempo razonable. Cambiamos el salto de ventana de 3 a 15 e iremos bajando a medida que ajustamos el parámetro gamma.

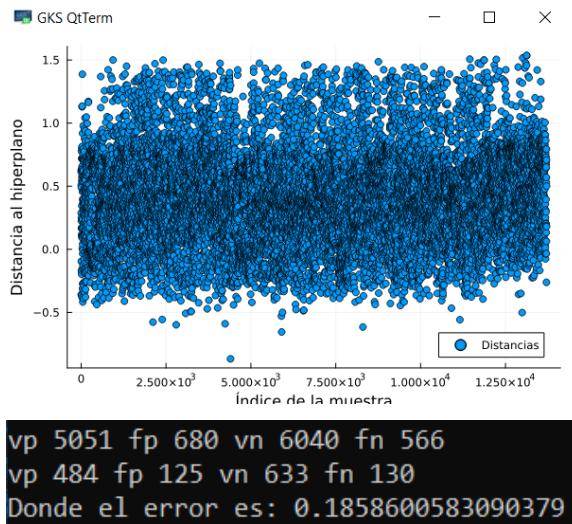
Una distancia del hiperplano positiva indica la pertenencia a una clase mientras que una negativa indica la no pertenencia, en nuestro caso carretera/no carretera.

Gamma = 1 y Coste = 1. Una línea grande de puntos con valores positivos. Dificultad a la hora de diferenciar los píxeles no carretera (valores negativos).



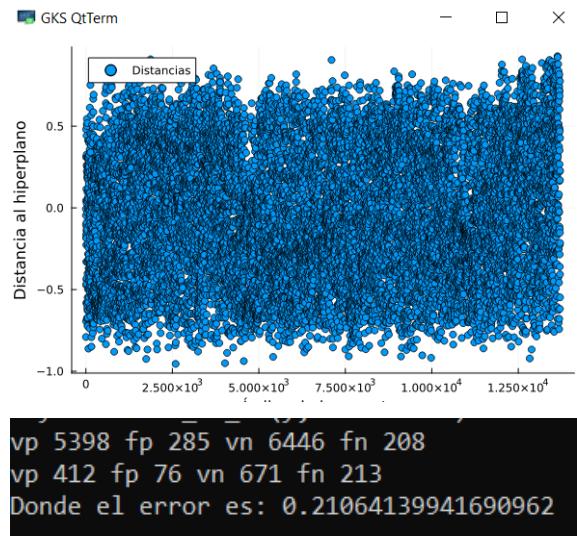
(fig 4.6)

Gamma = 10 y Coste = 1. Una masa de puntos repartidos entre valores positivos y negativos como medio el 0, dificultad a la hora de diferenciar las clases.



(fig 4.7)

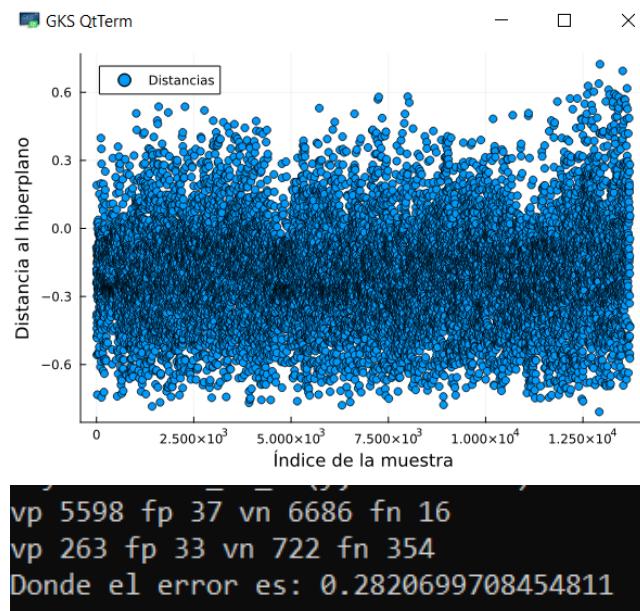
Gamma = 25 y Coste = 1. Mejor clasificación de píxeles carretera pero mayor error general.



(fig 4.8)

Con gamma alto conseguimos diferenciar más las clases en el entrenamiento pero puede llegar a clasificar mal la muestra al sobre ajustar sus parámetros y dará un mayor error en los test.

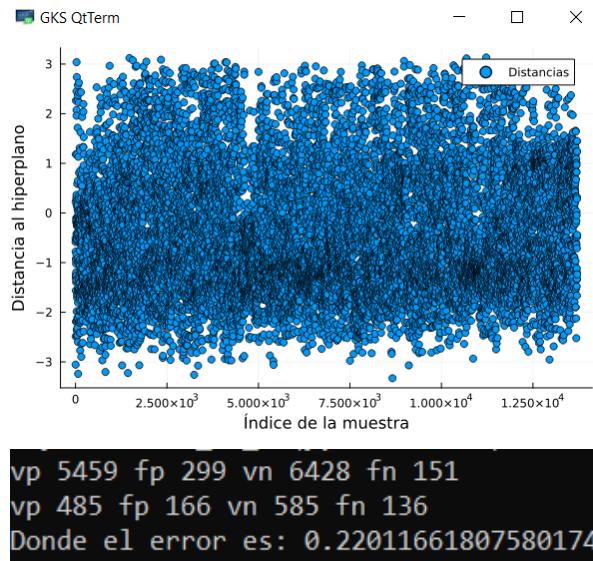
Gamma = 70 y Coste = 1. El modelo detecta no carreteras más fácilmente, por ello hay un mayor número de falsos negativos.



(fig 4.9)

Aumentando el Coste se le atribuye mayor peso al error:

Gamma = 8, Coste = 80.



(fig 4.10)

La tabla comparativa con las matrices de confusión de las anteriores iteraciones queda de la siguiente forma:

	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>Gamma = 1 Coste = 1</b>	505	122	639	106
<b>Gamma = 10 Coste = 1</b>	484	125	633	130
<b>Gamma = 25 Coste = 1</b>	412	76	671	213
<b>Gamma = 70 Coste = 1</b>	263	33	722	354
<b>Gamma = 8 Coste = 80</b>	485	166	585	136

El modelo SVM se ajusta mejor cuantos menos datos de entrada debido a la mayor facilidad a la hora de distinguir clases y menor cantidad de puntos como solución del hiperplano. Puede apreciarse que a mayor gamma, aunque en la matriz de confusión se vea un mejor desempeño en los entrenamientos, se sobre ajustan los parámetros dando peores resultados en los test.

En cuanto al **Algoritmo Decision Tree** ajustamos la profundidad máxima. Este es un hiper parámetro que determina la profundidad máxima del árbol de decisión durante el entrenamiento. Esto significa que el modelo no puede crear ramas más profundas que esta profundidad máxima.

Especificar una profundidad máxima puede ayudar a evitar el sobreajuste del modelo a los datos de entrenamiento, lo que significa que el modelo es demasiado complejo y se ajusta demasiado a los datos de entrenamiento, perdiendo su capacidad de generalización a datos nuevos. Por otro lado, si se establece una profundidad máxima demasiado baja, el modelo puede ser demasiado simple y no capturar las relaciones complejas en los datos.

Con profundidad 10:

```
vp 109828 fp 33855 vn 131940 fn 28350
vp 11918 fp 3572 vn 14860 fn 3426
MatrizConfusion Test[14860 3572; 3426 11918]
Donde el error es: 0.20718853623874942
```

Por último pasamos al **Algoritmo KNN**, el cual busca los k ejemplos más cercanos a un nuevo ejemplo de entrada en el conjunto de entrenamiento y asigna al nuevo ejemplo la clase más común entre esos k vecinos cercanos.

Un número más alto de vecinos puede dar como resultado una precisión más alta, pero puede hacer que el modelo sea más lento. Por otro lado, un número más bajo de vecinos puede dar como resultado un modelo más rápido, pero también puede reducir la precisión.

Vecinos = 15

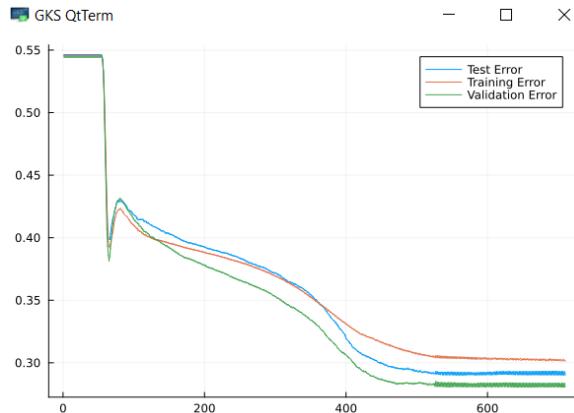
```
vp 11722 fp 3512 vn 14870 fn 3672
Donde el error minimo es es: 0.21269540502131692
```

## ● 2<sup>a</sup>-Aproximación

Como segunda aproximación decidimos utilizar la misma metodología pero cambiando el tamaño de la ventana, comprobando así su influencia en los resultados, ampliando o disminuyendo la información disponible. Probaremos la tendencia aumentando el tamaño a 15x15, luego 25x25 y después disminuyendo a 5x5.

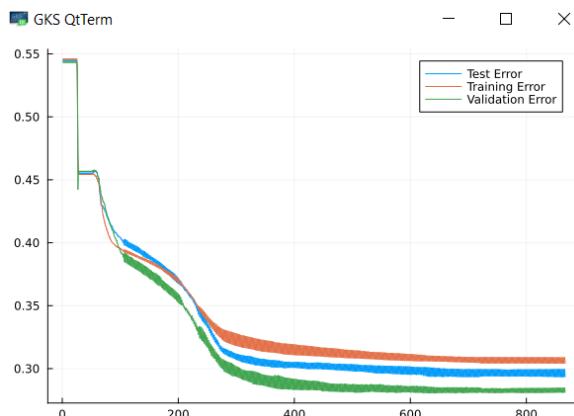
### ● 15x15:

En cuanto al **Algoritmo RRNNAA**, para la red elegida de la aproximación uno, topología [12 12] y aprendizaje 0.0015:



Test- Error Minimo: 0.2902276412997774 | Sensibilidad: 0.6964398270709588

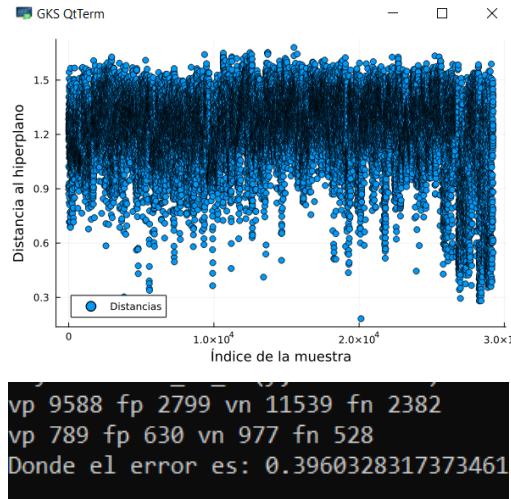
Otra topología junto a otra tasa de aprendizaje, [20 20 12] y 0.002:



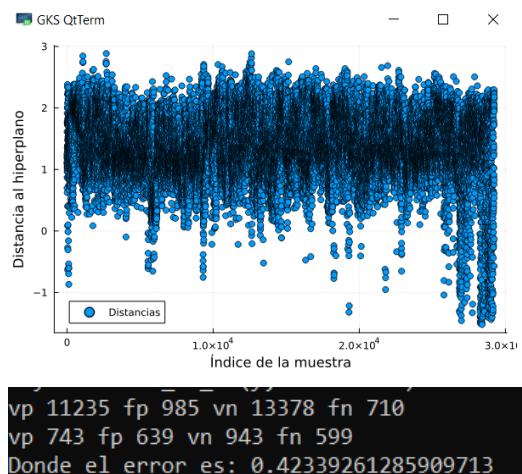
Test- Error Minimo: 0.2935011111778485 | Sensibilidad: 0.6950197622596457

Con **SVM** al igual que en la aproximación anterior, en este algoritmo usaremos un salto de ventana de 10 para obtener un tiempo razonable de entrenamiento.

Gamma = 5 y Coste = 10.



Gamma = 10 y Coste = 20.



Gamma=20 y Coste=40

vp 12341 fp 12 vn 14829 fn 2

vp 828 fp 415 vn 1220 fn 559

Donde el error es: 0.32230311052283256

La tabla comparativa con las matrices de confusión de las anteriores iteraciones queda de la siguiente forma:

	VP	FP	VN	FN
<b>Gamma = 5 Coste = 10</b>	789	630	977	528

<b>Gamma = 10</b> <b>Coste = 20</b>	743	639	943	599
<b>Gamma = 20</b> <b>Coste = 40</b>	828	415	1220	559

En cuanto a **Decision Tree** obtuvimos:

Con profundidad 8:

```
vp 95896 fp 49998 vn 113480 fn 40301
vp 10443 fp 5023 vn 13121 fn 4711
MatrizConfusion Test[13121 5023; 4711 10443]
Donde el error es: 0.29232986966184155
```

Con profundidad 10:

```
vp 100439 fp 49635 vn 113814 fn 35787
vp 10668 fp 5230 vn 12943 fn 4457
MatrizConfusion Test[12943 5230; 4457 10668]
Donde el error es: 0.29091837347588445
```

Con profundidad 15:

```
vp 108117 fp 38046 vn 125324 fn 28188
vp 10236 fp 5214 vn 13038 fn 4810
MatrizConfusion Test[13038 5214; 4810 10236]
Donde el error es: 0.3010391014475344
```

Finalmente con **KNN** los resultados fueron:

Vecinos = 22

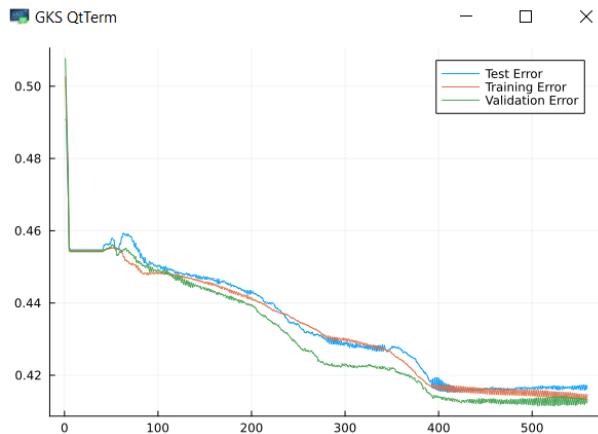
```
vp 10015 fp 4440 vn 13699 fn 5144
Donde el error minimo es es: 0.2878250946002763
```

Vecinos= 24

```
vp 10101 fp 4386 vn 13671 fn 5140
Donde el error minimo es es: 0.28608324824313774
```

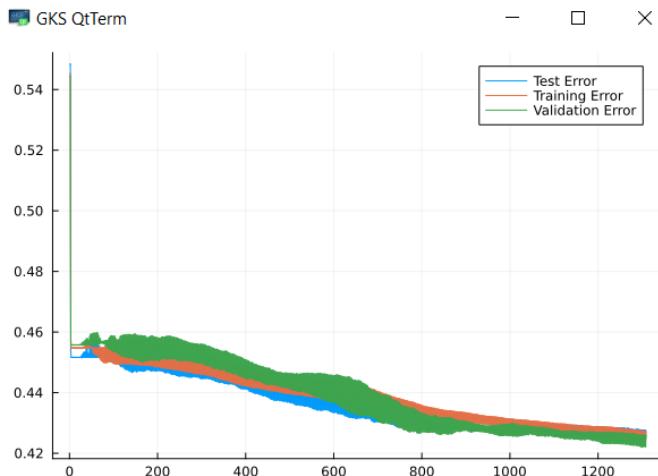
- **25x25:**

En cuanto a **RRNNAA**, para la red elegida de la aproximación uno, topología [12 12] y aprendizaje 0.0015:



Test- Error Minimo: 0.41534268636377714 | Sensibilidad: 0.5828611193955662

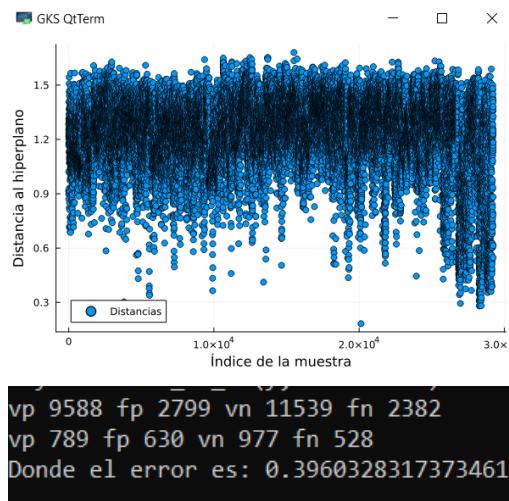
Otra topología junto a otra tasa de aprendizaje, [24 24 24] y 0.003:



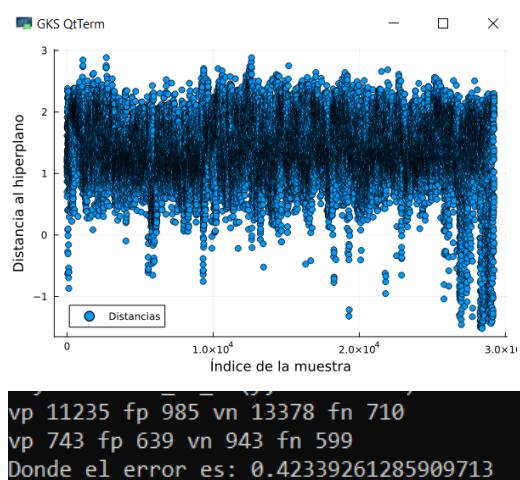
Test- Error Minimo: 0.42478862770602993 | Sensibilidad: 0.5709353277202964

Con **SVM**, al igual que en la aproximación anterior, en este algoritmo usaremos un salto de ventana de 10 para obtener un tiempo razonable de entrenamiento.

Gamma = 5 y Coste = 10.



Gamma = 10 y Coste = 20.



La tabla comparativa con las matrices de confusión de las anteriores iteraciones queda de la siguiente forma:

	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>Gamma = 5 Coste = 10</b>	789	630	977	528
<b>Gamma = 10 Coste = 20</b>	743	639	943	599

En cuanto a **Decision Tree** obtuvimos:

Con profundidad 15:

```
vp 86846 fp 50324 vn 108148 fn 45276
vp 8547 fp 7414 vn 10232 fn 6096
MatrizConfusion Test[10232 7414; 6096 8547]
Donde el error es: 0.41840874601257394
```

Con profundidad 17:

```
vp 89929 fp 45489 vn 113007 fn 42169
vp 8529 fp 7322 vn 10300 fn 6138
MatrizConfusion Test[10300 7322; 6138 8529]
Donde el error es: 0.4168602310384341
```

Con profundidad 20:

```
vp 101264 fp 37355 vn 121165 fn 30810
vp 8562 fp 7428 vn 10170 fn 6129
MatrizConfusion Test[10170 7428; 6129 8562]
Donde el error es: 0.41986435008826534
```

Finalmente con **KNN** los resultados fueron:

Vecinos = 22

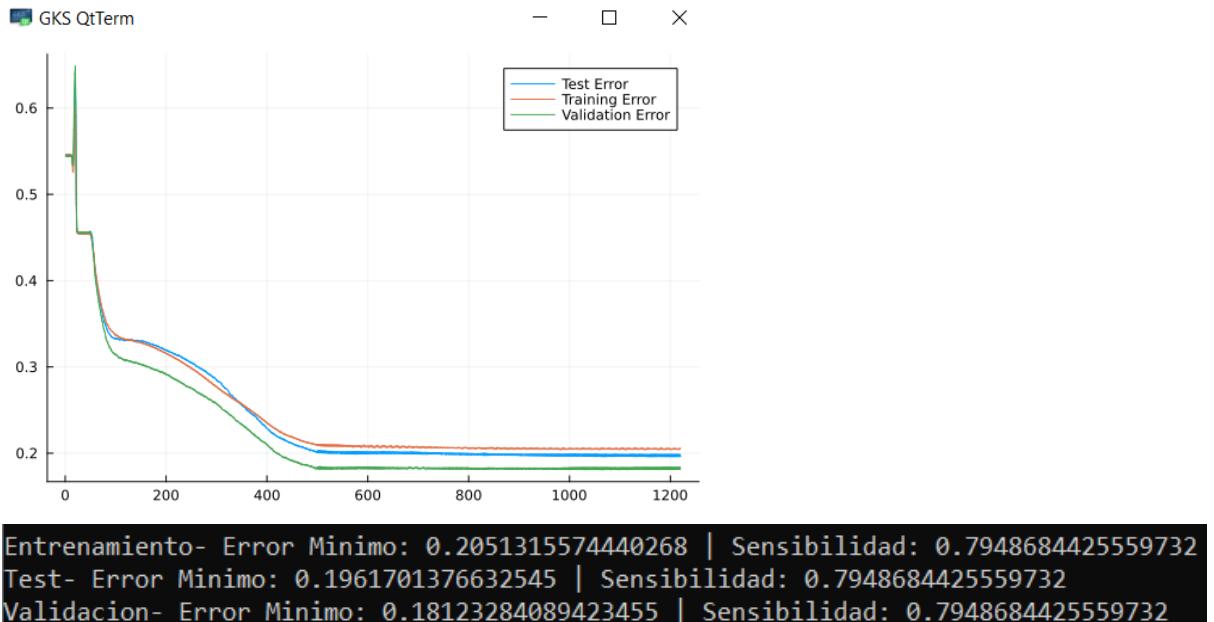
```
vp 7659 fp 5445 vn 12107 fn 7078
Donde el error minimo es es: 0.3878410604230543
```

Vecinos= 24

```
vp 7587 fp 5591 vn 12028 fn 7083
Donde el error minimo es es: 0.3925175756449565
```

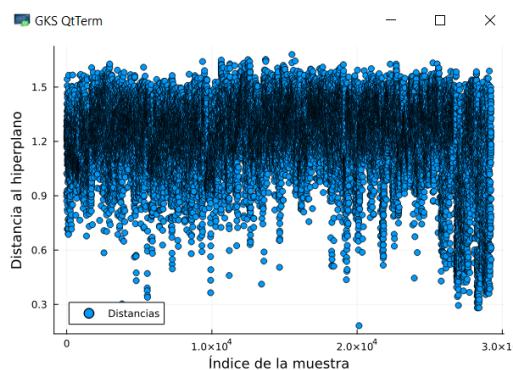
- **5x5:**

En cuanto a **RRNNAA**, para la red elegida de la aproximación uno, topología [12 12] y aprendizaje 0.0015:

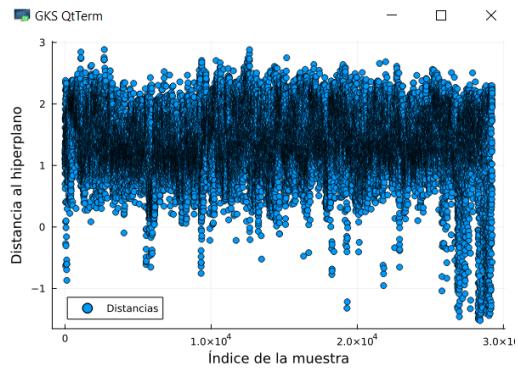


Con **SVM**, al igual que en la aproximación anterior, en este algoritmo usaremos un salto de ventana de 10 para obtener un tiempo razonable de entrenamiento.

Gamma = 5 y Coste = 10.



Gamma = 10 y Coste = 20.



La tabla comparativa con las matrices de confusión de las anteriores iteraciones queda de la siguiente forma:

	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>Gamma = 5 Coste = 10</b>				
<b>Gamma = 10 Coste = 20</b>				

En cuanto a **Decision Tree** obtuvimos:

Con profundidad 7:

```
vp 109876 fp 38021 vn 128936 fn 29118
vp 12050 fp 3995 vn 14474 fn 3477
MatrizConfusion Test[14474 3995; 3477 12050]
Donde el error es: 0.21979056359571714
```

Con profundidad 10:

```
vp 110763 fp 31303 vn 135519 fn 28366
vp 11912 fp 3401 vn 15203 fn 3480
MatrizConfusion Test[15203 3401; 3480 11912]
Donde el error es: 0.2024061654312272
```

Con profundidad 13:

```
vp 114882 fp 28035 vn 138718 fn 24316
vp 11823 fp 3487 vn 15186 fn 3500
MatrizConfusion Test[15186 3487; 3500 11823]
Donde el error es: 0.20552417931521355
```

Finalmente con **KNN** los resultados fueron:

Vecinos = 17

```
vp 11760 fp 3185 vn 15352 fn 3699  
Donde el error minimo es es: 0.20249441110718908
```

Vecinos= 20

```
vp 11517 fp 2889 vn 15531 fn 4059  
Donde el error minimo es es: 0.20437698552770914
```

## • Discusión:

Observando los resultados llegamos a la conclusión de que un mayor tamaño de ventana diluye más la información del píxel central a la hora de calcular la media y desviación típica. Esto es un problema debido a que la red neuronal tendrá más problemas a la hora de clasificar los píxeles centrales como carretera o no.

Como se ve en la sección “Resultados” la tendencia de error para ventanas cada vez más grandes es mayor que para las ventanas pequeñas. Las ventanas 25x25 y 15x15 tienden al 40% y 30% de error respectivamente. Por otro lado, las ventanas de 7x7 y 5x5 tienden al 20% y 19% de error.

También hay que tener en cuenta que sería un error el usar un tamaño de ventana demasiado pequeño, ej: 1x1, ya que sólo estaríamos clasificando un único píxel perdiendo mucha información del entorno de este, si es un borde de una carretera; si está rodeado por carretera; si no...

Como conclusión de esta segunda aproximación, elegir un buen tamaño de ventana es importante para no diluir la información de está y perder precisión a la hora de clasificar, pero tampoco podemos caer en el error de hacer todo lo contrario y acabar perdiendo de igual manera la información del entorno haciéndola inexistente.

## **5 - Conclusiones**

## **6 - Trabajo futuro**

## **7 - Bibliografía**

