

# **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>

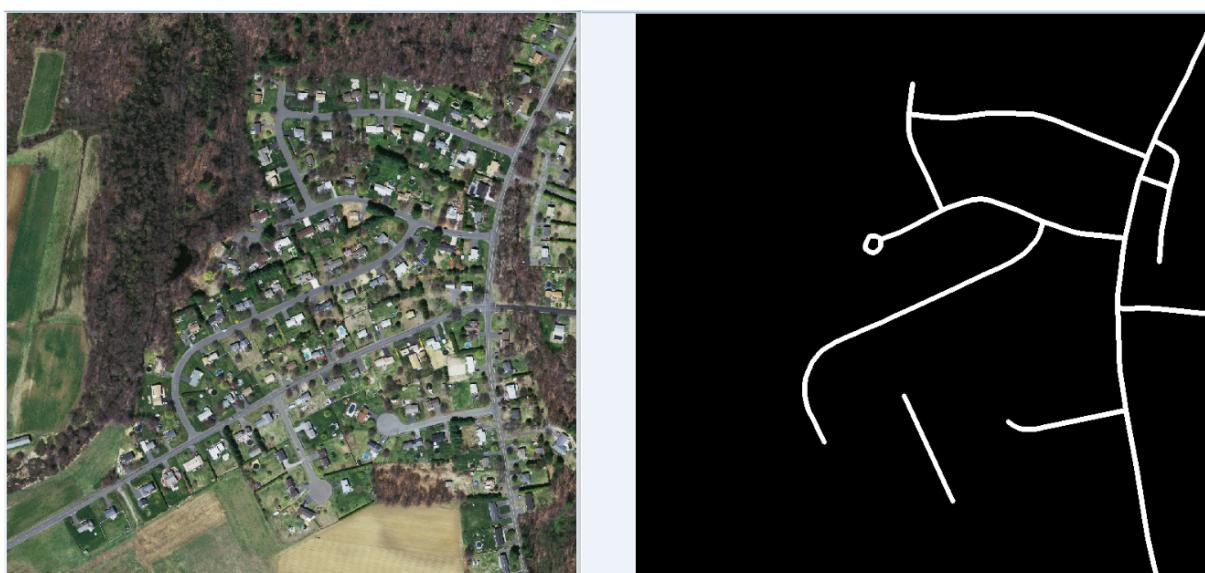
# 1 - Introducción

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**

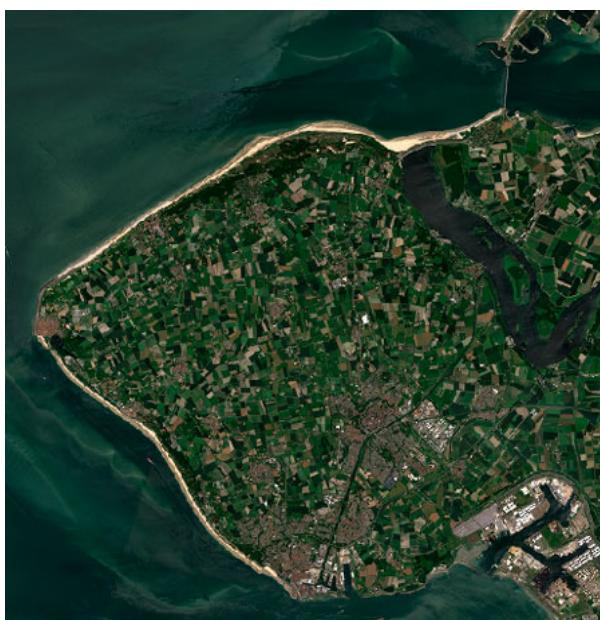
- **Descripción del problema:**

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)

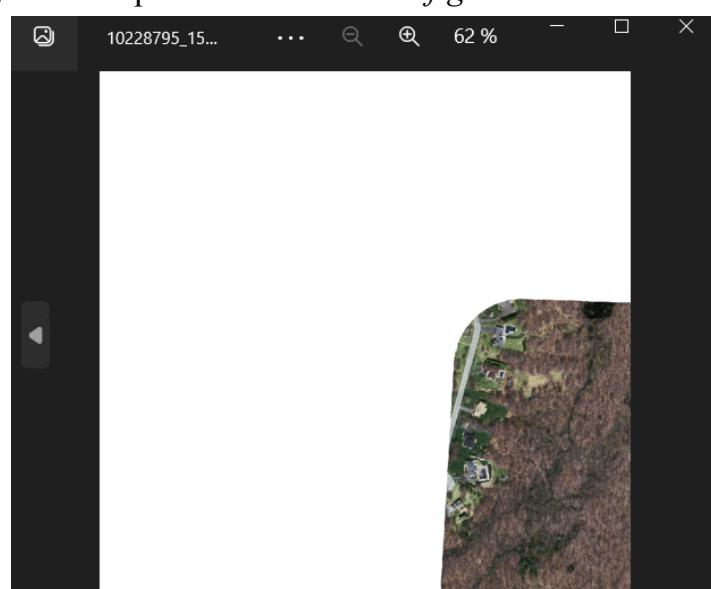




**(fig 2.4) Imágenes fragmentadas en 750x750**

- **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 casi 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]**. La técnica propuesta por este trabajo se basa en la combinación de una U-Net y una Res-Net. La U-Net se trata de una red neuronal utilizada en tareas de segmentación de imágenes, mientras que la Res-Net es una arquitectura de red neuronal que hace uso de conexiones residuales para mejorar la red. La combinación de ambas técnicas permite alcanzar una red neuronal con la capacidad de realizar la segmentación de carreteras con alta precisión y velocidad.

**"Road Detection using Multi-Scale Feature Learning and DeepLabv3+ Network"** de M. Zhang et al. [2021]. Este trabajo propone la combinación de una red neuronal llamada “DeepLabv3+” con técnicas de aprendizaje de características multi-escala. Utiliza una arquitectura de red convolucional profunda, que permite segmentar las imágenes de forma precisa y detectar las características de la carretera en diferentes escalas. Además, los autores también han hecho uso de técnicas de aumento de datos para mantener el rendimiento en diferentes escenarios y condiciones de iluminación.

**"Multi-Scale and Multi-Task Learning for Road Detection in Satellite Images"** de T. Lin et al. [2020]. Este también hace uso de un esquema de aprendizaje multi-escala y multi-tarea. Lo que lo diferencia es que, en vez de utilizar una única red neuronal para la detección de carreteras, se utilizan varias rede especializadas en diferentes escalas de la imagen y diferentes tareas de segmentación de imágenes, como la detección de objetos. Posteriormente, estas redes se combinan en una sola red, realizando una fusión de características para obtener el resultado final. Esta técnica, al igual que otras, también es capaz de generalizar bien los datos en diferentes escenarios de iluminación.

**"Road Network Extraction and Intersection Detection from Aerial Imagery using Deep Convolutional Neural Networks"** de S. Milz et al. [2018]. 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. No necesitamos normalizar los canales RGB debido a que estos ya vienen dados en un rango 0-1.

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 (carretera\*1.2 > 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.

Para entrenar a los algoritmos, aparte de dividir en test y entrenamiento, se añade validación cruzada. Los inputs se dividen en un 85% entrenamiento y un 15% para test. En el algoritmo RRNNAA de ese 85% de entrenamiento se extrae otro 15% para validación quedando: 72.25% entrenamiento, 15% test y 12.75% validación.

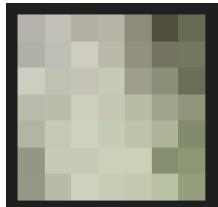
## ● 1<sup>a</sup>-Aproximación

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) consiguiendo 320.000 patrones.

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, podía subirse el salto de ventana si lo que se desea es bajar la muestra de carreteras y no carreteras por un sobreentrenamiento.

Los píxeles carretera pueden tener distintos patrones. Pueden ser carreteras que ocupen todo el kernel (tipo 1), carreteras que solo ocupen la mayor parte del kernel (tipo 2), carreteras que ocupen menos en el kernel (tipo 3) y por último que no haya carreteras en el kernel (tipo 4). Los kernel que tengan como pixel central un pixel carretera que sea el borde de una carretera serán del tipo 2, este patrón junto el tipo 1 serán los que mejor clasifique nuestra red neuronal al no haber mucha información diluida por sus vecinos en la ventana. Para los kernel con pixel central en una carretera muy pequeña o una esquina serán de tipo 3, estos serán los más difíciles de clasificar para nuestra red neuronal debido a que los vecinos generarán mucho “ruido” en los canales RGB distorsionando el valor del píxel central. El último tipo, tipo 4, generaría problemas si para nuestro kernel hay objetos con colores similares a las carreteras, ej: casas o montañas.

#### **-Ejemplo ventana 7x7:**



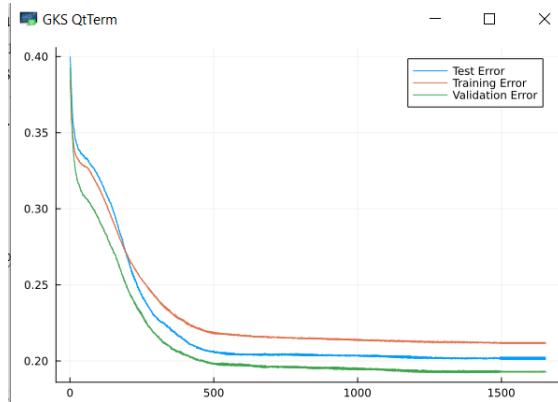
**(fig. 4.1) Pixel central Carretera**



**(fig. 4.2) Pixel central No carretera**

En el **Algoritmo RRNNAA** creamos 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. Decidimos añadir 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.

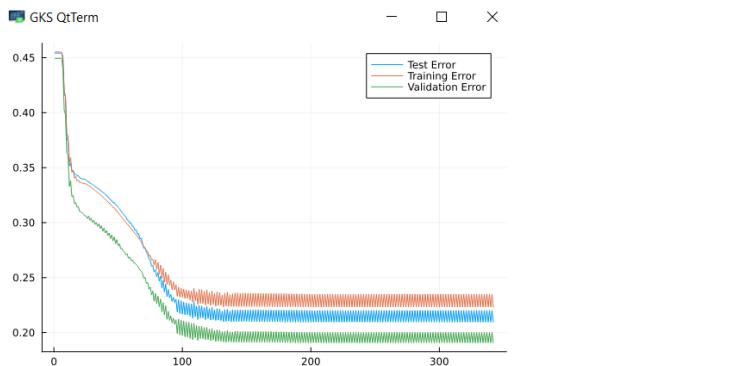
1. Topología [12], bitrate aprendizaje 0.0025



(fig 4.3)

Error Mínimo: **0.2011783514921838**

2. Topología [12 12], bitrate aprendizaje 0.01

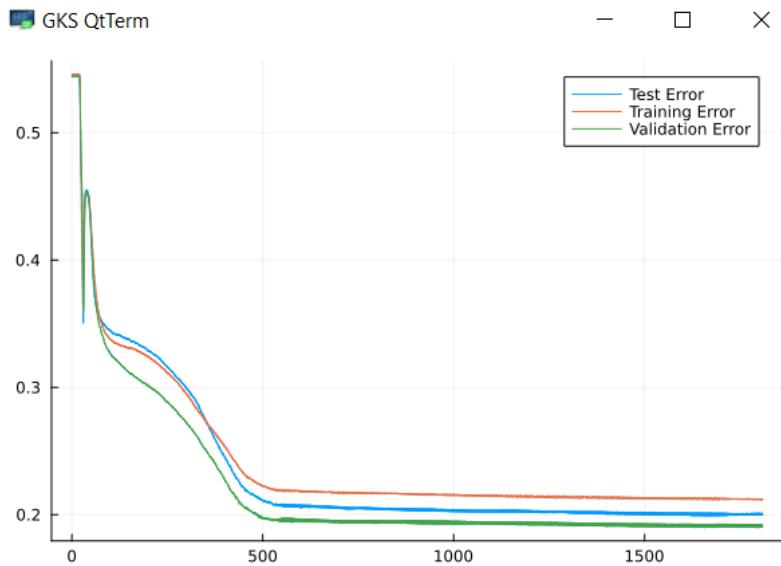


(fig 4.4)

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



**(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: 0.2167741935483871 | Sensibilidad: 0.7535461563203603
Test- Error Minimo: 0.20538252013263855 | Sensibilidad: 0.7535461563203603
Validacion- Error Minimo: 0.1927429436147115 | Sensibilidad: 0.7535461563203603
```

Topología y aprendizaje	VP	FP	VN	FN
[12] y 0.025	11486	3011	15332	3947
[12 12] y 0.015	11466	3028	15365	3917

Para el **Algoritmo SVM** decidimos usar una única llamada a la función debido a que esta ya se encarga de realizar las iteraciones de entrenamiento de manera interna. 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).

**Donde el error es: 0.1661807580174927**

**(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.

**Donde el error es: 0.1858600583090379**

**(fig 4.7)**

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

**Donde el error es: 0.21064139941690962**

**(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.

**Donde el error es: 0.2820699708454811**

**(fig 4.9)**

Aumentando el Coste se le atribuye mayor peso al error:

Gamma = 8, Coste = 80.

Donde el error es: 0.22011661807580174

(fig 4.10)

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

	VP	FP	VN	FN
<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.

Con profundidad 5:

```
Error Test: 0.25032567503552816  
Precision Test: 0.7813069513147618
```

Con profundidad 10:

```
Error Test: 0.2098531501657982  
Precision Test: 0.7737530077388307
```

Profundidad	VP	FP	VN	FN
5	12004	5095	11317	3360
10	11898	3609	14790	3479

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 = 12

```
error minimo es es: 0.21284343912837517
```

Vecinos = 17

```
Donde el error minimo es es: 0.20958668877309333
```

Vecinos	VP	FP	VN	FN
12	11201	3013	15386	4176
17	11631	3441	15066	3638

- **Discusión Aproximación 1:**

Para esta primera aproximación podemos observar la falta de precisión a la hora de clasificar los píxeles de carretera. Para un tamaño de ventana de 7x7 perdemos mucha información de los píxeles vecinos al píxel central. No sabemos si aumentar o disminuir este tamaño podría influir en el resultado por lo que será el objetivo de la siguiente aproximación.

Como mejor algoritmo seleccionaremos RRNNAA debido a presentar mejores resultados. Aunque el tiempo de entrenamiento sea mayor, es el algoritmo que mejores resultados obtiene para una gran cantidad de imágenes de entrada.

- **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. Como salto entre ventanas elegiremos un tamaño pequeño consiguiendo unos 260.000 patrones.

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 tamaño mayor conseguimos obtener mayor información de los píxeles vecinos al píxel central.

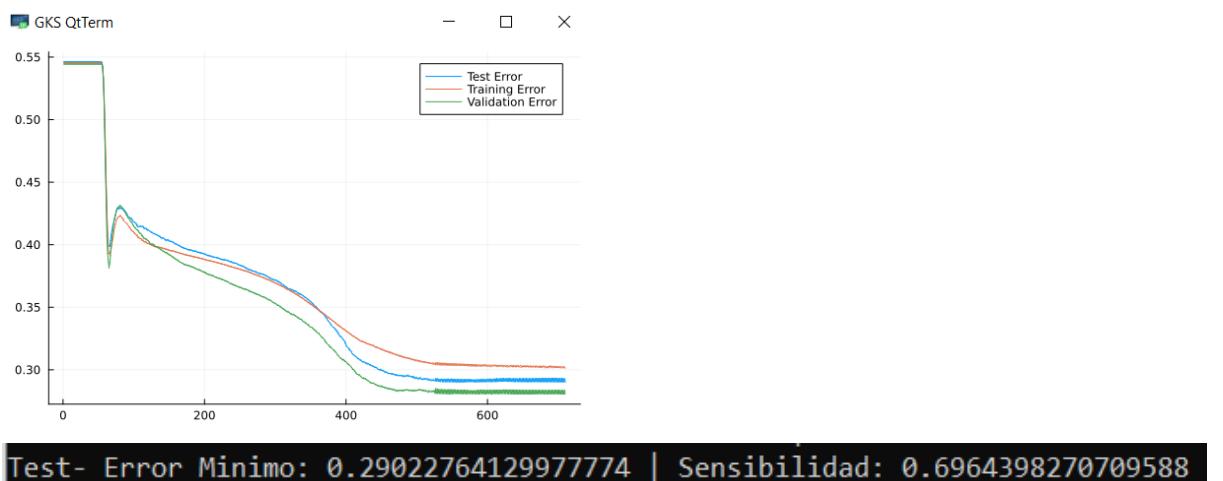
Buscamos una mejor distinción de los patrones de la primera aproximación.

Intentaremos mejorar la predicción para las ventanas tipo 3 (menor cantidad de pixel carretera), buscando diluir menos la información del píxel central, y que el tipo 4 (donde no hay carretera) no genere tantos problemas con otros patrones que no sean carretera por su color y añadiendo un mayor contexto al pixel central.

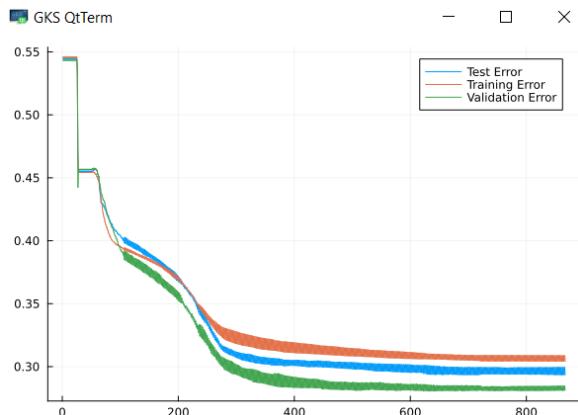
- **15x15:**

En cuanto al **Algoritmo RRNNAA**, creamos redes neuronales con topología [12 12] y [20 20 12] y a su vez usamos un bitrate de aprendizaje entre 0.0015 y 0.002. Con un aprendizaje bajo, al no generar grandes saltos en el aprendizaje, se impide saltar los valores de salida óptimos con facilidad. Decidimos añadir 150 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.

Con una topología [12 12] y aprendizaje 0.0015:



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



Test- Error Minimo: 0.2935011111778485 | Sensibilidad: 0.6950197622596457

Topología y aprendizaje	VP	FP	VN	FN
[12 12] y 0.0015	9778	4365	13906	5249
[20 20 12] y 0.002	6859	4922	13283	8234

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.

Donde el error es: 0.3960328317373461

Gamma = 10 y Coste = 20.

Donde el error es: 0.42339261285909713

Gamma=20 y Coste=40

Donde el error es: 0.32230311052283256

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
<b>Gamma = 20 Coste = 40</b>	828	415	1220	559

En cuanto a **Decision Tree** obtuvimos:

Con profundidad 8:

Error: **error es: 0.29232986966184155**

Con profundidad 10:

Error: **error es: 0.29091837347588445**

Con profundidad 15:

Error: **error es: 0.3010391014475344**

Profundidad	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>8</b>	10443	5230	13121	4711
<b>10</b>	10668	5230	12943	4457

<b>15</b>	10236	5214	13038	4810
-----------	-------	------	-------	------

Finalmente con **KNN** los resultados fueron:

Vecinos	VP	FP	VN	FN
22	10015	4440	13699	5144
24	10101	4386	13671	5140

Vecinos = 22

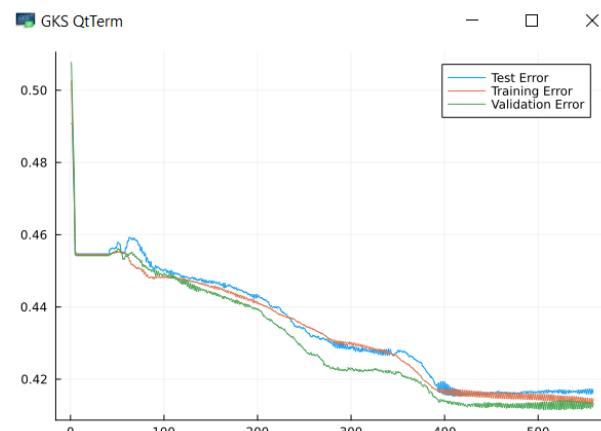
```
vp 10015 fp 4440 vn 13699 fn 5144
Donde el error minimo es es: 0.2878250946002763
```

Vecinos= 24

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

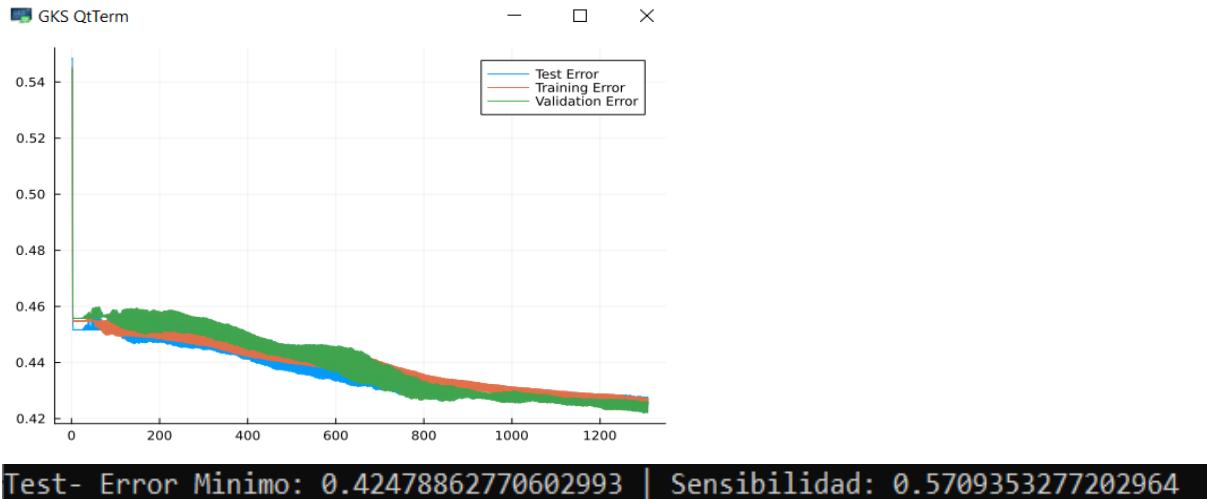
- 25x25:

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:



Topología y aprendizaje	VP	FP	VN	FN
[12 12] y 0.0015	4182	3613	13864	10630
[20 20 12] y 0.002	6263	5069	12565	8392

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.

Donde el error es: 0.39363885088919287

Gamma = 10 y Coste = 20.

Donde el error es: 0.426812585499316

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>	748	600	1025	551
<b>Gamma = 10 Coste = 20</b>	715	648	961	600

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
```

Profundidad	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>15</b>	8547	7414	10232	6096
<b>17</b>	8529	7322	10300	6138

<b>15</b>	8562	7428	10170	6129
-----------	------	------	-------	------

Finalmente con **KNN** los resultados fueron:

Vecinos = 22

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

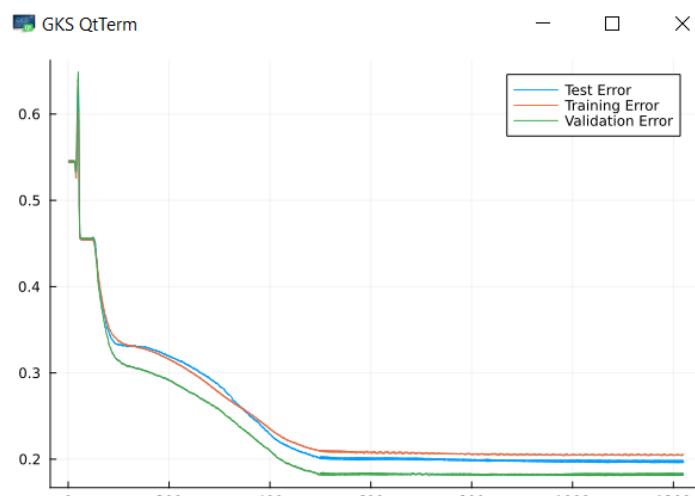
Vecinos= 24

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

Vecinos	VP	FP	VN	FN
22	7659	5445	12107	7078
24	7587	5591	12028	7083

- **5x5:**

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



```
Entrenamiento- Error Minimo: 0.2051315574440268 | Sensibilidad: 0.7948684425559732  
Test- Error Minimo: 0.1961701376632545 | Sensibilidad: 0.7948684425559732  
Validacion- Error Minimo: 0.18123284089423455 | Sensibilidad: 0.7948684425559732
```

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.

```
Donde el error es: 0.19406392694063926
```

Gamma = 10 y Coste = 20.

```
Donde el error es: 0.21428571428571427
```

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>	1091	312	1380	1061
<b>Gamma = 10 Coste = 20</b>	1038	330	1371	327

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
```

Profundidad	VP	FP	VN	FN
7	12050	3995	14474	3477
10	11912	3401	15203	3480
15	8562	7428	10170	6129

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
```

Vecinos	VP	FP	VN	FN
17	11760	3185	15352	3699
20	11517	2889	15531	4059

- **Discusión Aproximación 2:**

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.

Para la siguiente aproximación probaremos a añadir información a nuestra red neuronal. Probaremos a añadir varios tamaños de ventana, grande y pequeño a pares, para probar si la pérdida de información de los vecinos del tamaño pequeño puede ser compensada con el tamaño grande y si la pérdida de información del píxel central puede ser compensada por el tamaño pequeño.

- **3<sup>a</sup>-Aproximación:**

Como tercera aproximación decidimos utilizar distintos tamaños de ventana como input. Gracias a los resultados obtenidos en la aproximación 2, podemos observar que un menor tamaño de ventana ayuda a distinguir mejor si un píxel es carretera o, sin embargo ello también conlleva la pérdida de información de los píxeles vecinos al píxel central.

Debido a estos problemas decidimos añadir 6 nuevas entradas a la red neuronal, desviación típica y media de una segunda ventana. Estas entradas serán ordenadas añadiendo primero el tamaño pequeño y luego el grande, dándole más peso al principio del entrenamiento a la clasificación del píxel central mediante el tamaño menor y luego ir mejorando la clasificación de este gracias al contexto de la ventana grande. En cuanto a normalización, número de patrones y cross-validation mantenemos los de la aproximación 1.

En esta aproximación mezclaremos las soluciones para los patrones tipo 3 (menor cantidad de pixel carretera) y tipo 4 (donde no hay carretera). Usando el tamaño mayor añadimos mayor información de vecinos para el tipo 4 y con el tamaño menor disminuimos la pérdida de información de color del pixel central para el tipo 3.

El objetivo es observar si el uso de un tamaño de ventana pequeño, para obtener mayor precisión en el píxel central, junto a un tamaño mayor, para obtener mayor contexto en sus vecinos, es de utilidad a la hora de resolver el problema.

Los pares de ventanas los sacamos, primero obteniendo la ventana de mayor tamaño como en las aproximaciones anteriores y posteriormente sobre esa ventana sacamos como una subventana la de menor tamaño.

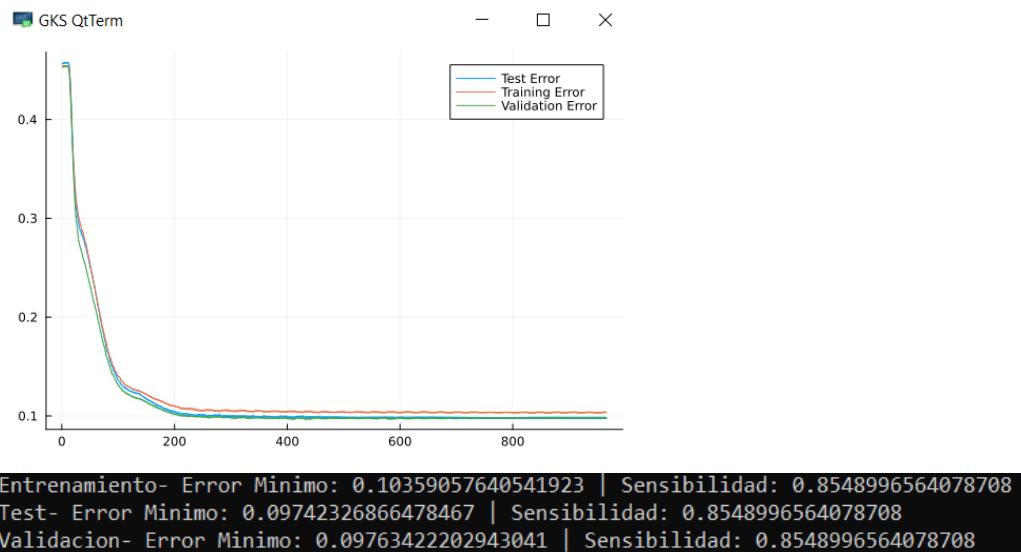
Un ejemplo de los tamaños de ventana a pares con los que trabajaremos:

-7x7 junto con 25x25

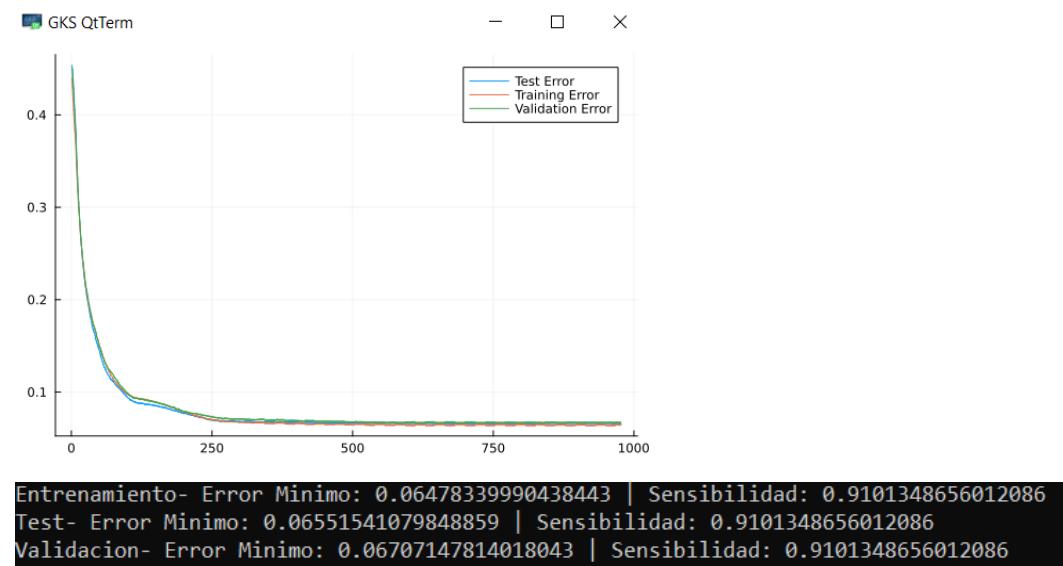


En el **Algoritmo RRNNAA** añadiremos una nueva topología [24 12] debido a que ahora la entrada son 12 inputs y no 6. También aumentaremos la tasa de aprendizaje a 0.0025 como respuesta a haber aumentado la primera capa oculta de 12 a 24.

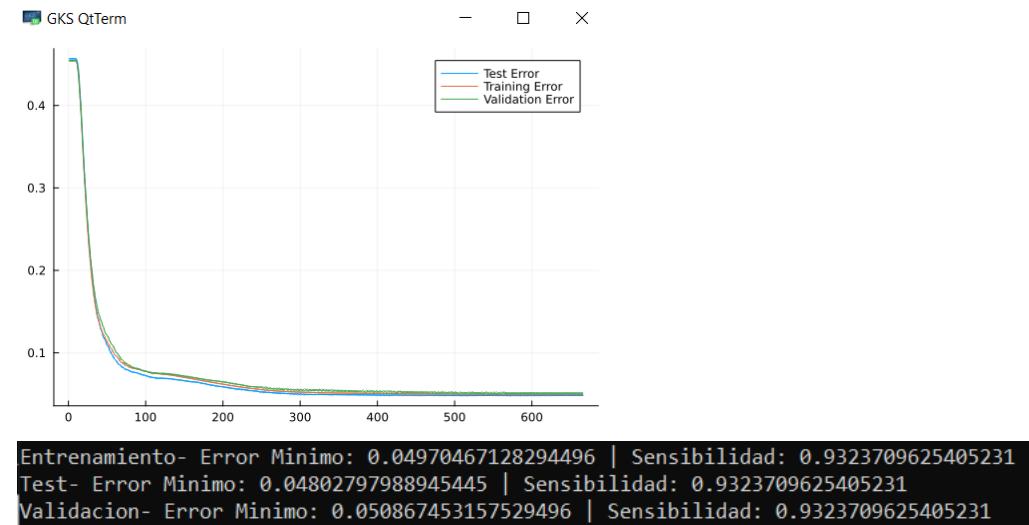
Para tamaños de ventana 7x7 y 15x15:



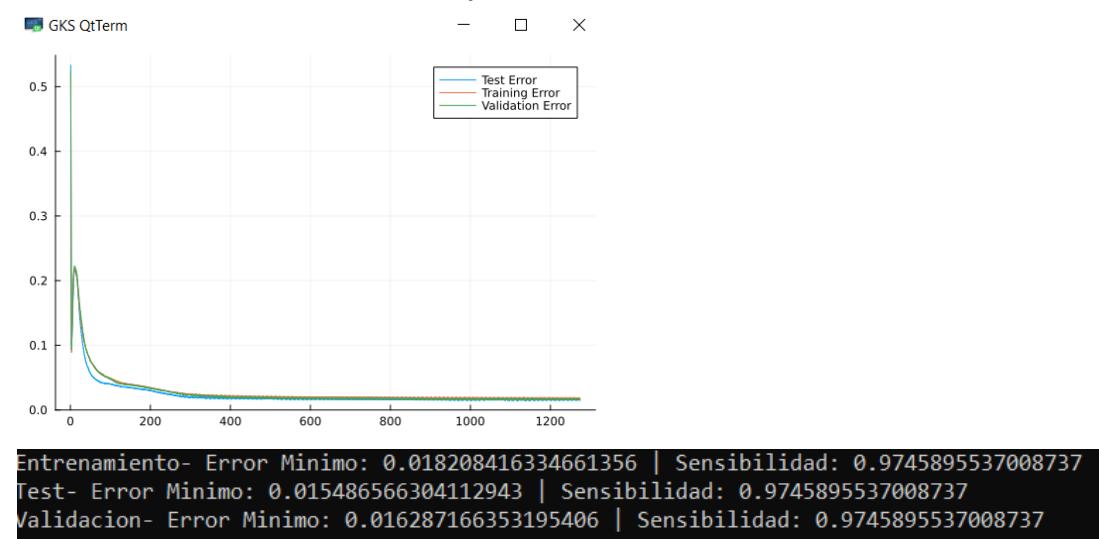
Para tamaños de ventana de 7x7 y 30x30:



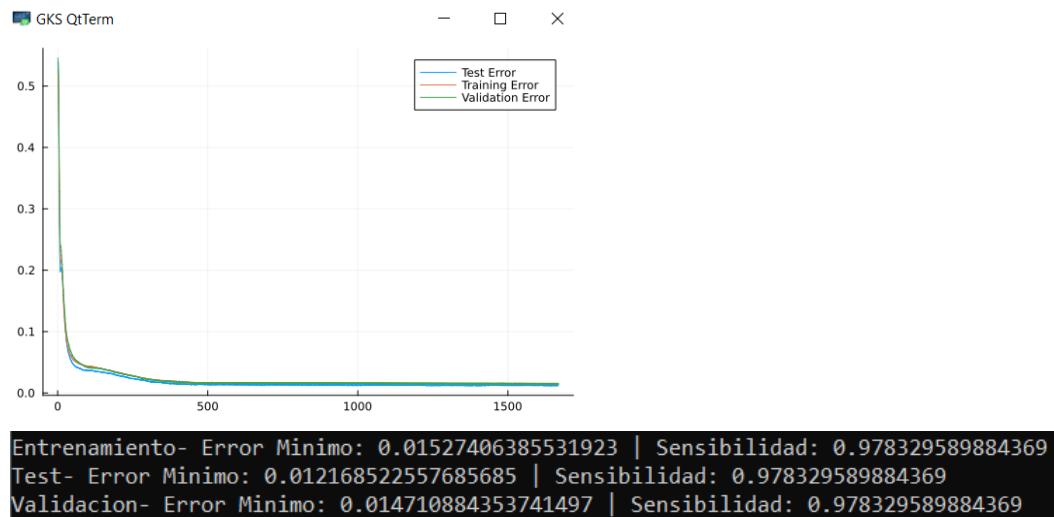
Para tamaños de ventana de 5x5 y 30x30 :



Para tamaños de ventana de 5x5 y 80x80 :



Para tamaños de ventana de 5x5 y 100x100 :



Ventanas	VP	FP	VN	FN
<b>7x7 y 15x15</b>	13146	1163	16908	2081
<b>7x7 y 30x30</b>	13285	774	11634	1130
<b>5x5 y 30x30</b>	13658	580	16819	966
<b>5x5 y 80x80</b>	12295	146	15040	285
<b>5x5 y 100x100</b>	11754	93	14057	229

Podemos observar que para este algoritmo, nuestra red neuronal funciona bastante mejor logrando un error mínimo del 1.5%, consiguiendo clasificar mejor el píxel central de un tamaño de ventana pequeño en base a un contexto mayor. Comprobamos también que la sensibilidad es bastante alta, consiguiendo clasificar correctamente un 97.8% de píxeles de carretera.

En el **algoritmo SVM** usaremos la función SVC() con distintas gamma para ajustar nuestro espacio de búsqueda de una solución óptima y conseguir una clara diferenciación entre clases. De nuevo, disminuimos el número de datos de entrenamiento para encontrar una solución óptima en un tiempo razonable, para ello cambiamos el salto de ventana de 3 a 10:

- Tamaño 7x7 y 25x25  
Donde el error es: 0.04377564979480164
- Tamaño 7x7 y 30x30  
Donde el error es: 0.03827751196172249
- Tamaño 5x5 y 30x30  
Donde el error es: 0.03212576896787423
- Tamaño 5x5 y 80x80  
Donde el error es: 0.011750881316098707

	<b>VP</b>	<b>FP</b>	<b>VN</b>	<b>FN</b>
<b>Tamaño 7x7 y 25x25</b>	1259	65	1537	63
<b>Tamaño 7x7 y 30x30</b>	1261	46	1553	66
<b>Tamaño 5x5 y 30x30</b>	1281	44	1551	50
<b>Tamaño 5x5 y 80x80</b>	1131	11	1392	19

Para el algoritmo **Decision Tree** variando el hiper parámetro profundidad máxima del árbol obtuvimos:

Con tamaño 7x7 y 25x25:

Profundidad 10:

Donde el error es: 0.06548418324329389

Profundidad 15:

Donde el error es: 0.06055022952253068

Con tamaño 7x7 y 30x30:

Profundidad 10:

Donde el error es: 0.13342005017188516

Profundidad 15:

Donde el error es: 0.12793830716343027

Con tamaño 5x5 y 30x30:

Profundidad 10:

Donde el error es: 0.049901633201136684

Profundidad 15:

Donde el error es: 0.044749086594010556

Con tamaño 5x5 y 80x80:

Profundidad 10:

Donde el error es: 0.017287329827847007

Profundidad 15:

Donde el error es: 0.017143268745948282

La matriz de confusión para cada tamaño de ventana cogiendo la mejor profundidad para cada caso es la siguiente:

Ventanas	VP	FP	VN	FN
7x7 y 15x15	12802	2265	15356	1866
7x7 y 30x30	13429	766	16655	1173
5x5 y 30x30	13737	545	16853	888
5x5 y 80x80	12438	221	14852	255

Finalmente para el algoritmo **KNN** probamos distintas combinaciones del número de vecinos. 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. Obtuvimos los siguientes resultados:

Con tamaño 7x7 y 25x25:

Vecinos 15:

```
vp 13027 fp 1533 vn 16027 fn 1702  
Donde el error minimo es es: 0.10018891882684505
```

Vecinos 20:

```
vp 12861 fp 1425 vn 16204 fn 1799  
Donde el error minimo es es: 0.0998482455325343
```

Vecinos 25:

```
vp 13124 fp 1568 vn 15963 fn 1634  
Donde el error minimo es es: 0.09916689894391278
```

Con tamaño 7x7 y 30x30:

Vecinos 25:

```
Donde el error minimo es es: 0.04643537457452456
```

Vecinos 30:

```
Donde el error minimo es es: 0.045998188801798706
```

Vecinos 40:

```
Donde el error minimo es es: 0.04827780033101209
```

Con tamaño 5x5 y 30x30:

Vecinos 15:

```
Donde el error minimo es es: 0.033632076944696
```

Vecinos 20:

```
Donde el error minimo es es: 0.03703588046091871
```

Vecinos 25:

```
Donde el error minimo es es: 0.036317646691440525
```

Con tamaño 5x5 y 80x80:

Vecinos 1:

```
Donde el error minimo es es: 0.010804581142404379
```

Vecinos 2:

```
Donde el error minimo es es: 0.011632932363322048
```

Vecinos 4:

```
Donde el error minimo es es: 0.010156306273860117
```

La matriz de confusión para cada tamaño de ventana cogiendo el mejor número de vecinos para cada caso es la siguiente:

Ventanas	VP	FP	VN	FN
7x7 y 15x15	13124	1568	15963	1634
7x7 y 30x30	13678	558	16872	915
5x5 y 30x30	13938	480	17008	597
5x5 y 80x80	12533	89	14951	193

- **Discusión Aproximación 3:**

Podemos observar en los resultados una mejoría bastante significativa a la hora de clasificar mejor los patrones tipo 3 (menor cantidad de pixel carretera) y tipo 4 (donde no hay carretera), esto puede verse al ver el decrecimiento de los falsos positivos y falsos negativos. Estos eran los que generaban errores en las aproximaciones anteriores al tener poco contexto de los vecinos o perder mucha información del píxel central. A su vez también mejoramos los pocos casos donde un kernel todo carretera era considerado no carretera, por haber un abrol como “ruido” en medio de la carretera, o donde un píxel con mucha carretera era considerado no carretera por, a lo mejor, tener unas tonalidades justas dentro del kernel para dar una desviación y media propia de una no carretera. Todos estos problemas son solucionados gracias a evitar perder todo tipo de información agregando pares de ventanas.

## **5 - Conclusiones**

Como primera parte comentaremos los resultados obtenidos en la última aproximación. En base a ellos podemos observar cómo se obtiene un error mínimo muy bajo y una sensibilidad a la hora de clasificar bien los píxeles carretera muy alta. Estos datos son bastante buenos teniendo en cuenta que solamente estamos resolviendo este problema mediante ventanas y su media y desviación típica en RGB.

Los tipo 1 y tipo 2 de por sí ya eran clasificados con exactitud en las primeras aproximaciones, pero en esta última mejoramos su clasificación junto con una mejora significativa en los tipo 3 (menor cantidad de pixel carretera) y tipo 4 (donde no hay carretera). Podemos observar como los falsos positivos son bastante menores, esto debido a que factores como casas ya no los clasifica como carretera y los falsos negativos como bordes de poca carretera los clasifica como carretera.

Continuando con las conclusiones discutiremos la viabilidad de estos sistemas a la hora de resolver este problema. Todos ellos resuelven el problema mediante color solamente, siendo muy dependiente el tamaño de las ventanas y su relación entre ellas

para obtener una desviación típica y media representativa del píxel central. Con otros sistemas como redes convolucionales podríamos minimizar el impacto de esta diferencia al ser la propia red la que regule los pesos que toma cada píxel dentro de un kernel. Este sistema también podría agregar una solución añadida al distinguir bordes y aprender a diferenciarlos añadiendo una mejora a los píxeles de tipo 3 y 4.

El mejor de los modelos podría ser el RRNNAA debido a que obtiene resultados muy parecidos al resto, en algunos casos hasta bastante mejores, y al ser no supervisado da la mejora de no tener que ir cambiando los hiper parámetros en base a la cantidad de parámetros de entrada. Esto puede suponer una mejora a la hora de añadir más imágenes de entrada, cambiar el salto entre ventanas o si se tiene la necesidad de cambiar los input para añadir alguna aproximación nueva. En base al error no se pudo comprobar que sea categóricamente mejor que el resto debido a que todas dan resultados bastantes similares al clasificar solamente por color el problema.

Otro algoritmo que soluciona bastante bien el problema es SVM, el problema con este algoritmo es la dificultad que tiene a la hora de encontrar una solución del hiperplano para las carreteras. A más patrones de entrada tarda mucho más que el resto, y al ser un entrenamiento supervisado, hace mucho más difícil y costoso el encontrar un hiperparametro adecuado para la resolución del problema en un tiempo razonable.

## **6 - Trabajo futuro**

En base a los resultados, discusiones y la propia naturaleza del problema, podríamos utilizar estas soluciones para otros problemas como encontrar moho en frutas, nieve en la carretera, marcar frutas en un árbol u otros problemas para detectar X color en una imagen.

Este problema podría aplicarse al mundo real, por ejemplo, para una aplicación tipo maps , donde en base a una imagen de una ciudad pudiera sacar las carreteras y realizar un recorrido por ellas y no recomendar una ruta por fuera de la carretera.

Nuevas técnicas a añadir podrían ser las redes convolucionales para añadir bordes a la solución del problema, aprendiendo y guardando en las capas ocultas distintas formas de la carretera y clasificándolas mejor al no solamente tener en cuenta el color.

Podríamos utilizar este sistema para resolver problemas más complejos como, por ejemplo, contar los árboles existentes en una imagen. Generando un mapa solo con los arboles resaltados en blanco y el resto en negro, podría utilizarse teoría de conjuntos para calcular los árboles.

## **7 - Bibliografía**