Proyecto Fin de Master

Master en Ingeniería Industrial

Estabilidad al vuelco de vehículos de carretera mediante técnicas de Machine Learning

Autor: Javier Díaz Díaz

Tutor: Francisco José Morales Sánchez

Francisco de Asís García Benítez

Área de Ingeniería e Infraestructura de los Transportes
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Máster
Máster en Ingeniería Industrial

Estabilidad al vuelco de vehículos de carretera mediante técnicas de Machine Learning

Autor:
Javier Díaz Díaz

Tutores:
Francisco José Morales Sánchez
Francisco de Asís García Benítez

Área de Ingeniería e Infraestructura de los Transportes
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de máster: Estabilidad al vuelco de vehículos de carretera mediante técnicas de Machine Learning

Autor: Javier Díaz Díaz

Tutor: Francisco José Morales Sánchez

Francisco de Asís García Benítez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

Este proyecto simboliza el fin de una etapa marcada por un sinfín de momentos y personas. Sería difícil enumerar todas las personas que han vivido estos duros, y a la vez formidables, años de estudiante a mi lado. No podría haberme planteado llegar hasta donde estoy ahora sin mi familia, luchadora y tenaz. Han estado siempre a mi lado y no se han rendido a pesar de los grandes contratiempos que hemos vivido.

A mis padres agradecer toda la paciencia que han tenido conmigo durante todos estos años, el apoyo que me han mostrado, tanto en los malos momentos como en los buenos. He crecido como persona gracias a ellos, he estudiado gracias a ellos y gracias a ellos también he sido capaz de culminar esta etapa de mi vida.

A mi hermana Elena, un gran pilar en mi vida personal y académica. Desde que empecé mi etapa de estudiante de ingeniería siempre ha estado a mi lado, entendiéndome, aconsejándome, pero sobre todo mostrando su compresión en todas las situaciones que hemos vivido.

A todo mi alrededor, mis amigos, "mis hermanos", gracias por darme fuerzas y sentiros orgullosos de todas las horas que no os he podido dedicar por tener que seguir estudiando. En especial a Bea, compañera de estudios y de vida, un pilar fundamental que me ha impulsado durante estos últimos años, haciendo que sea mejor persona cada día.

Por último agradecer a Fran, la otra persona encargada de que este proyecto cobrara vida, su esfuerzo, dedicación y compresión en estos momentos tan difíciles a nivel mundial y personal.

Gracias.

Javier Díaz Díaz

Sevilla, 2021

Resumen

Este proyecto nace con la intención de continuar con el estudio del vuelco de vehículos, tras la realización del proyecto fin de grado "Sistema Predictivo Anti-vuelco de Vehículos". Anteriormente el estudio se centró en la física de este tipo de maniobras y en las variables que juegan un papel fundamental, para ser capaces de realizar estimaciones cercanas a la realidad.

En este proyecto, yendo un paso más lejos, se va a intentar implementar la metodología Machine Learning al problema del vuelco de vehículos. El Machine Learning es una rama de la inteligencia artificial la cual, mediante el tratamiento de datos, es capaz de enseñar a un programa a predecir valores discretos o continuos.

La aplicación de esta metodología se realiza inicialmente con una breve introducción al vuelco y a las variables mecánicas más significativas. Posteriormente, se realiza una explicación del Machine Learning con sus diferentes variaciones como son los árboles de decisión y las redes neuronales.

A continuación, se realizan diferentes pruebas para determinar que metodología dentro del Machine Learning se adapta mejor a este problema, comparando los algoritmos típicos como son SVM (Máquinas de vectores soporte) y los árboles de decisión con las redes neuronales. Una vez realizada esta comparación, el proyecto se centra en la optimización del proceso para la predicción del índice de vuelco en diferentes situaciones y con diferentes vehículos, donde se verá la importancia de los datos y su obtención.

En la fase final del proyecto, se realizan dos hipótesis para la generalización del problema. Además, se realiza un estudio de la viabilidad de ambas y de cómo se podrían aplicar a los diferentes tipos de vehículos.

Se concluye que la aplicación del Machine Learning, en concreto las redes neuronales, son una herramienta con un gran potencial en la predicción del vuelco de vehículo. Las redes neuronales son capaces de predecir valores de parámetros que, en condiciones normales y con dispositivos que se puedan incorporar a un vehículo cualquiera, serían difíciles de obtener. Además, no solo son capaces de obtener parámetros o directamente el valor del índice de vuelco, sino que también son capaces de diferenciar entre distintos tipos de vehículos y obtener una buena respuesta para cualquiera de ellos.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Los accidentes de vehículos tienen un gran impacto económico y social en la actualidad. Especialmente, los accidentes de tráfico en los que están involucrados vehículos pesados, provocando daños más graves al ser más propensos a volcar.

En España en el año 2019 se produjeron un total de 104.080 accidentes con víctimas, un 2% más que en 2018. Sin embargo, el número de fallecidos por vuelco de vehículos aumento en un 23% y el número de hospitalizados aumentó un 17%. Concretamente, se han producido 600 muertes provocadas por el vuelco de un vehículo agrícola en la última década. El vuelco de vehículos agrícolas supone casi un 60% de todos los siniestros mortales por maquinaria y más del 50% de todos los siniestros mortales en el sector agrícola.(DGT, 2019)

Debido a este problema muchas investigaciones se centran en el desarrollo de sistemas de control de estabilidad. Estos sistemas son capaces de detectar el deslizamiento lateral que se produce en los vehículos al realizar maniobras en curvas o en situaciones de aceleración. En Europa se hizo de obligatorio cumplimiento que todos los vehículos, turismos o ligeros, nuevos a partir de 2014 tuviese que estar presente en el vehículo algún tipo de control de estabilidad.

Estos sistemas recogen información del vehículo hasta 25 veces por segundo para medir la variación con la trayectoria que debería de seguir el vehículo y con la real. Durante muchos años se han estado desarrollando este tipo de sistemas, donde inicialmente el sistema emitía señales al conductor y finalmente en 1995 Bosch, en colaboración con Mercedez-Benz desarrollo el primer sistema comercial (ESP) que controlaba la trayectoria del vehículo.

Estos sistemas llevan asociada una tecnología de procesado y recogida de datos dinámica instantánea para poder avisar, controlar o corregir el vuelco mediante la actuación directa sobre actuadores del vehículo como pueden ser sistemas de amortiguación activa, control del freno mediante el ABS o incluso control del sistema de dirección.

El principal problema de estos sistemas es la falta de capacidad para obtener directamente algunas variables dinámicas requeridas para la predicción del vuelco, como puede ser el ángulo de balanceo. A pesar de la utilidad de las medidas proporcionadas por dispositivos de un coste bajo, la predicción de valores para algunas de las variables necesarias no es lo suficientemente exacta, por lo que se dificulta la opción de comercializar estos sistemas de forma masiva.

En la actualidad, y con vistas al futuro con el vehículo autónomo, se están desarrollando multitud de sistemas basados en la inteligencia artificial, más concretamente en el Machine Learning y el Deep Learning. Estos sistemas son capaces de tratar y trabajar con millones de datos e imágenes de forma simultánea para ser capaces de predecir comportamientos y actuar en consecuencia.

Aprovechando la nueva dirección en la que se orienta el futuro del mercado de vehículos, junto con la capacidad de predicción de estos sistemas se abre la posibilidad de mejorar sustancialmente en la predicción de datos, situaciones y peligros con el fin de evitar accidentes y posibles víctimas.

1.1 Física del vuelco

1.1.1 Vuelco Cuasi-estático

1.1.1.1 Vehículo Rígido (SSRT: Steady State Rollover Threshold)

En la aproximación más simple para iniciar el estudio físico de la mecánica de vuelco, se supone un vehículo rígido el cual está definido por no tener suspensión. Se supone también que este vehículo tiene una masa m y describe una curva con peralte, como se muestra en la ilustración 1 (WIDEBERG, 2016).

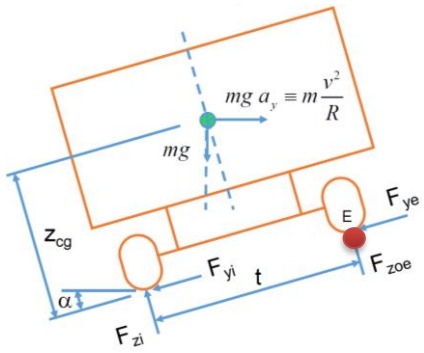


Ilustración 1 : Esquema Vehículo Rígido

Fuente: (WIDEBERG, 2016)

Donde:

- m : Masa del vehículo
- α : Peralte de la Curva
- a_y : Aceleración Lateral
- F_z : Fuerzas verticales — i: interior
- F_y : Fuerzas horizontales — e: exterior
- Z_{cg} : Altura al centro de gravedad
- v : Velocidad lineal del vehículo
- R : Radio de la curva
- g : Gravedad
- t : Vía del vehículo

Se realiza el sumatorio de fuerzas (1) (2) y de los momentos (3) respecto del centro del contacto de los neumáticos exteriores (punto E).

$$\sum F_y = 0 \quad \Rightarrow \quad mg\, a_y \cos\alpha - mg \sin\alpha = F_{yi} + F_{ye} \tag{1}$$

$$\sum F_z = 0 \quad \Rightarrow \quad mg\, a_y \sin\alpha + mg \cos\alpha = F_{zi} + F_{ze} \tag{2}$$

$$\sum M_E = 0 \quad \Rightarrow \quad mg\, a_y z_{cg} \cos\alpha - mg\, z_{cg} \sin\alpha - mg\, a_y \sin\alpha \frac{t}{2} - mg \cos\alpha \frac{t}{2} + F_{zi} t = 0 \tag{3}$$

La aceleración lateral (a_y) es una variable que permite definir el vuelco (4). Se puede calcular despejando de la ecuación (3), quedando en función de la carga normal que soportan las ruedas interiores para sistemas cuasi-estáticos:

$$a_y = \frac{z_{cg}\tan\alpha + \frac{t}{2} - \frac{F_{zi}t}{mg\cos\alpha}}{z_{cg} - \frac{t}{2}\tan\alpha} \qquad (4)$$

Para hallar la aceleración crítica (5) para el inicio del vuelco se toma como punto de referencia cuando la rueda interior se despega del suelo, que es el momento en que la fuerza normal del neumático interior es igual a cero ($F_{zi} = 0$). Además, para mayor simplificación, se supone que el peralte de la carretera es nulo ($\alpha = 0$). Operando y simplificando queda finalmente una definición de la aceleración lateral crítica mediante parámetros geométricos a_y*:

$$a_y^* = \frac{t}{2z_{cg}} \qquad (5)$$

Este parámetro, también llamado *Índice de Estabilidad Estático o Umbral de vuelco,* se usa como primera aproximación, ya que clasifica cualquier vehículo con tan solo dos parámetros geométricos constantes. Además, también es posible la expresión de una velocidad crítica (6) o velocidad de vuelco (v_{vuelco}). Definida a partir de la altura del centro de gravedad del vehículo y el radio de curvatura.

$$v_{vuelco} = \sqrt{\frac{t}{2z_{cg}}gR} \qquad (6)$$

Tras definir el Umbral de vuelco, se pueden agrupar los vehículos en tres tipos para hacer una comparación simplificada como se muestra en la Tabla 1.

Tipo de vehículo	Altura CDG (cm)	Vía (cm)	Umbral de vuelco
Turismo	45-60	127-165	1,1-1,7
Furgoneta	75-100	165-178	0,8-1,1
Camión	150-215	178-187	0.4-0,6

Tabla 1 : Comparación umbral de vuelco estático

Al analizar esta primera aproximación se tiene que el umbral de vuelco para los camiones es muy pequeño. Sin embargo, para los turismos este umbral es mayor que la adherencia de los neumáticos (μ) que, para una vía en buen estado, tiene un valor alrededor de 0,8. Es decir:

$\frac{t}{2h} > \mu$ → El vehículo se desliza

$\frac{t}{2h} < \mu$ → El vehículo vuelca

1.1.1.2 Vehículo con suspensión.

En este caso se tendrá en cuenta un vehículo similar al caso anterior, pero con suspensión (Ilustración 2). La elasticidad de la suspensión disminuye el umbral de vuelco, ya que permite que el centro de gravedad de la masa suspendida se desplace hacia el exterior, disminuyendo el brazo de la fuerza restauradora.

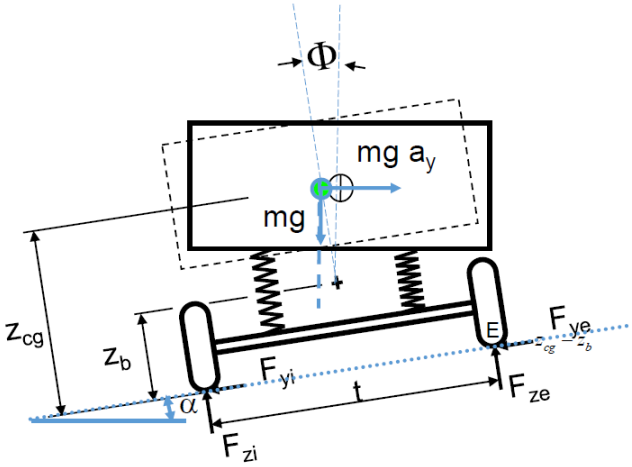


Ilustración 2 : Esquema Vehículo con Suspensión

Fuente: (WIDEBERG, 2016)

Cuando un automóvil describe una curva, se produce una transferencia de carga lateral como consecuencia de la transferencia vertical de las ruedas interiores a las exteriores. Si la variación de la carga lateral no es lineal, la medida de fuerzas laterales es inferior a la que recae sobre el eje en cuestión (perdida de equilibrio). Este equilibrio se puede reestablecer acomodando un ángulo de deriva superior.

La contribución será subviradora o sobreviradora dependiendo de la capacidad de la suspensión para transmitir el momento de balanceo (rigidez de balanceo de la suspensión) y de la rigidez de la estructura del automóvil. Para un bastidor totalmente rígido (ángulo girado el mismo en cada eje) se tiene una rigidez al balanceo de una suspensión de eje rígido (7):

$$K_{\emptyset} = \frac{M}{\emptyset} = \frac{K_s \emptyset \frac{s}{2} s}{\emptyset} = \frac{1}{2} K_s s^2 \qquad (7)$$

Donde:

- $K_{\emptyset}$: rigidez al vuelco
- M : Momento de vuelco
- $\emptyset$: Angulo girado por la caja del vehículo
- K_s : Rigidez de cada uno de los elementos elásticos
- S: Distancia entre apoyos de la suspensión

Realizando el equilibrio de momentos (8) en el eje respecto del centro de balanceo de la masa suspendida:

$$\Delta F_z = \frac{F_{zo} - F_{zi}}{2} = \frac{1}{t}[F_y h_r + K_\phi \phi] \qquad (8)$$

Transferencia de carga debida a la fuerza centrífuga.

Transferencia de carga debida al giro del vehículo.

De la misma forma que en el caso de vehículo rígido, se pueden realizar los sumatorios de fuerzas y momentos para despejar la aceleración lateral (a_y) y aplicar las simplificaciones ($F_{zi} = 0$ y $\alpha = 0$) para obtener la aceleración crítica para el inicio del vuelco o Índice de Estabilidad Estática para vehículos con suspensión (9):

$$a_y^* = \frac{t}{2z_{cg}} - \Phi\left(1 - \frac{z_b}{z_{cg}}\right) \qquad (9)$$

También se utiliza otra forma de expresar esta ecuación (Sanjuán, 1994) , utilizando para ello el *coeficiente de balanceo* $R_{\emptyset}$ (10*)* el cual queda definido por la variación del ángulo del vehículo respecto a la aceleración lateral (11) que experimenta el vehículo:

$$R_\phi = \frac{d\phi}{da_y} \approx \frac{\phi}{a_y} \qquad (10)$$

$$a_y = \frac{\frac{t}{2z_{cg}}}{1 + R_\Phi\left(1 - \frac{z_b}{z_{cg}}\right)} \qquad (11)$$

Si se grafica la aceleración lateral frente al coeficiente de balanceo se obtiene una curva (Ilustración 3) que muestra cuando se produce la inestabilidad en el vehículo en función del ángulo y de la aceleración. En esta gráfica se puede ver que, para vehículos con suspensión, la aceleración y el ángulo de balanceo son proporcionales y que la pendiente es el coeficiente de balanceo definido anteriormente. Además, el final de la zona de proporcionalidad indica el comienzo de la zona de inestabilidad, que coincide con el momento en que la rueda se ha levantado del suelo y está empezando el vuelco.

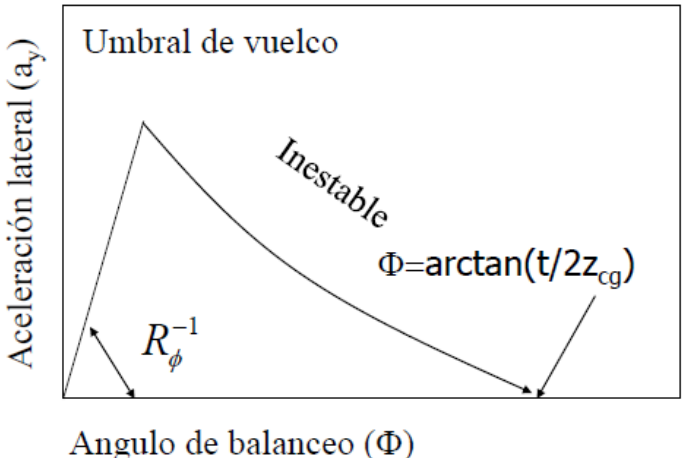


Ilustración 3 : Umbral de vuelco vehículos con suspensión.

Fuente: (WIDEBERG, 2016)

1.1.2 Vuelco dinámico

Actualmente el desarrollo de sistemas dinámicos está centrado en los sistemas de seguridad activa desarrollados por la industria del automóvil. Estos sistemas suelen utilizar modelos dinámicos que no integran modelos de neumático o integran modelos lineales. Sin embargo, para las situaciones fuera de carretera los efectos de deslizamiento que se producen pueden tener efectos muy importantes sobre la dinámica del vehículo.

Los modelos aplicados se basan en la transferencia de carga lateral (12) (Bouton *et al.*, 2007). Este parámetro puede tomar valores en el rango -1 a 1, donde 0 indica la total estabilidad y 1 o -1 indica que el vuelco y la inestabilidad han comenzado:

$$LLT = \left(\frac{F_{n2} - F_{n1}}{F_{n2} + F_{n1}} \right) \tag{12}$$

Donde:

- F_{n1} : Fuerza normal en la rueda derecha
- F_{n2} : Fuerza normal en la rueda izquierda

Si:

- $|LLT| = 1$: Se han levantado dos ruedas del mismo lado y comienza el vuelco
- $|LLT| < 0.8$: Situación de estabilidad
- $|LLT| > 0.8$: Situación de inestabilidad

1.2 Sensibilidad de los parámetros

Para entender como poder realizar ciertas mejoras o simplificaciones sin influir en la calidad de las lecturas y los avisos, es de especial utilidad el estudio de los parámetros más significativos en el vuelco. De esta forma, se puede conocer cómo pueden influir ciertas variaciones en los vehículos.

Este análisis se basa en los estudios previos de (Lock, 2000) (RP, BC and SJ, 2006), en los que se detecta el efecto en el tiempo de vuelco de variaciones de ±10% en varios parámetros. Este tiempo de vuelco se define en base a una variación constante del ángulo de inclinación del terreno, para distintos tipos de vehículo.

Como se puede observar en la ilustración 5, en el caso de las furgonetas, camiones pequeños y vehículos de geometría similar, los parámetros con mayor influencia en el vuelco son la altura del centro de gravedad, la velocidad y el radio de la curva, siendo también significativas las vías delanteras y traseras.

Se puede ver que aumentar un 10% los parámetros más importantes provocan disminuciones del tiempo de vuelco entre el 10% y el 20%. Además, también se observa que el parámetro que más estabilidad proporciona es la disminución del centro de gravedad.

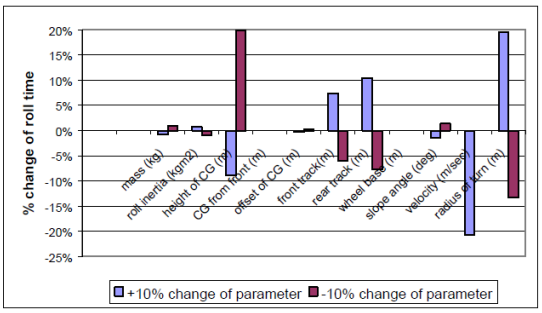


Ilustración 5 : Sensibilidad de parámetros furgonetas

Fuente: (Lock, 2000)

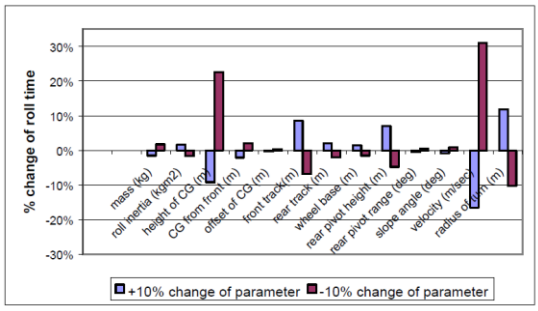


Ilustración 4 : Sensibilidad de parámetros camiones articulados

Fuente: (Lock, 2000)

En la ilustración 4, que hace referencia a camiones articulados, se puede ver que este tipo de vehículos está influenciado por los mismos parámetros que los anteriores. Sin embargo, esta vez gana mucho protagonismo la velocidad ya que, al ser vehículos muy pesados y con grandes masas suspendidas, provoca una gran influencia en la dinámica de vuelco. No obstante, esta velocidad es fácil de medir por lo que no supone ningún problema.

Por último, en la ilustración 6 se muestra el caso de camiones o camionetas con volquete. En ella se observa que la sensibilidad es menor que en los casos anteriores para la altura del centro de gravedad y se aumenta para los anchos de vías. A pesar de estas diferencias, los parámetros de interés son los mismos que para los demás casos.

Se puede concluir que los parámetros más importantes son dos: la altura del centro de gravedad y el ancho de vías. Además, hay otros dos muy influentes, que no dependen del vehículo sino del conductor: la velocidad y el radio de la curva.

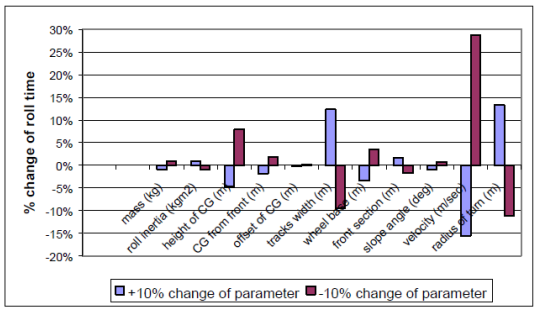


Ilustración 6 : Sensibilidad de parámetros camiones con volquete

Fuente: (Lock, 2000)

2 MACHINE LEARNING

2.1 Que es el Machine Learning

El Machine Learning (ML), o su nombre en español Aprendizaje Automático (AA), es la vertiente de la inteligencia artificial la cual tiene como objetivo desarrollar técnicas que permitan a las computadoras "aprender" (Caparrini, 2017).

Esta vertiente de la inteligencia artificial se identifica con la creación de algoritmos que generalizan el comportamiento y reconoce patrones a partir de una información suministrada. Desde el punto de vista de la estadística, el Machine Learning es un problema general de inferencia estadística. Se parte de casos particulares para llegar a una definición general, con la creación de un modelo. Se podría describir entonces la tarea del Machine Learning como la extracción de conocimiento sobre propiedades no observadas de un objeto basándose en las propiedades que si han sido observadas.

El Machine Learning engloba muchos tipos de problemas y la principal diferencia entre ellos es el tipo de objetos que intentan predecir. Los problemas más habituales son:

- ***Clasificación (multiclase o binaria)***: engloba todos los problemas donde se requiere la clasificación de objetos sobre un conjunto de clases prefijadas. Por ejemplo, clasificar o identificar que letra del abecedario ha escrito una persona en función de diferentes parámetros (longitud, anchura, tiempo de escritura, etc…). Este tipo de problema de clasificación se definiría como un problema multiclase, ya que se permiten más de dos clases (en este caso todo el abecedario).
- ***Ranking***: Se caracteriza por predecir el orden óptimo de un conjunto de objetos en función de su relevancia, previamente predefinida. Por ejemplo, el orden que un buscador de internet devuelve en una búsqueda de un usuario.
- ***Regresión***: es el tipo de problemas en el que se centra este proyecto y consiste en la predicción de un valor real. El objetivo es pronosticar un valor predicho aproximado lo más cercano posible al valor real. En este caso, a diferencia de la clasificación, se busca la predicción de una magnitud numérica, mientras que en el problema de clasificación es una magnitud discreta. Por ejemplo, predecir el consumo de un vehículo basándose en los gastos obtenidos en simulaciones.

Es importante enmarcar cada problema en alguna de estas clases. En función de cómo se enmarque cada problema, su error respecto a la realidad se medirá de forma diferente.

Dependiendo del tipo de salida y del problema se presentan diferentes tipos de algoritmos:

- ***Aprendizaje supervisado:*** se engloban en este grupo aquellos problemas donde hay una asociación entre los elementos que se van a tratar y el valor objetivo. Se obtiene una función que establece una relación entre los datos de entrada y los datos de salida, que han sido etiquetados o definidos a priori, es decir, datos que se conoce previamente su clasificación o valor real.
- ***Aprendizaje no supervisado:*** se desconoce previamente la clasificación o valor real de los parámetros objetivos. El proceso se lleva a cabo tan solo con un conjunto de entradas, por lo que se intenta que el algoritmo sea capaz de reconocer patrones y relaciones para poder predecir valores de las nuevas entradas.
- ***Aprendizaje semi-supervisado:*** es una combinación de los dos algoritmos anteriores. En este caso se contaría con ejemplos clasificados o conocidos y con ejemplos no clasificados o no conocidos.

- ***Aprendizaje por refuerzo:*** en este caso, hay un doble flujo de información. Se produce un intercambio entre el algoritmo y los datos de entrada y entre los datos de entrada y el algoritmo. Se realiza un proceso de ensayo-error reforzando aquellas acciones que reciben una respuesta positiva.
- ***Transducción:*** similar al aprendizaje supervisado. En este caso también se tienen en cuenta las predicciones de datos futuros. Por lo que se asemeja a un aprendizaje supervisado dinámico.
- ***Aprendizaje multitarea:*** conjunto de todos los métodos con el objetivo de enfrentar problemas parecidos a los ya analizados.

Los modelos de aprendizaje supervisado se definen como los modelos donde se aprenden funciones/relaciones que asocian entradas con salidas. Se ajustan al conjunto de datos de los que se conocen sus entradas y sus salidas, siendo un modelo de regresión si la salida es un valor continuo.

En un modelo de regresión (13), la relación entre las entradas o predictores y la respuesta se puede describir de forma matemática:

$$Y = \sum_{k=0}^{n} c_k f_k(x_1, x_2, \dots, x_m) \qquad (13)$$

Donde:

- Y: respuesta
- x_1, x_2, …, x_m: Entradas del modelo o predictores
- n: número de ejemplos
- c_k: coeficiente para minimizar el error de predicción.
- f_k: función de predicción de las variables de entrada. Estará definida por el modelo matemático especifico utilizado para el análisis con ML.

Dentro de los modelos de regresión se pueden diferenciar tres tipos de regresiones:

- Regresión lineal: es una de las técnicas de regresión más simples. Es una técnica paramétrica en la que la respuesta se modela como una formula conocida dada en términos de las variables predictoras (14).

$$Y_t = \beta_0 + \beta_1 X_1 + \varepsilon \qquad (14)$$

- Regresión polinomial: es un caso específico de la regresión lineal, se añaden predictores adicionales obtenidos al elevar sus términos a una potencia (15).

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \cdots + \beta_p X_1^p + \varepsilon \qquad (15)$$

- Regresión multivariable: es común el uso de la regresión multivariable porque se usan modelos lineales entrenados en funciones no lineales de los datos (16). De esta manera, se mantiene el rendimiento de los métodos lineales a la vez que se le permite ajustarse a un rango de datos más amplio.

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \qquad (16)$$

Donde:

Y_t: $Variable\ dependiente$

$X_1, X_2, \cdots, X_p$: $Variable\ explicativa, independiente.$

$\beta_1, \beta_2, \cdots, \beta_p$: $Parámetros, miden\ la\ influencia\ que\ las\ variables\ explicativas\ tienen\ sobre\ la\ variable\ dependiente.$

β_0 : $Intersección\ o\ termino\ constante.$

ε : $Termino\ aleatorio$

2.1.1 Modelos de Machine learning.

2.1.1.1 K-NN K- Nearest Neighbour

Es un modelo de Machine Learning de tipo supervisado. Se suele usar para clasificar valores discretos, nuevas muestras, aunque también se puede usar para obtener valores continuos. Es un método sencillo y se basa en la búsqueda de los puntos de datos "más similares" por cercanía que se han aprendido en la etapa de aprendizaje.

El algoritmo clasifica los nuevos datos según tenga *K* vecinos más cerca de un grupo que de otro. Lo realiza mediante el cálculo de la distancia del nuevo elemento con cada uno de los ya existente, y ordenando las distancias de menor a mayor se determina al grupo al que pertenece.

K-NN, a diferencia de otros modelos de Machine Learning de tipo supervisado, no genera un modelo tras la etapa de aprendizaje, sino que el aprendizaje sucede en el mismo momento en el que se introducen los nuevos datos.

El modelo funciona de la siguiente manera:

1. Se calculan las distancias entre los datos nuevos a clasificar y el resto de datos del dataset, que forman el conjunto de datos de la fase de entrenamiento.
2. Se seleccionan los *K* elementos más cercanos. En este paso se pueden utilizar diferentes funciones de calcular la distancia, pero el concepto es el mismo.
3. Se ordenan las distancias de menor a mayor y con ella se puede clasificar entre los K puntos más cercanos como se define o que clase se le asigna al nuevo dato

La importancia del valor *k* es muy significativa pues determina cuantos puntos se toman para comparar las distancias y por ello en un valor importante que puede determinar cómo clasificar. Es especialmente importante en las zonas de "frontera", es decir, donde colisionan dos clases diferentes de datos (Ilustración 7).

Las formas más habituales de medir las distancias entre datos son la distancia Euclidiana o la CosineSimilarity, que mide el ángulo de los vectores distancia (Thangavel, 2018).

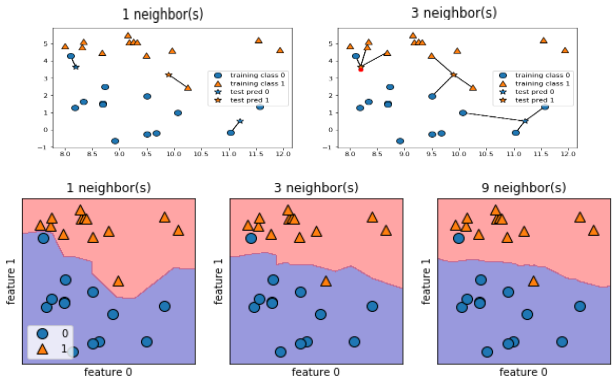


Ilustración 7: Modelo K-NN

Fuente: (Thangavel, 2018).

2.1.1.2 SVM Máquinas de vectores soporte

Este algoritmo creado por Vladimir Vapnik se puede utilizar para regresión y clasificación. La resolución se lleva a cabo mediante una etapa de entrenamiento, donde se introducen multitud de ejemplos resueltos. En segundo lugar, se lleva a cabo una etapa de prueba o test, donde se realiza la comprobación del funcionamiento del algoritmo.

De forma intuitiva el SVM es un modelo que separa las clases en espacios lo más amplios posible mediante un hiperplano. Este hiperplano se define mediante un vector que es el que recibe el nombre de "vector soporte". La función que realiza este hiperplano es la de separar en clases para que cuando se introduzca un nuevo dato se sepa en que espacio está, y así ser capaz de clasificarlo.

De una forma más teórica, el SVM, dado un conjunto datos donde cada uno de ellos puede pertenecer a cualquiera de las categorías posibles, construye un modelo capaz de predecir la categoría de un nuevo dato. El SVM busca un hiperplano que separe las diferentes categorías, que ocasionalmente ha podido ser proyectado a un espacio de dimensión superior. La forma de hallar el hiperplano es lo que caracteriza este método, buscando que dicho hiperplano tenga la máxima distancia con los datos que estén más cercanos entre sí.

En el caso de tener dos dimensiones, la implementación es muy visual (Martinez Heras, 2019). En este caso el vector soporte define una línea que separa ambas clases (puntos azules y rojos) (Ilustración 8). La línea que divide ambas clases está definida por ser la que está a mayor distancia de ambos conjuntos.

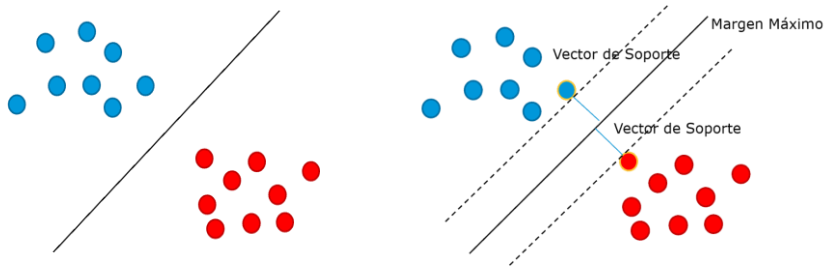


Ilustración 8: Ejemplo SVM

Fuente: (Martinez Heras, 2019)

Los vectores soporte son el instrumento para definir un hiperplano con la máxima separación. Se les llama vectores porque tienen tanto elementos como dimensiones tenga el espacio de los datos de entrada. En estos casos y cuando se da el caso de no encontrar hiperplanos que sean capaces de encontrar separaciones se recurre al método de Kernel. Este tipo de casos se definen como clases no linealmente separables.

El método de Kernel (Ilustración 9) consiste en añadir una dimensión nueva por la que se pueda encontrar un hiperplano capaz de separar las clases del problema.

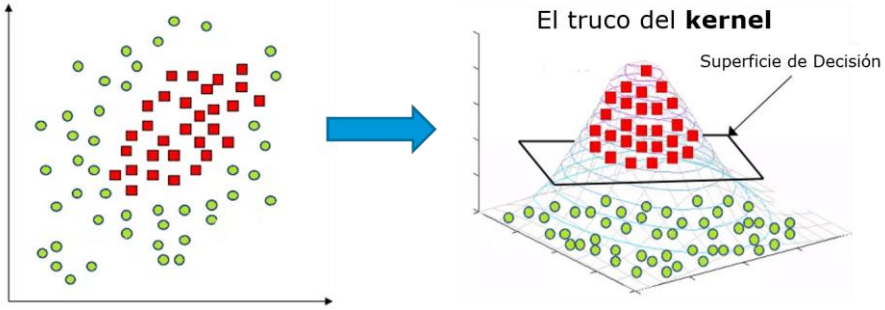


Ilustración 9: Metodo de Kernel en SVM

Fuente: (Martinez Heras, 2019)

En el caso de la regresión se utilizan los mismos principios, pero esta vez se define una curva que modela la tendencia de los datos (Ilustración 10) (Merkle, 2020). Esta curva estará definida por el vector soporte e igualmente en el problema de clasificación para problemas no lineales, se puede utilizar el método de Kernel (Ilustración 11).

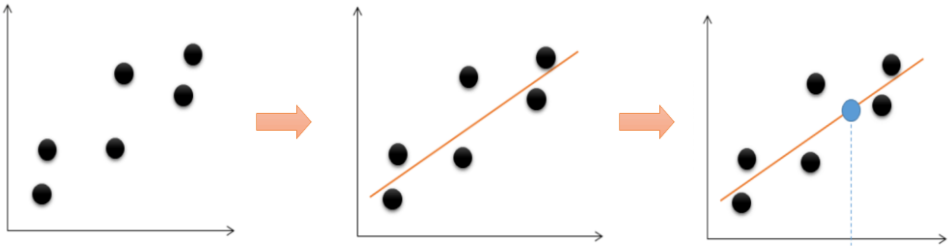


Ilustración 10: Problema de regresión con SVM

Fuente: (Merkle, 2020)

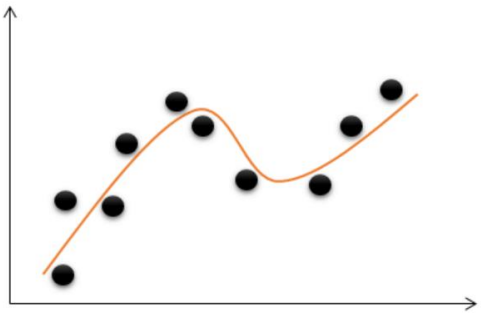


Ilustración 11: Problema de regresión con SVM aplicando Kernel

Fuente: (Merkle, 2020)

2.1.1.3 Análisis discriminante lineal (LDA) y cuadrático (QDA)

El Análisis Discriminante Lineal es un método de clasificación supervisado. Mediante el teorema de Bayes, el algoritmo estima la probabilidad de que un nuevo dato pertenezca a alguna clase. Es una alternativa a la regresión logística cuando la variable cualitativa tiene más de dos niveles.

El teorema de Bayes (17), considerando dos eventos A y B, establece la probabilidad de que B ocurra habiendo ocurrido A como la probabilidad de que A y B sucedan simultáneamente entre la probabilidad de A.

$$P(B|A) = \frac{P(AB)}{P(A)} \tag{17}$$

Aplicando el teorema a la probabilidad previa calculada es posible la obtención de la probabilidad a posterior de cada clase con la que posteriormente se definirá la función de discriminación

La ventaja de este método (Rodrigo, 2016), es que presenta mayor estabilidad en los resultados si las clases están bien separadas y tienen una buena predicción cuando el conjunto de datos es reducido y la función de probabilidad de las variables es aproximadamente normal.

El proceso para este análisis consta de varios pasos:

- Dado un set de datos, del que se conocen las clases a las que pertenece cada dato, se calculan las probabilidades previas (prior probabilities).
- Se determinada si la varianza o matriz de covarianzas es homogénea para determinar si se usa el LDA o QDA.
- Estimar parámetros necesarios para las funciones de probabilidad condicional y calcular los resultados con la función discriminante, es decir, la función define a que grupo se le asigna a la nueva entrada.

Por otro lado, el clasificador cuadrático QDA es muy similar al LDA. Sin embargo, el QDA considera que cada clase tiene su propia matriz de covarianza, lo que se transforma en que la función discriminante toma una forma cuadrática.

2.1.1.4 Arboles de decisión

Los árboles de decisión son algoritmos supervisados que dividen el espacio agrupando por valores similares para la variable dependiente. Se utilizan tanto para problemas de clasificación como de regresión. Para ello es importante como se subdivide en regiones más pequeñas hasta fragmentar el espacio en regiones menores que agrupan datos de la misma clase.

Los árboles se estructuran por nodos y se realiza una lectura de arriba abajo (Merayo, 2020). Se diferencian tres tipos de nodos (Ilustración 12):

- Nodo raíz: donde se produce la división inicial en función de las variables más destacadas.
- Nodos internos: Realizan la misma función que los nodos raíz, dividen el conjunto de datos en subconjuntos en función de las variables destacas.
- Nodos terminales u hojas: están situados en la parte baja de la estructura y su función es definir la clasificación definitiva.

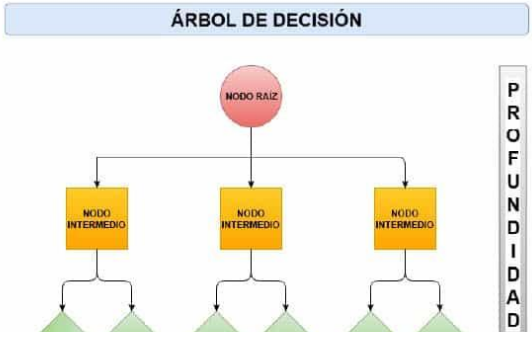


Ilustración 12: Estructura árbol de decisión.

Fuente: (Merayo, 2020).

Para la aplicación de los árboles de decisión en Machine Learning es necesario utilizar el algoritmo de Hunt. Se basa en la división en subconjuntos de forma óptima. El funcionamiento consiste en introducir un set de datos (entrenamiento) en un nodo, si pertenecen a la misma clase ese nodo se define como un nodo terminal (Ilustración 13). Si por el contrario, pertenece a varias clases se dividen los datos en subconjuntos diferentes en función de una variable. Este proceso se repite constantemente, haciendo que cada subconjunto sea más pequeño y finalmente obteniendo todos los nodos terminales del problema.

Para seleccionar que variable es la más destacada se pueden utilizar varios métodos: Error de clasificación, índice de Gini o la entropía. La división del árbol (Orellana Alvear, 2018) sigue un enfoque de división binaria recursiva que analiza la mejor variable para ramificar solo en el proceso de división actual.

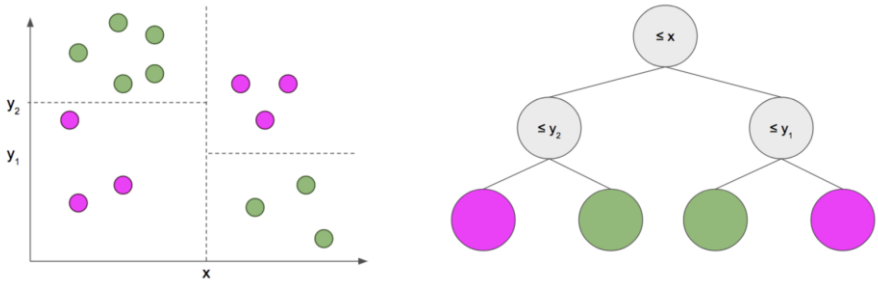


Ilustración 13: Árbol de decisión para Machine Learning

Fuente: (Orellana Alvear, 2018)

2.1.1.5 Redes neuronales

Una neurona conforma el elemento básico del sistema nervioso del ser humano, conteniendo más de 10^{11} neuronas interconectadas entre ellas, lo que conlleva más de 10^{15} conexiones. Las neuronas, funcionalmente, están formadas por tres partes: un cuerpo celular, unido a un axón y a un conjunto de dendritas (Ilustración 15). La forma en la que trabajan las neuronas se compone de una recepción de señales que son captadas por las dendritas. Estas señales proceden de los axones de otras células a través de conexiones sinápticas y a su vez las señales pasan al cuerpo celular combinándose.

Pasado un periodo de tiempo, si la señal combinada excede un valor limite la neurona se activa o lo que es igual, se produce un impulso de salida que transportándose por el axón se propaga por la red. La combinación de esta recepción, tratamiento y propagación de señales conjunta es lo que caracteriza las tareas que son capaces de realizar los seres inteligentes.

Una Red Neuronal Artificial (RNA), o su término en ingles *Neural Network (NN),* es un modelo matemático inspirado en el comportamiento biológico de las neuronas. Consiste en un conjunto de entidades, calificadas como neuronas artificiales, que están interconectadas para la transmisión de datos (Ilustración 14). El objetivo de la red es que los datos de entradas se sometan a diferentes operaciones produciendo unos datos de salida.

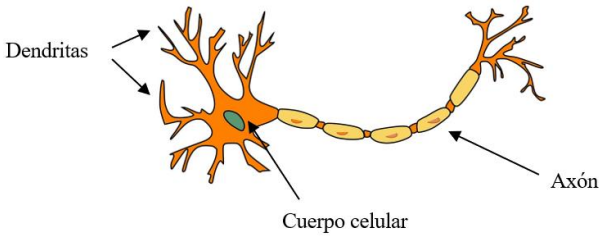


Ilustración 14 Neurona

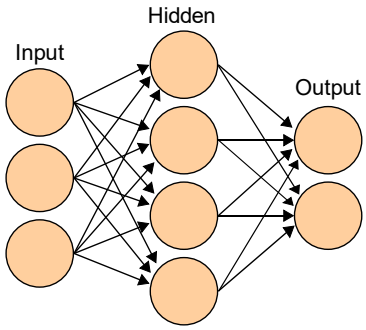


Ilustración 15 Red neuronal

2.1.1.5.1 Elementos básicos. Modelo neuronal de McCulloch-Pitts

El primer modelo de red neuronal artificial fue creado en 1943 para llevar a cabo tareas simples. La red neuronal (Caparrini, 2017) (Ilustración 16):

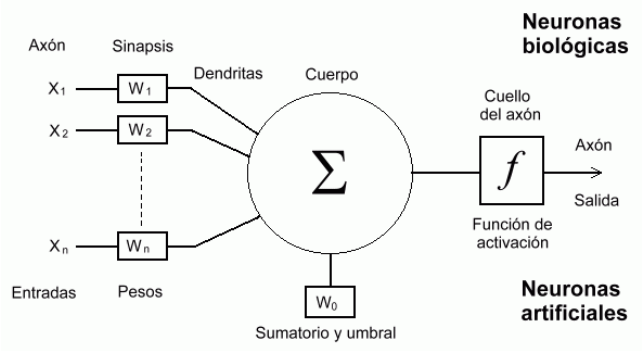


Ilustración 16: Modelo neuronal de McCulloch-Pitts

Fuente: (Caparrini, 2017)

El funcionamiento de esta red neuronal se asemeja al funcionamiento explicado de la neurona biológica y el estímulo de la neurona se produce por las entradas del sistema que modelan el entorno. La salida es la respuesta de la agregación y operación del estímulo. La neurona se adapta al medio y aprende de él mediante la variación de los pesos sinápticos, que se conocen también como parámetros libres, ya que se pueden modificar dependiendo de cada tarea.

La salida de este modelo se define por una neurona de salida denominada Y:

$$Y = f(\sum_{i=1}^{n} w_i x_i) \qquad (17)$$

Donde:

$x_1, \cdots, x_n$: $Conjunto\ de\ entradas$

$w_1, \cdots, w_n$: $Pesos\ sinápticos\ correspondientes\ a\ cada\ entrada$

$\sum_{i=1}^{n}$: $Función\ de\ agregación\ o\ propagación$

f: $Función\ de\ activación$

Y: $Salida$

Un parámetro muy importante en los modelos de redes neuronales es la función de activación (Alba, 2016). La tarea de esta función, resumidamente, es el equivalente biológico al valor limite por el cual la neurona emite un impulso. En este caso, la función de activación define la salida de un nodo (neurona) dada una o varias entradas. Las funciones de activación más comunes son (Ilustración 17):

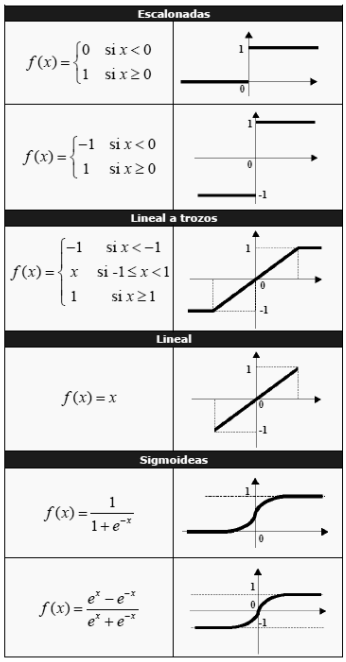


Ilustración 17: Funciones de activación

Fuente: (Alba, 2016)

2.1.1.5.2 El perceptrón simple

El perceptrón es la forma más simple de una red neuronal. Se caracteriza por ser un modelo neuronal de clasificación, donde la salida es una clasificación linealmente separable, es decir, es posible trazar una recta que separe ambas clases.

Además, utiliza el método de error-corrección para adaptar los pesos sinápticos, que consiste en ajustar los pesos de las conexiones sinápticas en función del error cometido a la salida, es decir, de la diferencia entre los valores obtenidos y los valores reales. El algoritmo se desarrolló en un proceso de aprendizaje por Rosenblatt para entrenar el perceptrón del cerebro. Para explicar el algoritmo del perceptrón simple se utilizará un ejemplo (Caparrini, 2017) (Ilustración 18):

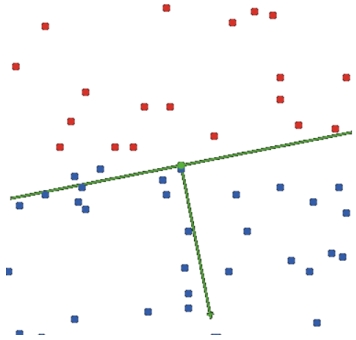


Ilustración 18: Conjunto de datos de ejemplo.

Fuente: (Caparrini, 2017)

Suponiendo que se tienen dos clases, C_1 y C_2, representadas con los colores rojos y azul respectivamente, donde cada elemento se representa en el plano por un punto (x,y). Se supondrá también que estas clases son separables linealmente. Además, se definen unos pesos sinápticos, w_1 y w_2, y el termino independiente de la ecuación de la recta b, con los que la ecuación de la recta se define como:

$$y = -\frac{w_1}{w_2}x - \frac{b}{w_2} \tag{18}$$

Se utiliza como función de activación la función signo definida por:

$$\Gamma(s) = \begin{cases} 1, & \text{si } s \geq 0 \\ -1, & \text{si } s < 0 \end{cases} \tag{19}$$

Y para este ejemplo de clasificación lineal separable, donde la salida neuronal es Y estará dada por:

$$Y = \begin{cases} 1, & \text{si } w_1 x_1 + w_2 x_2 + b \geq 0 \\ -1, & \text{si } w_1 x_1 + w_2 x_2 + b < 0 \end{cases} \tag{20}$$

Esta recta separa ambas clases y si un punto (x_0, y_0) está en el espacio C_1, entonces:

$$w_1 x_0 + w_2 y_0 + b < 0 \tag{21}$$

Si en caso contrario (x_0, y_0) está en el espacio C_2, entonces:

$$w_1 x_0 + w_2 y_0 + b > 0 \tag{22}$$

La neurona clasificara de la siguiente manera:

$$\begin{aligned} (x_0, y_0) \in C_1 \Leftrightarrow Y = -1 \\ (x_0, y_0) \in C_2 \Leftrightarrow Y = 1 \end{aligned} \tag{22}$$

Para poder realizar esta clasificación, anteriormente se ha tenido que realizar un proceso de aprendizaje. Para aplicar este proceso en necesario un conjunto de datos (datos de entrenamiento) D, del cual se conoce tanto la entrada como la salida.

El proceso de aprendizaje consiste en tomar aleatoriamente unos valores iniciales incluidos en D e introducirlos en la neurona. Una vez se ha introducido en la neurona, si la neurona clasifica de forma errónea el punto entonces se aplica una corrección de los pesos sinápticos. Si la neurona clasifica satisfactoriamente, entonces no se realiza ninguna corrección.

Este proceso se repite con todos los puntos del conjunto D, reintroduciendo los mismos datos si es necesario, hasta que la neurona clasifica de forma satisfactoria todos los elementos del conjunto D. Si los conjuntos son separables linealmente, converge en un número finito de pasos.

2.1.1.5.3 El perceptrón multicapa

Las redes neuronales actuales son la agrupación de perceptrones simples o como también se llama, perceptrón multicapa (Ilustración 19). La característica principal de la agrupación de perceptrones simples es que la salida unos se convierte en la entrada de otro perceptrón. Aunque la estructura de ordenación es libre, comúnmente se organiza en capas ordenadas donde las salidas de una capa de neuronas es la entrada de la siguiente capa.

Las redes neuronales constan de tres capas:

- Capa de entrada: configurada por las neuronas encargadas de recibir la información que posteriormente será tratada en la red
- Capa de salida: configurada por las neuronas encargadas de transmitir la información que ya ha sido tratada por la red, es decir, la salida del sistema.
- Capa oculta: configurada por las neuronas agrupadas entre la capa de entrada y la capa de salida. Una red neuronal puede estar formada por varias capas ocultas, al igual que no tener.

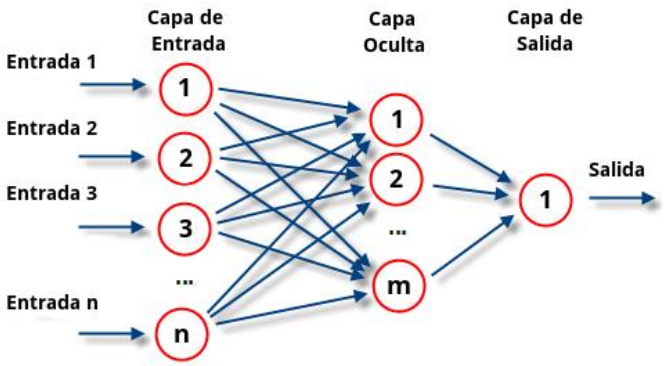


Ilustración 19: Estructura red neuronal

Tras presentar la estructura de las redes neuronales más comunes, se presenta el problema de optimizar los pesos sinápticos de la red. Este problema surge porque son los únicos parámetros libres del sistema y por tanto, el objetivo es encontrar los pesos de la red que minimicen el error entre los datos reales y la salida obtenida.

El procedimiento habitual consta de dividir el conjunto de datos en dos subconjuntos, un subconjunto de entrenamiento y un subconjunto de test. El conjunto de entrenamiento permite medir puntualmente el comportamiento de la red en la búsqueda de los pesos, a la vez que guiar en la dirección en la que se deben dar los siguientes pasos. Posteriormente, para verificar el comportamiento de la red se utiliza el conjunto de test. Está porción de los datos originales obtenida de forma aleatoria del conjunto completo permite calcular el error de los datos obtenidos en datos que la red aún no había tratado.

La independencia de los datos test con los utilizados para entrenar la red es muy importante ya que, si se utilizasen datos que ya se han introducido en la red, el error obtenido podría ser excesivamente optimista debido al sobreajuste en los datos utilizados para entrenar la red

2.1.1.5.4 Análisis PCA

Un análisis PCA, o análisis de componentes principales, es un método utilizado para describir un conjunto de datos en términos de nuevas variables no correlacionadas. La técnica ordena las componentes por su cantidad de varianza original y por ello es una técnica usada para reducir la dimensión de un conjunto de datos.

El análisis PCA determina la proyección según la cual los datos quedan mejor representados en términos de mínimos cuadrados. Para convertir un conjunto de variables observadas con una posible correlación en un conjunto de valores sin correlación lineal, llamadas componentes principales.

Cada componente principal es una combinación lineal de las variables originales y todas las componentes principales son ortogonales entre sí, por lo que no hay información redundante. Además, las componentes principales forman una base ortogonal para todo el espacio de los datos.

Las componentes principales se obtienen por rotación de los ejes en el espacio de los parámetros, definiendo las nuevas variables no correlacionadas. Estas direcciones no tienen por qué tener una interpretación evidente.

Considerando un conjunto de observaciones (Ilustración 20), que forman una nube de puntos en un espacio p-dimensional, donde p son las propiedades o parámetros que definen las observaciones, y teniendo diferentes correlaciones entre las p variables, la distribución de puntos no estará orientada paralelamente a los ejes definidos:

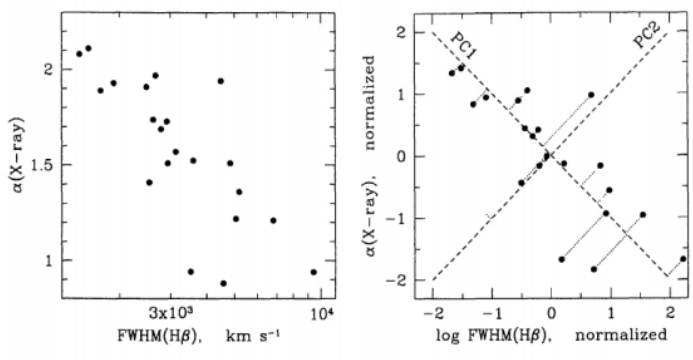


Ilustración 20: Representación de nube de datos en análisis PCA

Fuente: Francis & Wills,1999

A través del análisis PCA se determinan los ejes principales de la nube de puntos y esto se realiza calculando la rotación que minimice la suma de distancias a los ejes, o lo que es lo mismo, que se maximice la proyección de los datos sobre los mismos ejes.

Para llevar a cabo estos cálculos es necesario la compresión de los términos de los autovalores y autovectores. Los autovalores de una matriz son aquellos vectores que, al multiplicarlos por dicha matriz, resultan en el mismo vector o en un múltiplo entero del mismo. Los autovalores son el resultado de multiplicar una matriz por sus autovectores y se obtiene un múltiplo del vector original.

En el análisis PCA se corresponde cada una de las componentes con un autovector y el orden lo determinan los autovalores de forma decreciente. Por lo tanto, la primera componente es el autovector con el mayor autovalor.

Una forma de interpretar el análisis PCA (Amat Rodrigo, 2017) consiste en interpretar las componentes principales desde el punto de vista geométrico. En un conjunto de observaciones con dos variables (X_1, X_2) el vector que define la primera componente principal (Z_1) tiene la dirección donde las observaciones definidas por ambas variables tienen mayor variabilidad (Ilustración 21). El valor de la primera componente principal recoge los valores de las proyecciones de cada observación en dicha dirección.

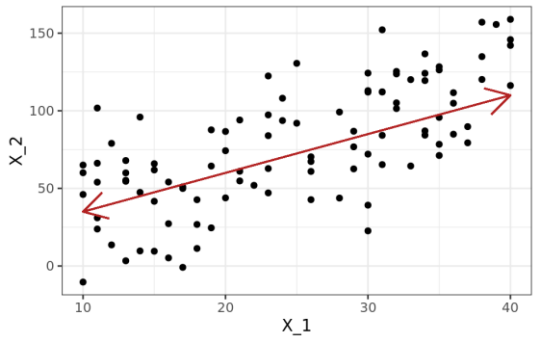


Ilustración 21: Dirección de la primera componente principal

Fuente: (Amat Rodrigo ,2017)

La segunda componente (Z_2) tiene la segunda dirección donde los datos muestran mayor varianza. Además, ambas direcciones principales no están correlacionados, es decir, son ortogonales (Ilustración 22).

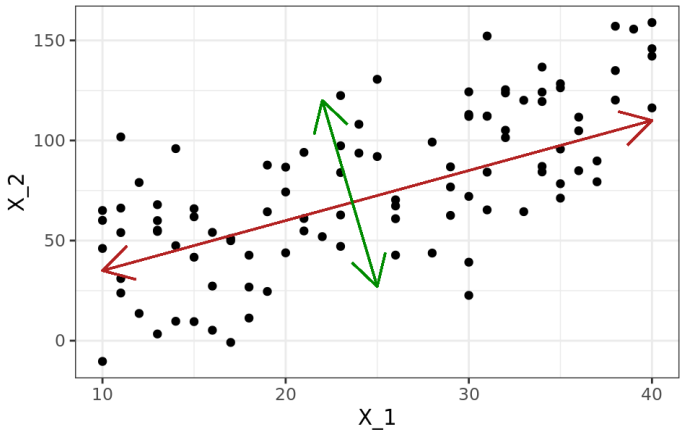


Ilustración 22: Dirección de la segunda componente principal

Fuente: (Amat Rodrigo ,2017)

Las componentes principales se obtienen por combinación lineal de las variables originales. Aunque hay varios métodos para obtener dichas componentes principales, el más intuitivo es el método donde se calculan los autovectores y autovectores de la matriz de covarianza del set de datos.

3 ESTADO DEL ARTE

El número de víctimas de accidentes automovilísticos tiene un gran impacto social y económico. Además, los accidentes donde están involucrados vehículos pesados aumentan la probabilidad de que ocurran daños graves, provocado por su facilidad para volcar respecto al resto de vehículos. Por este motivo, muchos estudios se han centrado en el análisis y estudio de la seguridad para este tipo de accidentes.

Los sistemas de seguridad pueden reducir el número de siniestros, a la vez que se reduce la gravedad de los que no se puedan evitar. En el ámbito de la automoción, más claramente en el ámbito de vehículos pesados, se empezó analizando los sistemas de seguridad pasiva.

Los sistemas de seguridad pasiva tienen la característica de proteger al conductor y ocupantes del vehículo una vez se ha producido el accidente. Estos sistemas forman parte del vehículo como podrían ser barras antivuelco o arcos, estructuras parciales de protección o incluso estructuras completas que protegen al conductor en caso de vuelco.

Tras conseguir la meta de proteger a los ocupantes del vehículo se continuó con el desarrollo de sistemas activos, es decir, sistemas con la finalidad de evitar un accidente. Estos sistemas son capaces de prevenir el siniestro o avisar al conductor del vehículo antes de llegar a la situación de inestabilidad que provoca el vuelco. Dentro de estos sistemas destacan los siguientes:

- **Sistema ABS:** sistema antibloqueo de frenos, en ingles Anti-Lock Braking System, es un dispositivo que permite la variación de forma dinámica en la fuerza de frenado. Aunque el uso principal del ABS, y el más conocido, es evitar situaciones de perdida de adherencia en situaciones longitudinales también es muy importante en las situaciones de perdida de adherencia lateral o asimetrías en cuanto a la adherencia.

- **Sistema ESP:** control de estabilidad, en ingles electronic stability progam, es un sistema de control de estabilidad que utilizando 4 sensores (sensor de ángulo de volante, sensor de giro de rueda, sensores de aceleración lateral y un giroscopio) y la actuación conjunta del ABS evita que se pierda el control del vehículo, incluso es capaz de modificar la trayectoria.

  Mediante el sensor de ángulo de volante el sistema interpreta la dirección que se quiere seguir y con qué velocidad se quiere modificar dicha trayectoria. Posteriormente, el sensor de giro de las ruedas junto con el sensor de aceleración lateral indica si el vehículo está describiendo la trayectoria que el conductor ha indicado con el volante.

  El sistema entra en acción cuando la variación entre la dirección teórica y la real es mayor a la permitida o cuando el giroscopio detecta una variación excesiva. Lo hace frenando la rueda que más convenga, para generar una fuerza opuesta a la que está creando la desviación. De esta forma, se compensan las fuerzas y se vuelve a la situación de estabilidad (Ilustración 23).

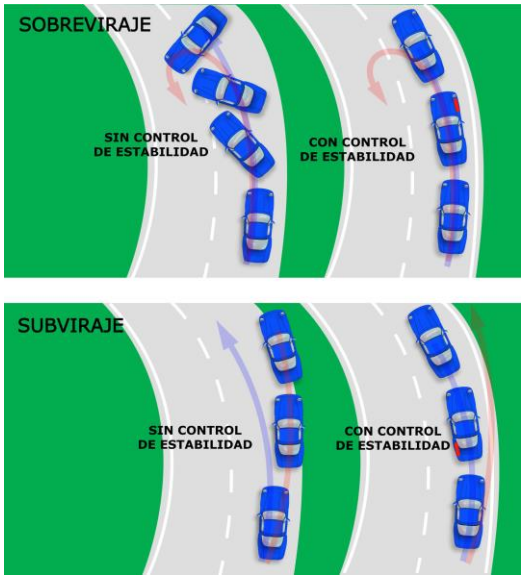


Ilustración 23 : Sistema ESP

Fuente: (Adautomotor, 2018)

- **Sistema ERM:** control electrónico de inclinación, en inglés Electronic Roll Mitigation, es un sistema que detecta las situaciones de perdida de estabilidad lateral en vehículos. Este sistema agrupa dos tipos de actuaciones: por un lado, es la actuación directa sobre el sistema de frenado, el pedal del acelerador. Por otro lado, es la acción de prevenir al conductor de la situación de inestabilidad mediante una señal sonora o visual.

  Se trata del mismo sistema del ESP pero esta vez orientado a la estabilidad lateral. El sistema trabaja por mantener el control y el equilibrio del vehículo con los mismos sensores que el ESP, detectando en este caso la posibilidad de vuelco.

  Cuando el sistema, comparando la variación del giro del volante y la velocidad del vehículo, determina que puede ser la causa de la elevación de una rueda se activa (Ilustración 24). Al activarse el sistema se acciona, mediante el uso del ABS, el freno de la rueda correspondiente y disminuye la potencia del motor provocando la disminución del riesgo de vuelco.

Ilustración 24: Sistema ERM

Sin embargo, estos sistemas tienen algunos inconvenientes. La acción del vuelco se mide solamente mediante el giro del volante y velocidades longitudinales pudiendo provocar situaciones donde, habiendo un peligro de vuelco existente no se detecte. Esto se produce porque dichos parámetros se han introducido previamente bajo consideraciones de velocidades longitudinales para determinados radios de curvatura. Por lo tanto, se excluyen situaciones donde por ejemplo varia la adherencia, la carga o el estado de la vía por donde circula el vehículo.

Debido a esta problemática, los sistemas de seguridad activa se han seguido desarrollando. En el caso del vuelco e inestabilidades en los vehículos se ha empezado a introducir el tratamiento de datos masivos para la determinación de parámetros en tiempo real que mejoren la predicción de situaciones adversas o incluso algoritmos que detectan la perdida de estabilidad clasificando en diferentes estados según algunos parámetros de entradas.

3.1.1 Detección de sobreviraje en automóviles con Machine Learning

Para ser más concretos, en el caso de BMW (Freudling, 2018) se estudia la situación de inestabilidad del sobreviraje. El sobreviraje es una condición que se produce cuando los neumáticos traseros del vehículo pierden su adherencia. El eje trasero "trata de adelantar" al delantero provocando una situación que genera que el vehículo gire en exceso. La deriva del eje trasero se produce en mayor medida que la del eje delantero y, por lo tanto, si estamos tomando una curva la trayectoria tiende a cerrarse.

El sobreviraje se puede producir debido a diversas causas como pueden ser: neumáticos con poca adherencia, condiciones de deslizamiento de la carretera, giros demasiados rápidos, frenadas bruscas cuando se realiza un giro o una combinación de todos estos factores.

Los sistemas de control de estabilidad actuales se diseñan para actuar automáticamente cuando se producen situaciones de inestabilidad. Sin embargo, este enfoque es difícilmente aplicable debido a la interacción de muchos factores que afectan a la dinámica del vehículo.

Por esta difícil implementación, la vía desarrollada por BMW consiste en modificar la predicción de valores limites que provoquen una actuación en el vehículo, por un análisis de Machine Learning para detectar la situación de sobreviraje. Este enfoque se centra en un prototipo ECU basado en aprendizaje automático supervisado, más concretamente en un modelo de clasificación.

En primer lugar, se realiza una recopilación de datos masiva y reales, es decir, datos obtenidos de pruebas reales sin utilizar simulaciones. Con la colaboración de un piloto profesional, se realizan pruebas de conducción donde se capturan las señales de aceleración longitudinal, aceleración lateral, ángulo de dirección y velocidad de guiñado del vehículo.

Para determinar en qué momento se produce el sobreviraje, se usa un procedimiento manual donde el piloto avisa al copiloto, que registra instantáneamente cuando el piloto indica que el vehículo está sobrevirando. El copiloto acciona un botón para indicar cuando se inicia el sobreviraje y se mantiene hasta que el piloto indica que el estado de inestabilidad se ha revertido. Al estar realizando el proceso de esta manera, lo que se realiza es un proceso de etiquetado de datos. Los datos obtenidos de los sensores en todo momento se registran sabiendo si corresponden a valores que provocan una situación de sobreviraje o no.

Una vez se han recopilado los datos, es necesario tratarlos para poder realizar la labor de aprendizaje del algoritmo. En primer lugar, los datos obtenidos de ensayos se filtran para eliminar el ruido producido en la toma de datos y posteriormente se caracterizan las curvas de las señales de entradas utilizando un análisis de identificación de picos (Ilustración 25). Este paso, aunque tedioso, es muy importante porque debido a la limpieza de datos, pasándolos incluso por filtros, se mejoran los resultados desde un 75-80% a un 95%.

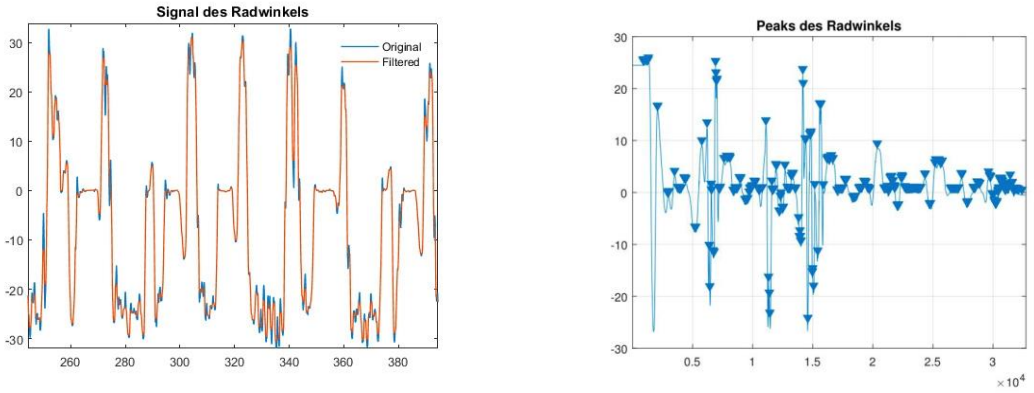


Ilustración 25: Tratamiento de datos, filtrado y caracterización de picos.

Fuente: (Freudling, 2018)

Después de filtrar y tratar los datos, se continua con el proceso de entrenamiento del modelo de Machine Learning. En este caso, se utilizan diferentes enfoques para la clasificación. Se utiliza el clasificador de K vecinos cercanos (KNN), el clasificador de vectores de soporte (SVM), análisis discriminatorio cuadrático y arboles de decisión. El resultado expresado en la tabla 2, muestra como todos los modelos de Machine Learning de clasificación funcionan con una tasa de verdaderos positivos mayor al 98% (Exceptuando el modelo que utiliza el análisis de componentes principales para la reducción de parámetros)

	Verdadero positivo (%)	Verdadero negativo (%)	Falso positivo (%)	Falso negativo (%)
K-Vecino más cercano con PCA	94,74	90,35	5.26	9,65
Máquinas de vectores soporte	98,92	73.07	1.08	26,93
Análisis discriminante cuadrático	98,83	82,73	1,17	17.27
Árboles de decisión	98,16	95,86	1,84	4.14

Tabla 2: Resultados de los diferentes algoritmos utilizados por BMW

Estas tasas de acierto se verifican mediante las pruebas con otro vehículo con el que se obtiene un 95% de tasa de cierto. Por lo que se puede concluir que es viable la clasificación de estados de inestabilidad, en este caso el sobreviraje, mediante Machine Learning. El aprendizaje automático brinda la oportunidad de desarrollar software que sea capaz de utilizar los datos disponibles para conocer el comportamiento del vehículo y mejorar así la seguridad en los vehículos de forma activa.

3.1.2 Estimación del ángulo del balanceo mediante redes neuronales.

Debido a la dificultad de conocer la dinámica del coche y obtener valores certeros de algunos de sus parámetros surge este estudio. El problema principal es la poca capacidad para obtener directamente valores dinámicos en un vehículo, como es el caso del ángulo de balanceo.

En el diseño de sistemas ERM o RSC es importante conocer la dinámica del vehículo. El ángulo de balanceo es uno de los parámetros más importante y a la vez más difícil de medir directamente, por lo que es necesario su estimación mediante integración. Además, para la implementación de un sistema ERM se deben cumplir algunos requisitos compartidos con otros sistemas del vehículo:

- Adquirir información de sensores con alta tasa de refresco
- Procesar información del sensor en tiempo real
- Incluir actuadores con respuesta rápida

Para solventar este problema (García Guzmán *et al.*, 2018) se opta por un desarrollo de arquitectura IoT(Internet of Things), integrando una red neuronal con sensores de bajo costo.

Para abordar esta problemática se han realizado estudios que analizan la capacidad de los sensores, de mayor o menor calidad, para la obtención de datos en tiempo real y de manera confiable (Tafner, Reichhartinger and Horn, 2014) y (Vargas-Meléndez *et al.*, 2016).

Por otro lado, se conoce la capacidad de las redes neuronales como mecanismo para la estimación de variables dinámicas no conocidas. En (Mangeas, Glaser and Dolcemascolo, 2002) se usan para la estimación de pesos estáticos o en (Gajdar, Rudas and Suda, 1997) para predecir el coeficiente de fricción de las ruedas de un tren.

La arquitectura basada en IoT, se ha desarrollado uniendo una unidad de movimiento inercial (IMU) y una pequeña computadora que toma los datos de la IMU para estimar el ángulo de balanceo mediante una red neuronal. El resultado de las estimaciones, posteriormente, se compara con las mediciones obtenidas por un equipo profesional capaz de estimar el ángulo de balanceo (VBOX de Racelogi).

El diseño de este prototipo consiste en dos partes, o como se define en el artículo, hardware y software. La parte de hardware está compuesta por dos pequeñas computadoras, Raspaberry Pi 3 y un Intel Edison System-on-Chip, además de la ya mencionada VBOX de Racelogi.

La parte de software está compuesta básicamente por la capa de percepción, es decir, los elementos que recogen la información necesaria para ser capaz de monitorear las situaciones de riesgo de vuelco. En segundo lugar, las conexiones entre los sensores y el vehículo. Por último, el conjunto de aplicaciones y filtros capaces de mejorar el tratamiento y percepción de las señales, donde se incluye la red neural capaz de estimar el ángulo de balanceo.

Este sistema para estimar el ángulo de balanceo, tiene como peculiaridad que no necesita características físicas detalladas del vehículo para obtener un sistema preciso. La arquitectura de la red neuronal que se propone utiliza tres capas: una capa de entrada, una capa oculta con 15 neuronas y una capa de salida. La salida de la red para estimar el ángulo de balanceo, φ_e, se define como:

$$\varphi_e = g_2\left(\sum_{k=1}^{15}(v_k o_k) + b_2\right) \qquad o_k = g_1\left(\sum_{l=1}^{4}(w_{lk} i_l) + b_{1l}\right) \qquad (23)\ (24)$$

Donde v_k son los pesos de la capa oculta, b_2 es la constante en la capa de salida y g_2 la función de activación. O_k se define como la salida de la K-ésima neurona de la capa oculta. W_{lk} son los pesos de la red de entrada, b_{1l} es la constante en la capa oculta, g_1 es la función de activación y i_l son las entradas de la red neuronal.

Para el entrenamiento de la red se obtienen los datos mediante un modelo de vehiculo TruckSim. Simulando diferentes maniobras (cambio de carril doble cambio de carril y giro en J) a diferentes velocidades para la caracterización no lineal del vehículo. Las entradas de la red neuronal son la aceleración lateral a_{ym}, aceleración longitudinal a_x, velocidad de guiñado $\dot{\psi}$ y velocidad de balanceo $\dot{\varphi}$ (Ilustración 26).

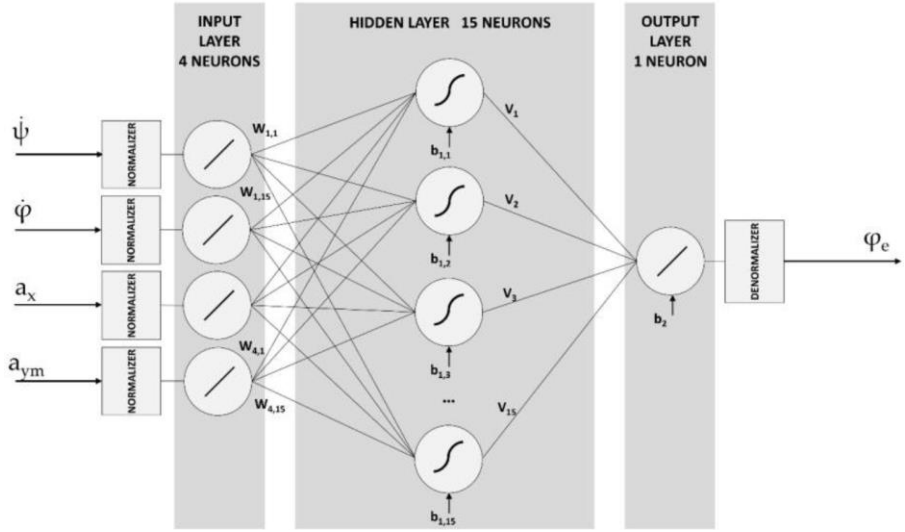


Ilustración 26: Arquitectura red neuronal

Fuente: (García Guzmán *et al.*, 2018)

La ventaja de utilizar redes neuronales, en contra posición de los métodos derivativos, es que el ángulo de balanceo del vehículo se estima en cada conjunto de datos. Al utilizar la información obtenida directamente de lo sensores, sin integrar la señal, no se produce el fenómeno del error acumulado.

Los resultados de las pruebas realizadas se comparan mediante tres pruebas. En la primera prueba se analiza la respuesta del sistema ante la entrada de un giro en J, lo que corresponde a realizar una rotonda. La maniobra se realiza a una velocidad constante de 40km/h y el radio de la rotonda es de 22 m. Las estimaciones de la red en la primera prueba, comparándolo con el sistema que se considera como referencia (puntos rojos) (Ilustración 27), se considera una buena estimación en los dos dispositivos con los que se predice el ángulo de balanceo y, por tanto, se ratifica el buen resultado ofrecido por las redes neuronales.

Mas concretamente, se ratifica el buen funcionamiento de la red neuronal comparando el error cuadrático medio de los dos sistemas de computación a estudiar y el sistema VBOX respecto a una toma de datos real realizada mediante antenas GPS a ambos lados del vehículo.

	Angulo de inclinación	
	Error RMS(º)	Error máximo(º)
Raspberry Pi 3 Modelo B	0,7405 ± 0,0823	3,54
Intel Edison	0,7965 ± 0,0743	3,84
VBOX IMU Racelogic	0,5792 ± 0,0322	2,74

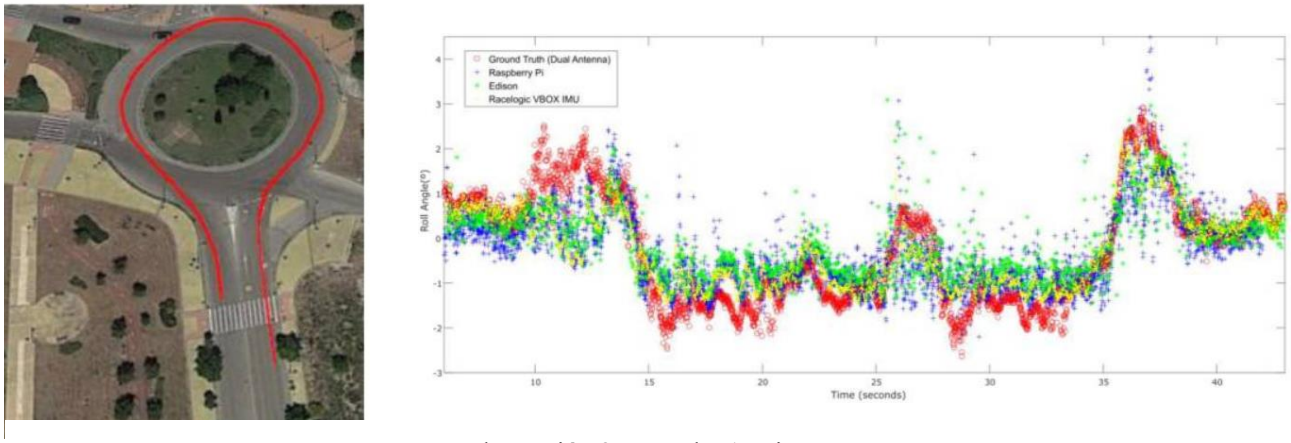


Ilustración 27: Prueba 1. Giro J

En la segunda prueba (Ilustración 28) se realiza una maniobra de cambio de carril doble que consiste en un slalom a velocidad constante. En este caso el resultado también muestras una gran similitud, entre la realidad y los dispositivos que utilizan redes neuronales.

	Angulo de inclinación	
	Error RMS(º)	Error máximo(º)
Raspberry Pi 3 Modelo B	0,5302 ± 0,0681	2,54
Intel Edison	0,5075 ± 0,0432	2,36
VBOX IMU Racelogic	0,4521 ± 0,0215	1,95

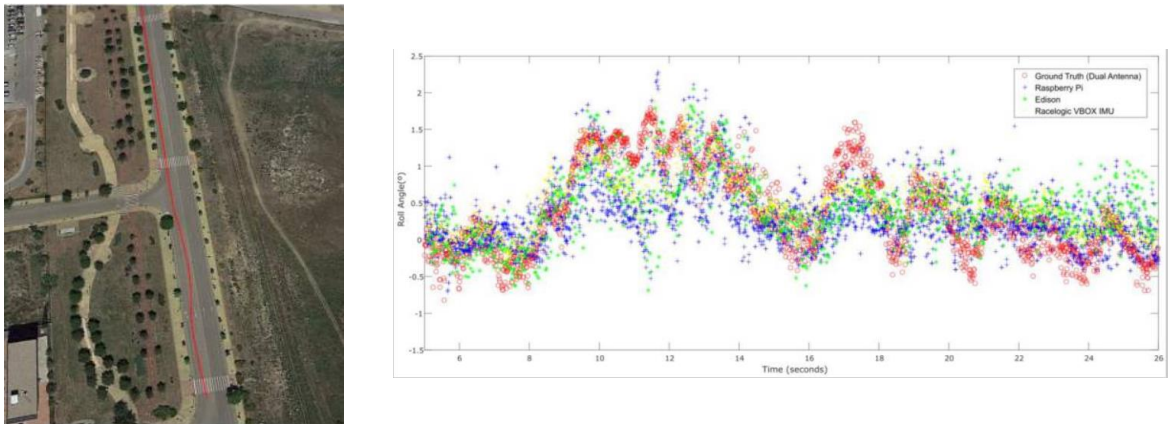


Ilustración 28: Prueba 2. Cambio de carril doble

Por último, la tercera prueba (Ilustración 29) consiste en una circulación general. Se realizan maniobras de giro en J y cambio de carril además de completar un recorrido en condiciones de circulación normal. En estas situaciones el vehículo varió su velocidad entre 20 y 60 km/h. En este caso el error cuadrático medio y los errores máximos aumentan.

	Angulo de inclinación	
	Error RMS(º)	Error máximo(º)
Raspberry Pi 3 Modelo B	1.0321	5,92
Intel Edison	1.3297	4.41
VBOX IMU Racelogic	0.9431	5.29

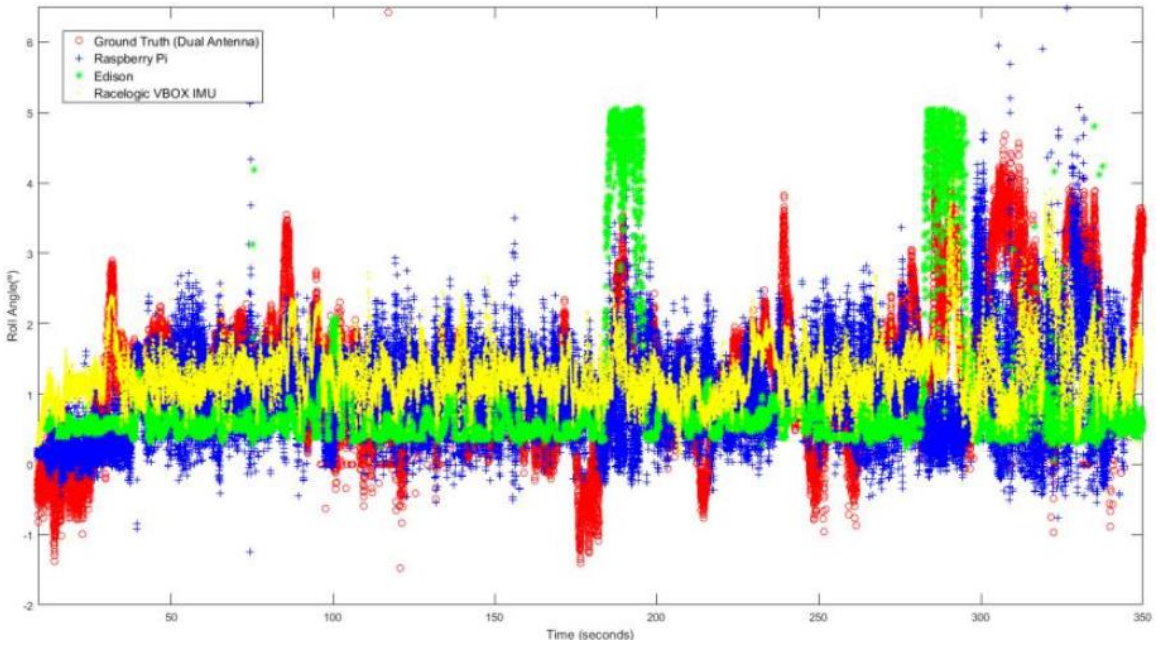


Ilustración 29: Prueba 2. Circulación general

Los resultados muestran una estimación del ángulo de balanceo mediante sensores económicos casi idéntica que la que se mide directamente con un sistema considerado como referencia (Racelogic VBOX). Uno de los inconvenientes del uso de sensores económicos es la introducción de ruidos en algunas franjas de los datos, más concretamente a baja velocidad y con movimientos suaves.

Se puede concluir que los kits que incluyen estimaciones con redes neuronales integradas proporcionan una estimación del ángulo de balanceo muy cercana a los valores reales, capaces de estimar dicho ángulo en situaciones de riesgo de vuelco que cumplen con las restricciones en tiempo real.

Este estudio es un ejemplo de la capacidad de las redes neuronales para estimar valores que en otros casos serían difíciles de obtener, además de tener un intervalo de confianza demasiado grande. Con estos resultados se puede afirmar que con la introducción de sensores algo más complejos, combinado con redes neuronales y filtros se podría obtener un sistema aún más fiable, incluso incluyendo lecturas de la carretera donde se podría introducir el estudio en el aprendizaje profundo o Deep learning.

4 IMPLEMENTACIÓN DE MACHINE LEARNING

Este proyecto se basa en la mejora de la predicción del vuelco de vehículos utilizando las metodologías que del Machine Learning. Para ello, se llevan a cabo diferentes simulaciones de distintos vehículos. Posteriormente, se utilizarán las herramientas del Machine Learning para el tratamiento de los datos obtenidos de las simulaciones y la obtención de los resultados.

A diferencia del proyecto anterior (Díaz Díaz, 2018), no se estudiará la física del vuelco, sino la mejora en la predicción de indicadores para la predicción y medición de la tendencia de un vehículo a volcar. Principalmente, se centra en la predicción del índice de vuelco basado en la transferencia de carga lateral (LLT), que se demostró que es el indicador más fiable para determinar cuando un vehículo está en peligro de vuelco.

El objetivo del proyecto se resume en la búsqueda de una predicción de un valor numérico, por lo que este tipo de problemas se engloba dentro de los problemas de Machine Learning de regresión. Además, mediante la simulación de distintos escenarios se obtendrán los datos necesarios para obtener el resultado del índice de vuelco LLT previamente, es decir, se trata de un problema de aprendizaje supervisado. Se conoce previamente el resultado asociado a los elementos que se van a tratar como datos de entrada y se obtendrá una función que establece la relación entre los datos de entrada y de salida.

Como en el proyecto (Díaz Díaz, 2018) se utiliza el software ADAMS CAR® para la simulación de diferentes escenarios (cambio de carril, rampas, eslalon, maniobra Fish-hook, etc…) con diferente tipología de vehículos (autobús, deportivo, todoterreno). Además, este software permite la obtención de los datos necesario para el posterior tratamiento.

Para el tratamiento de datos, limpieza e implementación de modelos de Machine Learning se ha utilizado el software MATLAB®. Este software es una herramienta muy útil dado que permite leer los datos obtenidos en las simulaciones, transformación y limpieza de datos para posteriormente ser introducidos, mediante el propio módulo de Matlab de Machine Learning o mediante programación, en algoritmos de entrenamiento para la obtención de resultados.

La metodología que se ha seguido en el trabajo se basa en la comparación de diferentes posibilidades de implementación de la metodología de Machine Learning. En primer lugar, se compararán los algoritmos tradicionales como son los árboles de decisión o SVM con las redes neuronales. Tras esta comparación se continua con un proceso de verificación de los resultados más exhaustivo en la que se intentará, utilizando como apoyo (García Guzmán et al., 2018), hallar las variables que mejor explican la física del vuelco con Machine Learning. Además, una vez realizado este estudio, se podrá comparar los datos y variables obtenidas con los resultados de otros estudios.

Por último, se trata la generalización este problema para cualquier tipo de vehículo en función de las variables que desempeñan un papel más importante en el vuelco de vehículos.

4.1 Comparación de algoritmos tradicionales de Machine Learning y redes neuronales.

En este apartado se recogen los pasos seguidos en la aplicación de algoritmos y modelos de Machine Learning al problema del vuelco de vehículos. Se compara, en primer lugar, los resultados que ofrecen los algoritmos que se incluyen dentro del Machine Learning, como pueden ser SVM, K-NN, con los resultados que ofrecen las redes neuronales. Se estudiarán en diferentes situaciones con un mismo vehículo de prueba: un vehículo deportivo que ofrece el software de simulación(Ilustración 30).

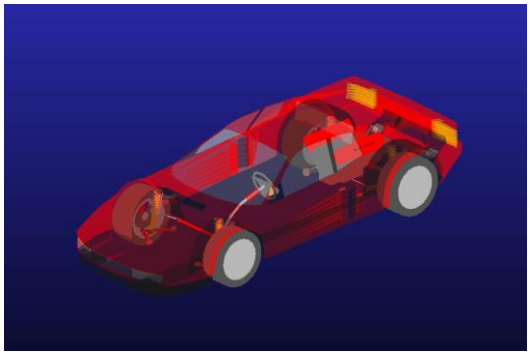


Ilustración 30: Vehículo deportivo de ensayo

Se compara el resultado ofrecido por las variables utilizadas en (García Guzmán et al., 2018), aceleración lateral, aceleración longitudinal, velocidad de guiñado y velocidad de balanceo, con todas las variables que se han obtenido de la simulación, que son 15. También se estudia el funcionamiento del análisis estadístico PCA para la disminución de variables, que se detallará más adelante.

4.1.1 Procedimiento y programación en Matlab

El procedimiento seguido para las cuatro pruebas realizadas es similar. A continuación se explica de manera general los pasos seguido y la obtención de resultados

- **Simulación de maniobra en ADAMS CAR®:** mediante el software se simulará una misma maniobra variando algunos parámetros, como pueden ser la velocidad, ángulo de giro, ángulo de giro máximo, etc. Con estas variaciones se busca la variación de la trayectoria provocando una diferencia entre las simulaciones. En el ejemplo (Ilustraciones 31 y 32), mediante la variación del ángulo de giro máximo, se consiguen dos trayectorias diferentes, realizando la misma maniobra que en este caso es un cambio de carril. Además de estas simulaciones, se realiza una simulación que será la que se utilizara para el test del programa mientras que las anteriores se utilizaran para su entrenamiento. La diferencia entre ambas simulaciones es que en la del test se realizará más de una modificación, además del giro máximo se modificará la velocidad a la que se realiza la maniobra.

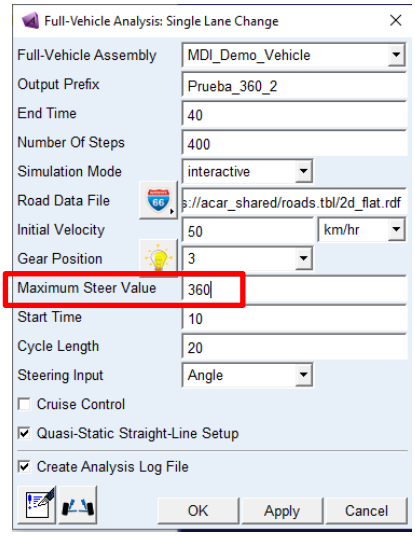


Ilustración 31: Cuadro de configuración de ejemplo

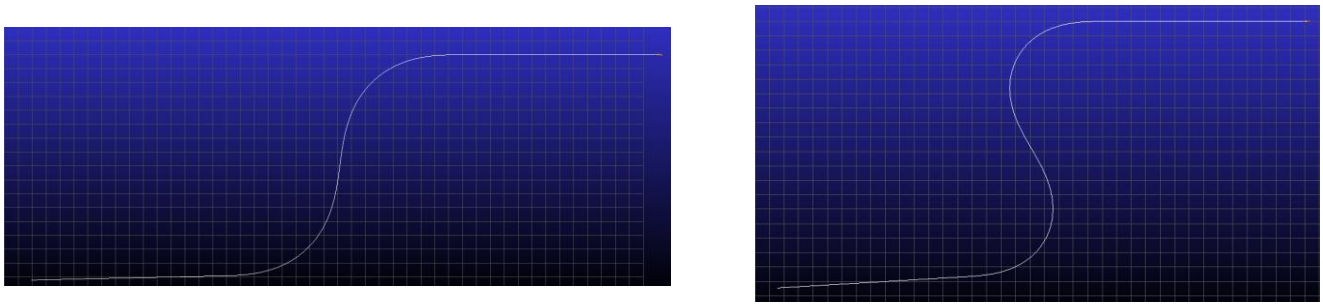


Ilustración 32: Maniobra Line Change con valores de giro máximo de 60 y 90 respectivamente.

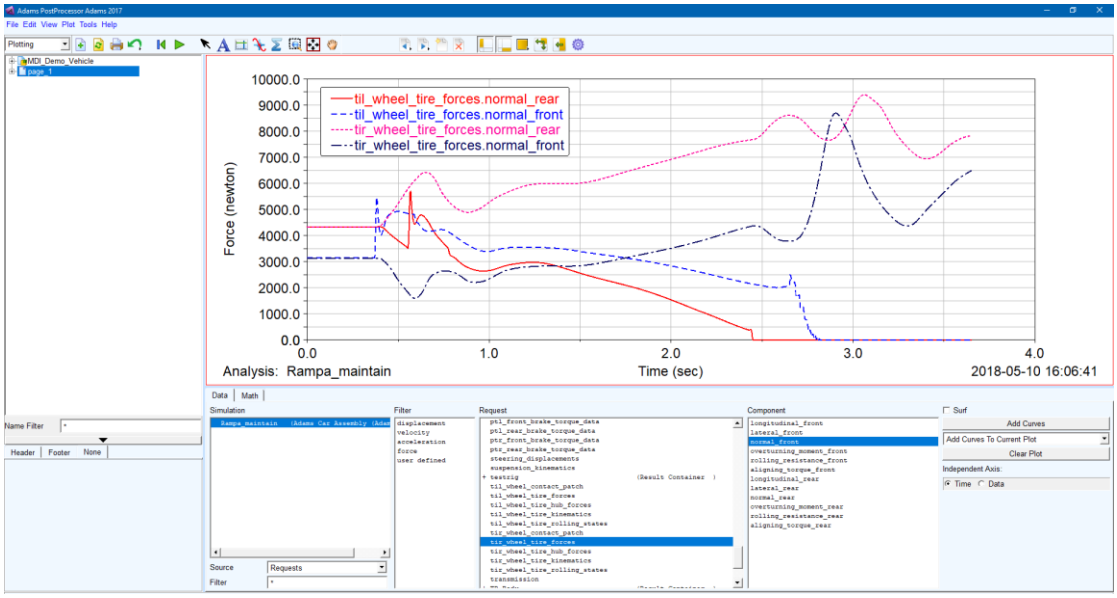


Ilustración 33: Ventana de postprocesado

- **Obtención de datos en ADAMS CAR®:** Se utiliza el módulo de postprocesado (Ilustración 33) del mismo software para la obtención de los datos. Se genera un archivo .txt donde se recogen las siguientes variables:

  - Aceleración longitudinal
  - Aceleración lateral
  - Aceleración vertical
  - Aceleración de balanceo
  - Aceleración de guiñado
  - Desplazamiento longitudinal
  - Desplazamiento lateral
  - Desplazamiento vertical
  - Desplazamiento de balanceo
  - Desplazamiento de guiñado

- Velocidad longitudinal
- Velocidad lateral
- Velocidad vertical
- Velocidad de balanceo
- Velocidad de guiñado
- Fuerza normal en el neumático izquierdo frontal
- Fuerza normal en el neumático izquierdo trasero
- Fuerza normal en el neumático derecho frontal
- Fuerza normal en el neumático derecho trasero

- **Análisis e implementación de Machine Learning en MATLAB®:** el tercer paso engloba la lectura, preprocesado de datos, tratamiento de datos y la implementación del algoritmo. Se leen los archivos .txt obtenidos en la simulación, generando dos bases de datos. Una será la que se utilice para el entrenamiento y otra para el test con los datos recogidos respectivamente. Se obtiene la transferencia de carga lateral (LLT) tanto en los datos de entrenamiento como en los del test, para poder realizar un aprendizaje supervisado.

  A continuación, se realizan diferentes modelos donde se compara la diferencia entre los algoritmos del Machine Learning, los tradicionales (SVM, K-NN, etc…) y las redes neuronales. A su vez, se realizan diferentes modelos en función de sus variables: con un número de variables determinadas o todo el conjunto de variables, con y sin análisis PCA:

  - Modelo Machine Learning para 4 entradas (aceleración lateral, longitudinal y velocidad de guiñado y de balanceo).
  - Modelo Machine Learning para 4 entradas con PCA.
  - Modelo Machine Learning para todos los datos.
  - Modelo Machine Learning para todos los datos con PCA.
  - Modelo con redes neuronales para 4 entradas.
  - Modelo con redes neuronales para 4 entradas con PCA.
  - Modelo con redes neuronales para todos los datos.
  - Modelo con redes neuronales para todos los datos con PCA

  Los diferentes modelos se componen de una misma estructura (ilustración 34) donde:

  1. Conversión de las bases de datos en matrices para poder trabajar mejor con ellas
  2. Entrenamiento del algoritmo con los datos que se utilizan para el entrenamiento
  3. Predicción de los resultados mediante el modelo creado con los datos para el test.
  4. Representación de resultados.

Red Neuronal 4 Entradas

```
1  dataTrain_NN_4=dataTrain{:,[2:3 15 16]};
   dataTest_NN_4=dataTest{:,[2:3 15 16]};

2  NN_4=fitnet(15);
   NN_4.divideParam.trainRatio=70/100;
   NN_4.divideParam.valRatio=15/100;
   NN_4.divideParam.testRatio=15/100;
   [NN_4,~]=train(NN_4,dataTrain_NN_4',LLT_Del');

3  LLT_Del_PRED_NN_4=NN_4(dataTest_NN_4');

4  plot(dataTest.Time,LLT_Del_Test)
   hold on
   plot(dataTest.Time,LLT_Del_PRED_NN_4)
   title('LLT NN 4 ENTRADAS')
   legend("LLT REAL","LLT PRED NN 4")
   hold off
   xlabel('Tiempo')
   ylabel('LLT')


   plot(dataTest.Time,LLT_Del_Test-LLT_Del_PRED_NN_4')
   title('NN LLT')
   xlabel('LLT REAL')
   ylabel('LLT NN ')
```

Ilustración 34: Ejemplo de programación

En el caso de algoritmos clásicos de Machine Learning, el software de MATLAB® permite hacerlo de una forma interactiva para entrenar varios tipos de algoritmos y así poder elegir el que mejor se ajuste al problema. Se utiliza dentro del módulo de Machine Learning la aplicación "Regression Learnner" (Ilustración 35) siguiendo los siguientes pasos:

1. Se inicia introduciendo lo que será la base de datos para entrenar, el resultado esperado, las columnas de datos que se utilizaran como predictores, que en este caso son todas, y por último se selecciona un tipo de validación del modelo. Se utiliza la validación "Holdout validation" que utiliza un porcentaje de los datos para realizar la validación.

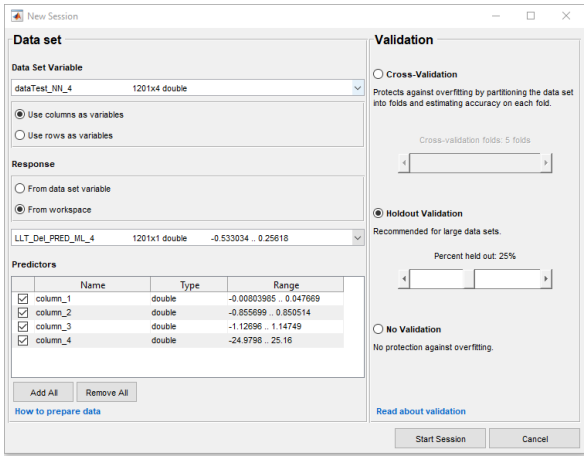


Ilustración 35: Ventana inicial "Regression Learnner"

2. Se entrenan los diferentes modelos y se obtienen los errores de la validación. En consecuencia, se obtiene el que mejor se ajusta a la realidad. Además, esta aplicación permite de forma automática introducir en el entrenamiento un proceso previo PCA para la reducción de la dimensión del problema, y por lo tanto obtener los mismos resultados con menos variables (Ilustración 36).

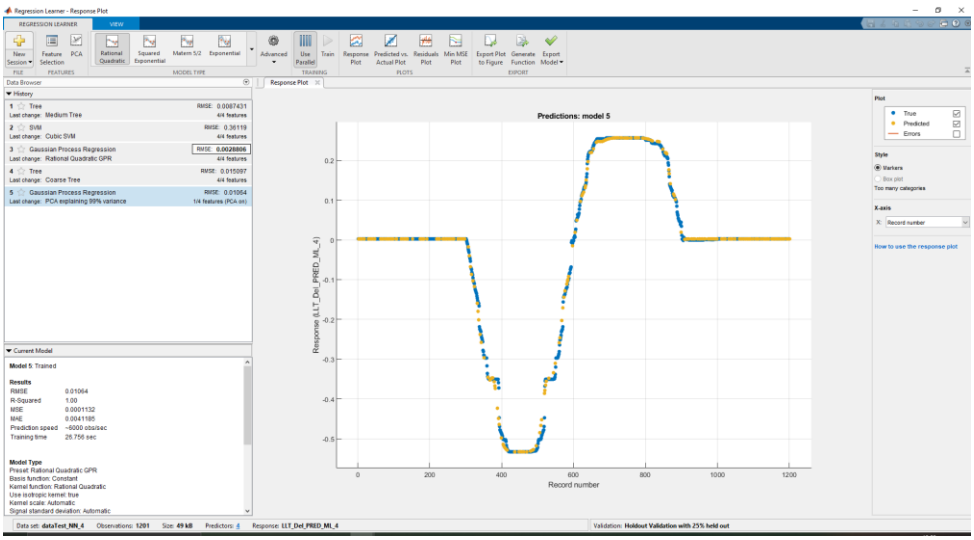


Ilustración 36: Ventana de entrenamiento "Regression Learnner

3. Por último, los modelos creados se pueden exportar mediante código al archivo .m para poder crear una secuencia continua y obtener los resultados conjuntamente.

4.1.2 Resultados obtenidos de la comparación de Machine Learning con redes neuronales.

A continuación, se presentan los resultados obtenidos de la comparación en diferentes situaciones para estudiar el comportamiento y funcionamiento de las dos formas diferentes de tratar el problema, los algoritmos tradicionales (SVM, K-NN,…) y redes neuronales.

4.1.2.1 Prueba con cambio de carril

En el primer caso, se ha optado por una simulación del vehículo completo "Full-vehicle Analysis: single Lane Change". En esta maniobra el vehículo realiza un cambio de carril con diferentes ángulos de giro (Maximum Steer Value) para completar la acción.

Se ha simulado la misma maniobra con diferentes "Maximum Steer Value" variando desde 30 a 360 el valor.

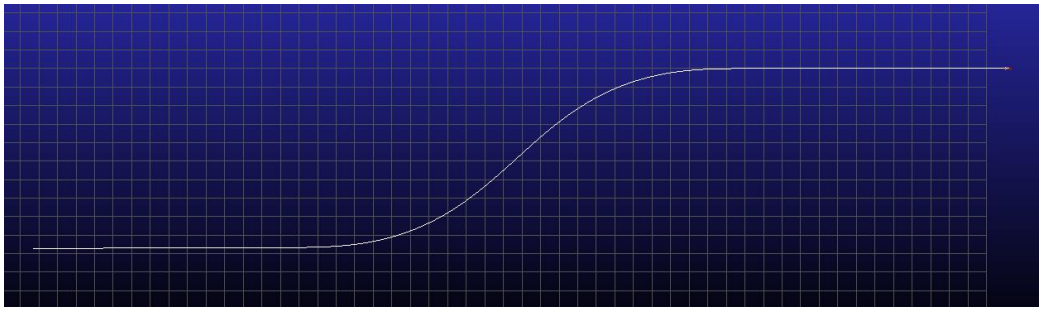


Ilustración 37: Maniobra Line change a 50 Km/h y Max steer value 30

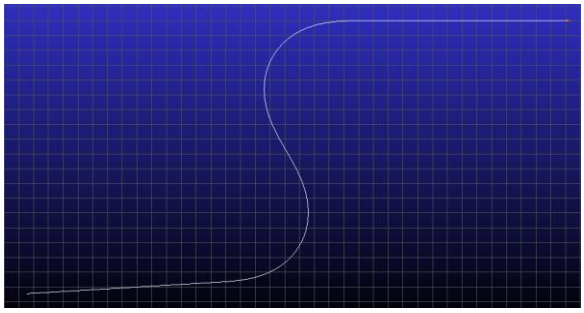


Ilustración 38: Maniobra Line change a 50 Km/h y Max steer value 90

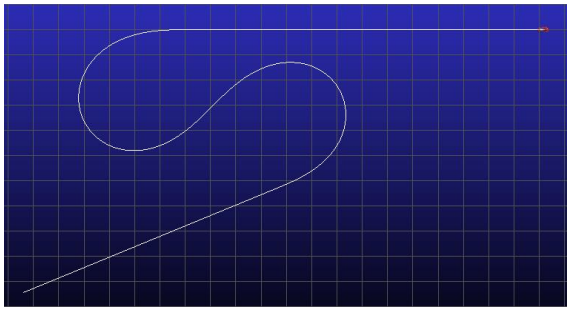


Ilustración 40: Maniobra Line change a 50 Km/h y Max steer value 180

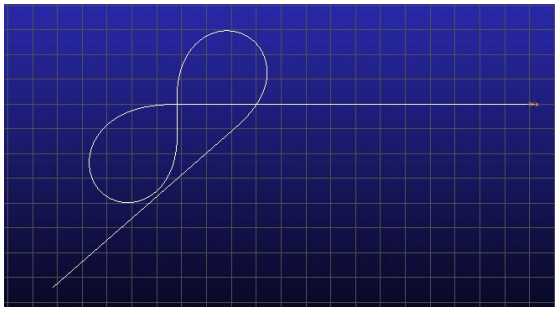


Ilustración 39: Maniobra Line change a 50 Km/h y Max steer value 225

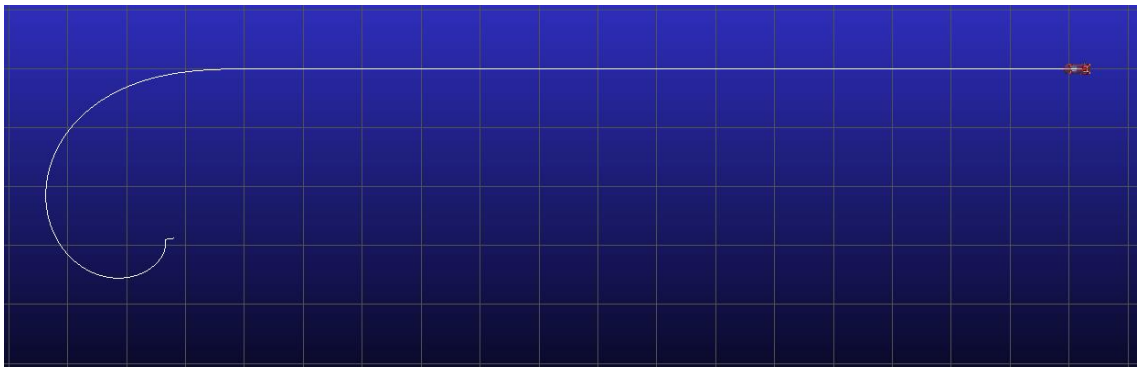


Ilustración 41: Maniobra Line change a 50 Km/h y Max steer value 270

En las ilustraciones 37-40 se muestran las diferentes trayectorias seguidas por el vehículo en estas simulaciones. Así mismo, en la ilustración 41 se puede observar una anomalía de esta trayectoria debida a que el vehículo llega a perder adherencia con el terreno, este problema se reproduce al aumentar el ángulo de giro máximo.

Para probar como ha funcionado las diferentes soluciones propuestas se simula la misma maniobra, pero con velocidad y un valor giro diferente (70 km/h y 100 respectivamente) (Ilustración 42).

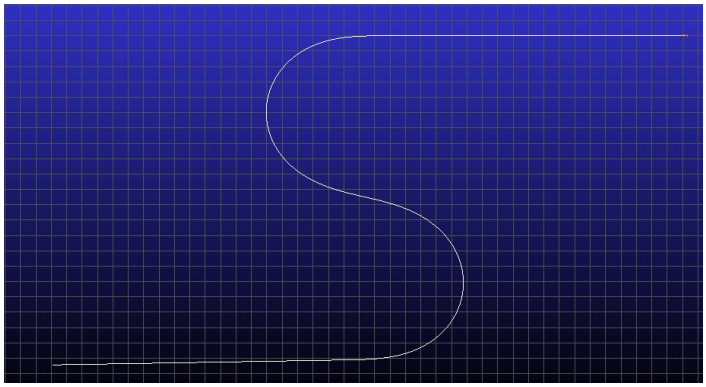


Ilustración 42: Maniobra Line change a 50 Km/h y Max steer value 100

Los resultados obtenidos se muestran en las ilustraciones 43-56. Las ilustraciones 43 y 44 corresponden a los resultados obtenido mediante algoritmos tradicionales de Machine Learning, siendo en este caso los algoritmos de árboles de decisión los que han presentado una respuesta con menor error medio. Aunque estos hayan sido los algoritmos que mejor representan la realidad en este caso, el resultado no es realmente satisfactorio debido a que en las zonas donde se produce una mayor diferencia de carga es donde hay un mayor error, siendo excesivo. Además, al introducir el análisis PCA los resultados se vuelven más inestables. Esto ocurre ya que tras procesar los datos mediante esta metodología tan solo se requiere una variable, que no es capaz de representar la realidad como se necesita.

Por otro lado, se muestra una mejoría en los resultados cuando se introducen una mayor cantidad de variables. Aunque esta mejora es sustancial, no es capaz de aproximarse a la curva real. En algunas zonas se cometen errores puntuales de hasta un 20%.

En las ilustraciones 44 y 45 se muestran los resultados que proporcionan los mismos datos, pero esta vez siendo procesados mediante redes neuronales. Al igual que anteriormente, se produce una mejoría generaliza cuando se pasan de 4 entradas a 16 del sistema.

En la ilustración 44, con 4 entradas, se ha producido una aproximación a la curva real mejor que en el caso de Machine Learning, con y sin análisis PCA. Además, la respuesta que se produce es más continua y cercana a la curva real, incluso en el caso con reducción de dimensión los resultados son mejores y estables.

En la ilustración 45, con 15 entradas, se producen dos fenómenos importantes. En primer lugar, la aproximación producida por las redes neuronales se acerca mucho a la curva real por lo que se está obteniendo una respuesta satisfactoria al problema. Sin embargo, al introducir el análisis PCA se está produciendo una deformación de la respuesta no pareciéndose en nada a la respuesta real.

El fenómeno que se produce por culpa del análisis estadístico, se debe a que el análisis PCA utiliza la varianza de las variables para poder clasificar que variables describen de mejor forma el problema. Esto implica que al tener variables y datos aun sin procesar y en valores absolutos haya variables que varían mucho, como pueden ser desplazamientos en varios ejes. Esto provoca que el modelo asuma variables para entrenar la red que no describen bien el problema provocando grandes errores.

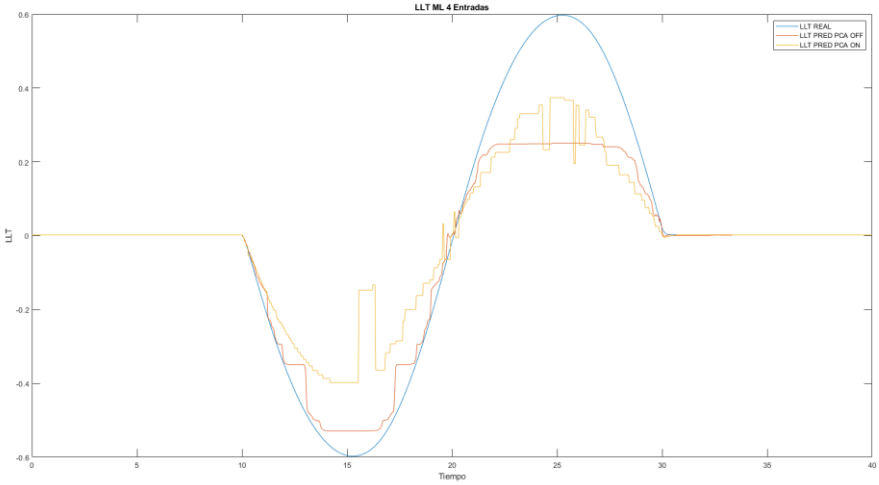


Ilustración 43: Comparativa para 4 datos de entrada con ML con la maniobra single lane change

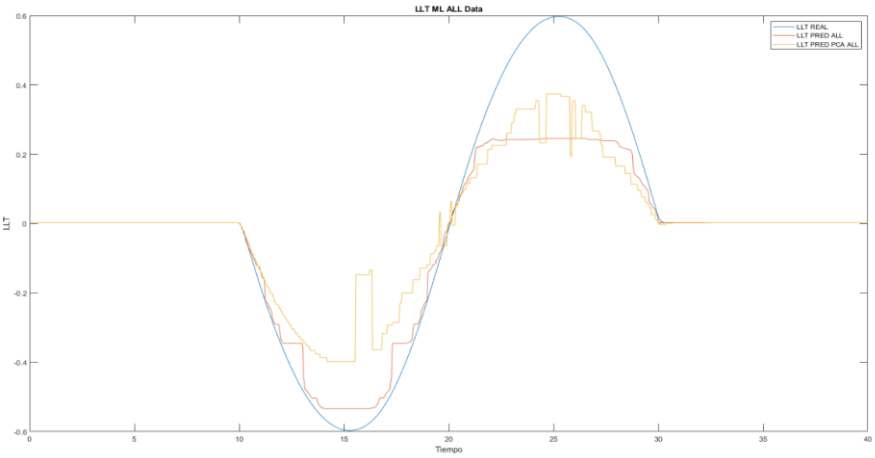


Ilustración 44: Comparativa para 15 datos de entrada con ML con la maniobra single lane change

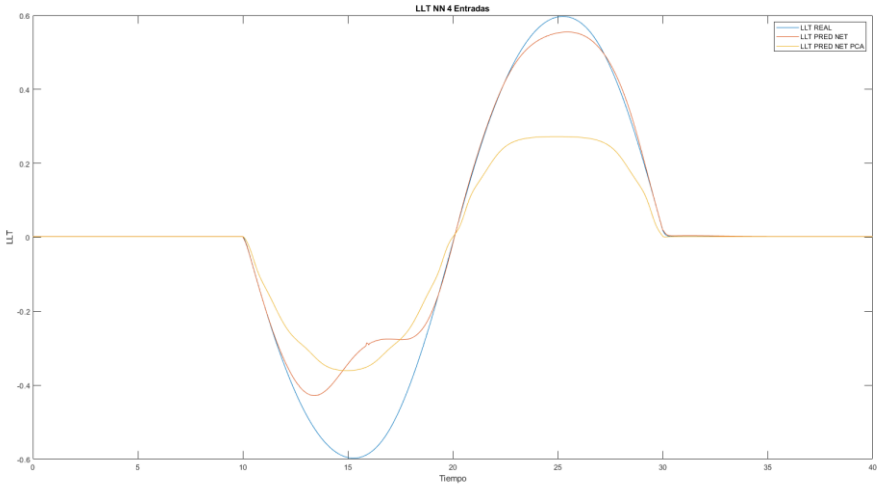


Ilustración 45: Comparativa para 4 datos de entrada con NN con la maniobra single lane change

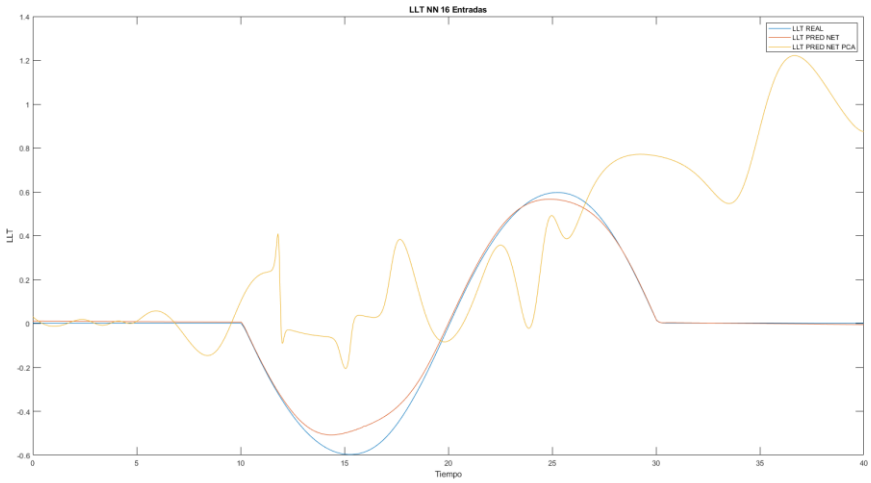


Ilustración 46: Comparativa para 15 datos de entrada con NN con la maniobra single lane change

4.1.2.2 Prueba con rampa

En el segundo caso, se simula la maniobra "Straight_Line Maintain" con el objetivo de que la perturbación la produzca el trazado. De esta forma, se utiliza una rampa creada a partir del archivo del propio software de rampa con obstáculo (Ilustración 47).

En el caso de la rampa la variable que se modifica es la velocidad lineal. Se han utilizado diferentes velocidades para la obtención de los datos (20,30,50,60,70 Km/h), mientras que para el test se modifica ligeramente la altura de la rampa y la velocidad selecciona fue 40km/h.

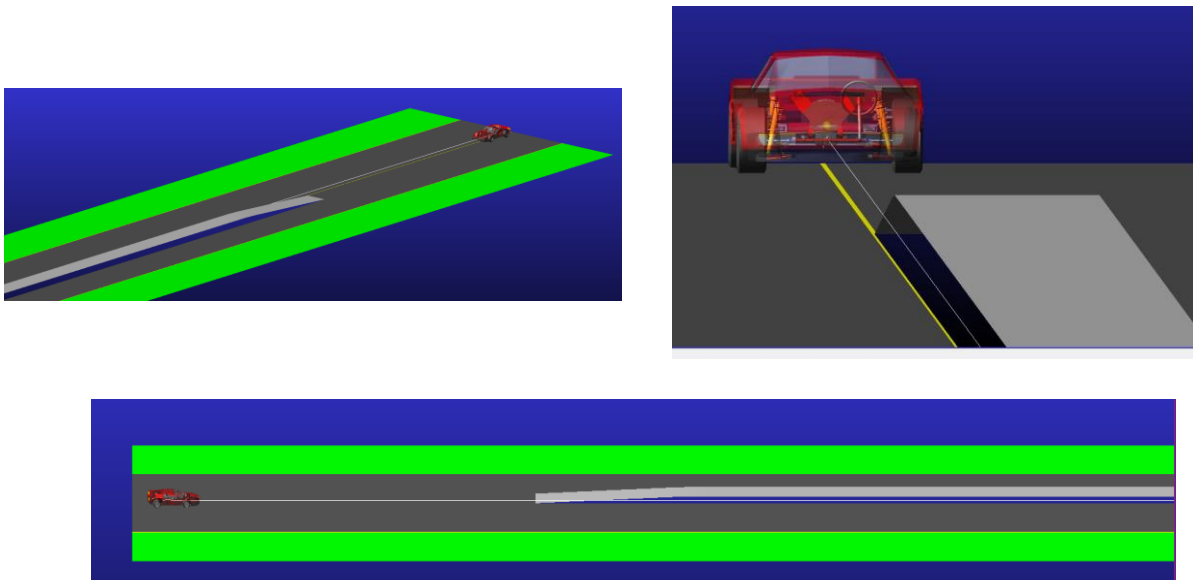


Ilustración 47: Maniobra Straight Line Maintain con rampa

Los resultados obtenidos se muestran en las ilustraciones 48-51. Las ilustraciones 48 y 49 corresponden a los resultados obtenido mediante algoritmos tradicionales de Machine Learning. En este caso los algoritmos SVM son los que han presentado una respuesta con menor error medio.

En el caso de estos algoritmos se obtienen resultados bastante parecidos a la realidad. Aunque estos resultados son por lo general aceptables presentan discontinuidades a lo largo del test, provocando que en zonas puntuales los errores se hagan enormes. Al contrario que en el caso anterior, la introducción de análisis PCA para la obtención de los resultados no hace que la señal de respuesta se haga más inestable ni discontinua, teniendo en cuenta que la señal sin PCA ya lo es.

Los resultados no mejoran considerablemente cuando se introducen más variables, se mantienen prácticamente igual por lo que en este caso el aumento de variables no se traduce en una mejora. Se mantienen las inestabilidades puntuales que generan errores puntuales altos.

En las ilustraciones 50 y 51 se muestran los resultados obtenidos con redes neuronales. Estos muestran una mayor continuidad y menos picos puntuales. A su vez, estos resultados con menos datos presentan un error generalizado mayor.

En la ilustración 50 se muestra el error cometido inicialmente donde el automóvil está aún en una zona lisa y el resultado obtenido es un 20% inferior al valor real. Tras analizar las posibles causas de este error se llega a la conclusión de que hay dos factores que pueden estar alterando los resultados. En primer lugar, las redes neuronales suelen necesitar un número mayor de datos "con información", es decir, valores que ayuden a enseñar al algoritmo aportando alguna característica nueva. En estas simulaciones iniciales se están introduciendo todos los datos directamente sin un previo pre-procesado. Además, debido a que en el momento inicial el software "deja caer" el vehículo hasta que hace contacto y empieza la simulación, se generan unos primeros instantes de desequilibrio. Esto puede confundir al algoritmo a la hora de definir su centro de referencia.

En segundo lugar, el programa de simulación nos ofrece resultados de la aceleración del cuerpo. Al estar generando aceleraciones provocadas por la inclinación del terreno, producidas por la gravedad, se puede estar generando una diferencia entre la aceleración lateral real y la transferencia de carga que se produce en la simulación.

Por el contrario, y aunque se sigue produciendo el mismo problema que anteriormente al utilizar el análisis PCA, al utilizar 15 variables de entradas para la obtención de los datos mediante redes neuronales el resultado es totalmente diferente. Además, se generan unos resultados continuos, sin picos ni alteraciones. La curva se iguala prácticamente a la curva real, teniendo unos errores mínimos.

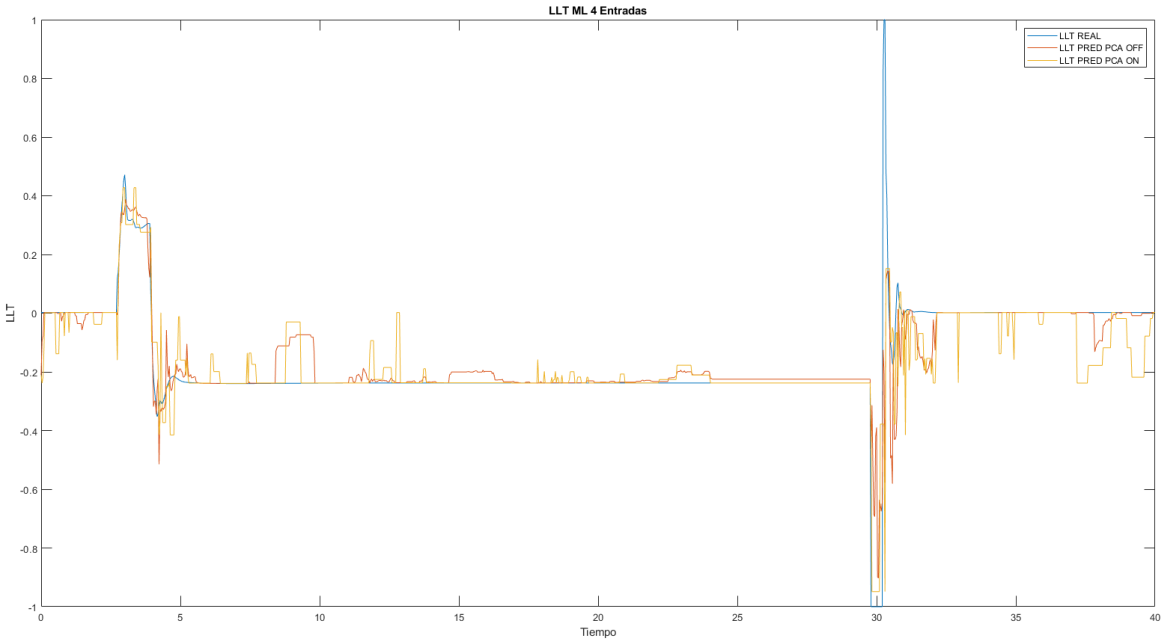


Ilustración 48: Comparativa para 4 datos de entrada con ML con la maniobra rampa

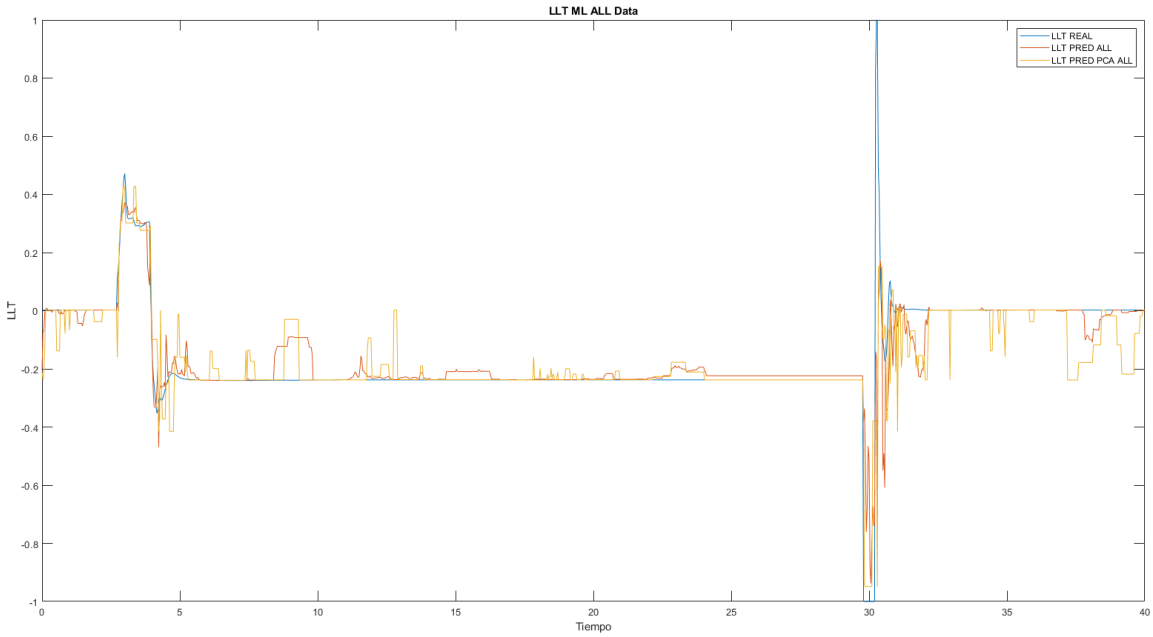


Ilustración 49: Comparativa para 15 datos de entrada con ML con la maniobra rampa

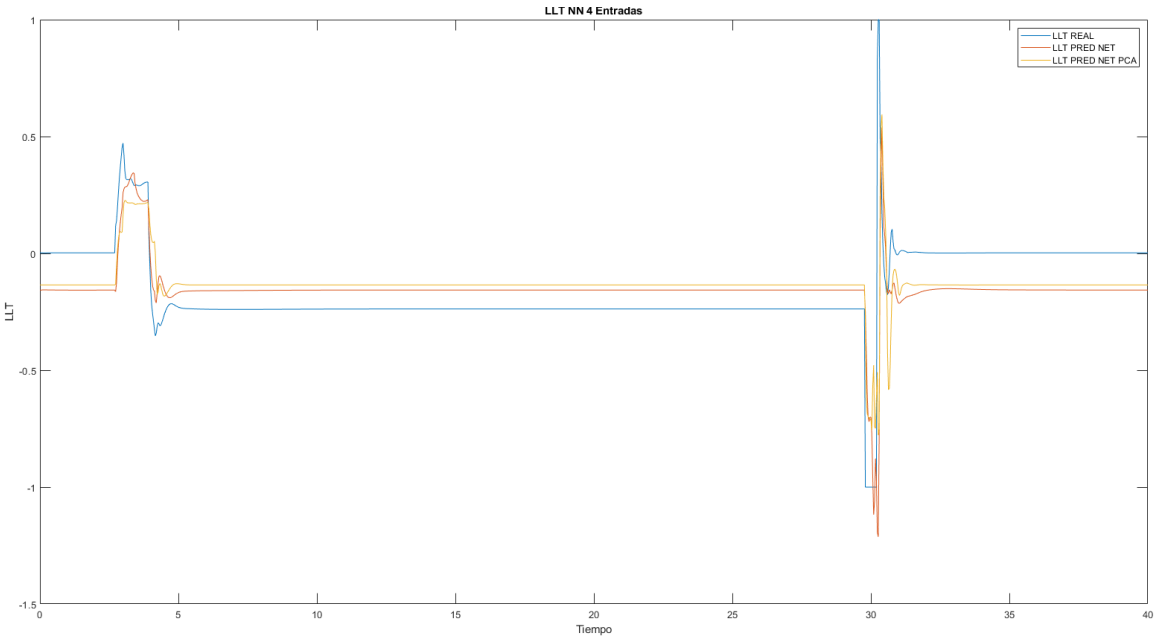


Ilustración 50: Comparativa para 4 datos de entrada con NN con la maniobra rampa

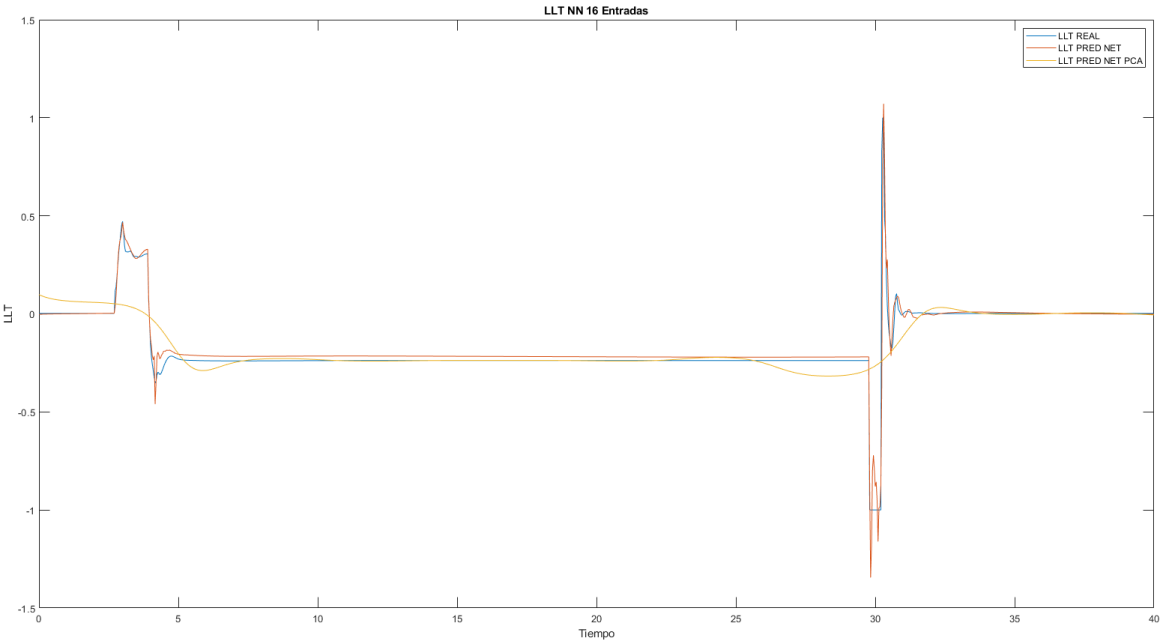


Ilustración 51: Comparativa para 15 datos de entrada con NN con la maniobra rampa

4.1.2.3 Prueba con eslalon

En esta simulación se utiliza la operación " Swept-sine steer" o lo que es lo mismo, un eslalon. En este caso se puede diferenciar el valor de Maximun Steer Value o valor máximo de giro que, al aumentar su valor hace que la amplitud de la velocidad lateral aumente también. Además, se pueden modificar los valores Initial Frecuency y Maximun Frequency o frecuencia máxima y mínima, que junto con el Frequency Rate, o tasa de aumento de frecuencia hacen que los giros se realicen más veces.

Para esta simulación, se han realizados ensayos (Ilustraciones 52 y 53) para los datos de entrenamiento con la misma velocidad lineal y se ha modificado el valor máximo de giro obteniendo:

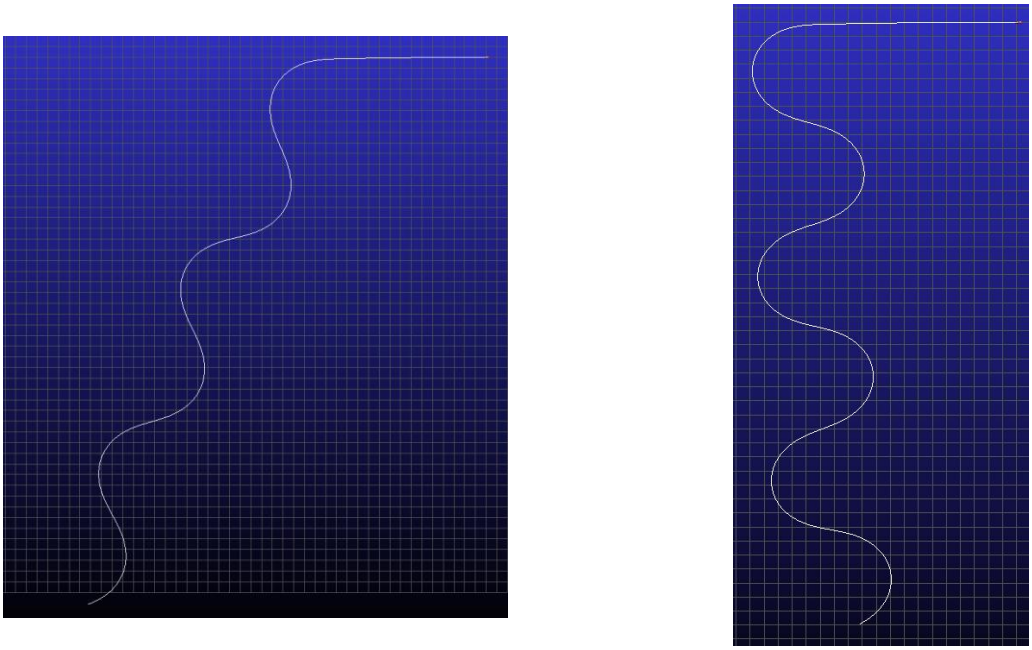


Ilustración 52: Maniobra eslalon a 50 Km/h y Max steer value 90 y 135 respectivamente

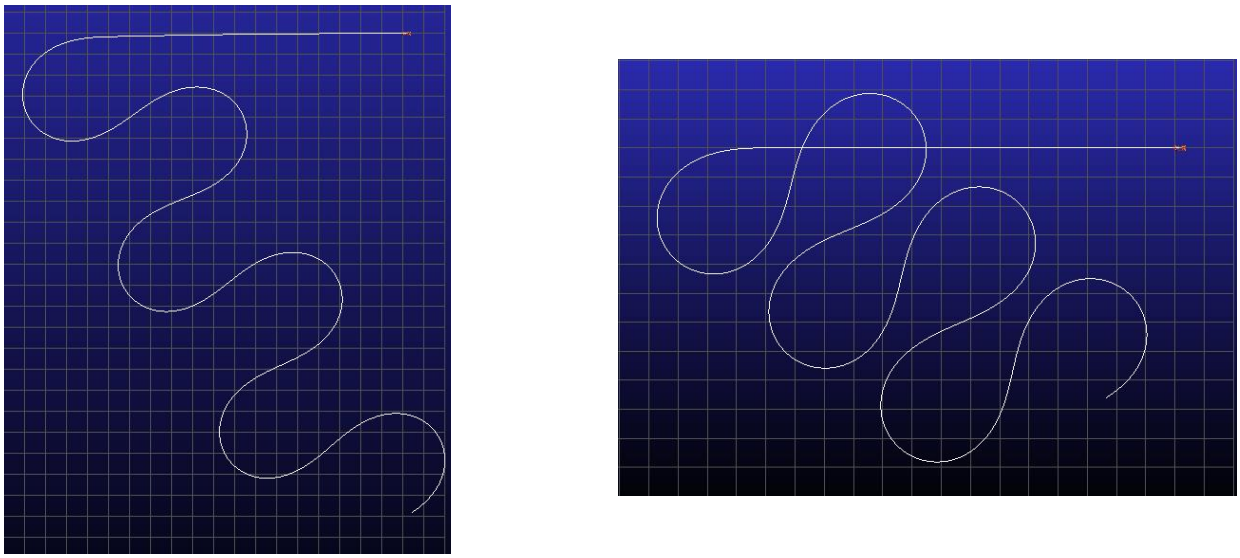


Ilustración 53 Maniobra eslalon a 50 Km/h y Max steer value 180 y 225 respectivamente

En cambio, para obtener los datos que se utilizarán para el test (Ilustración 54) del algoritmo entrenado se ha modificado la velocidad a 30 km/h y el ángulo de giro máximo a 45.

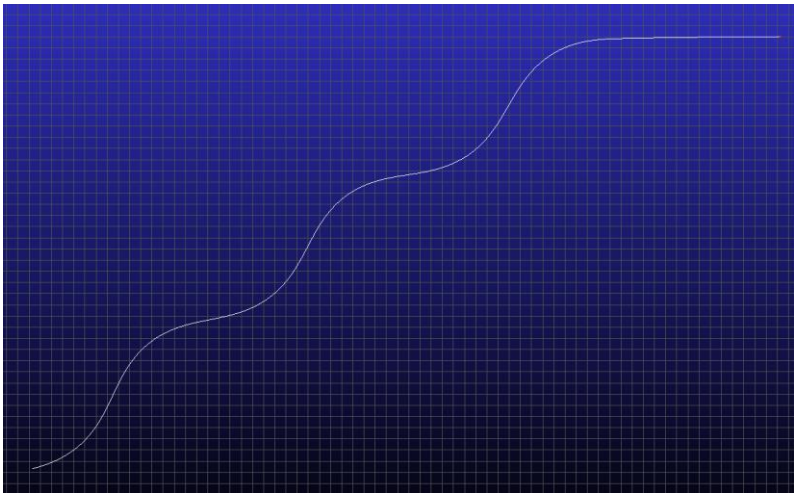


Ilustración 54: Maniobra eslalon a 30 km/h y Max steer value 45

Los resultados obtenidos se muestran en las ilustraciones 55-59. Las ilustraciones 55 y 56 corresponden a los resultados obtenido mediante algoritmos tradicionales de Machine Learning, siendo en este caso los algoritmos de árboles de decisión los que han presentado una respuesta con menor error medio. Las ilustraciones 57 y 58 son las correspondientes a los resultados ofrecidos por el algoritmo con redes neuronales.

En estas simulaciones, en comparación con los resultados obtenidos anteriormente en las pruebas con la rampa y el cambio de carril, se obtienen unos resultados excepcionales. Las curvas de datos obtenidas como resultados son prácticamente iguales a la realidad.

En el caso del Machine Learning, los resultados son ligeramente peores que los resultados ofrecidos por las redes neuronales. Al utilizar 4 variables de entrada, con Machine Learning, las curvas son similares a la realidad pero se siguen produciendo, como en casos anteriores aunque no tan acentuado, pequeños saltos en algunas zonas. En cambio, el resultado tanto con 4 variables como con 15 es muy bueno. Sin embargo, al introducir un análisis PCA se produce una reducción de la calidad de los resultados.

Los resultados que ofrece el algoritmo de redes neuronales son prácticamente idénticos a la realidad. En este caso la diferencia entre la utilización de 4 o 16 variables no es apreciable. En cambio, al utilizar la reducción de dimensiones, los resultados empeoran considerablemente si se compara con la realidad, aunque si se compara con los resultados obtenido con el Machine Learning, se puede ver que empeora de la misma forma.

Por último, y a modo de ejemplo con los casos anteriores, se comparan los errores a lo largo del test de las diferentes formas de entrenar los algoritmos. Se comprueban que los errores relativos a los algoritmos tradicionales de Machine Learning además de ser mayores, muestran una geometría con muchos saltos. Sin embargo, los errores que producen las redes neuronales son menores a la vez que tienen una menor varianza (Ilustración 59).

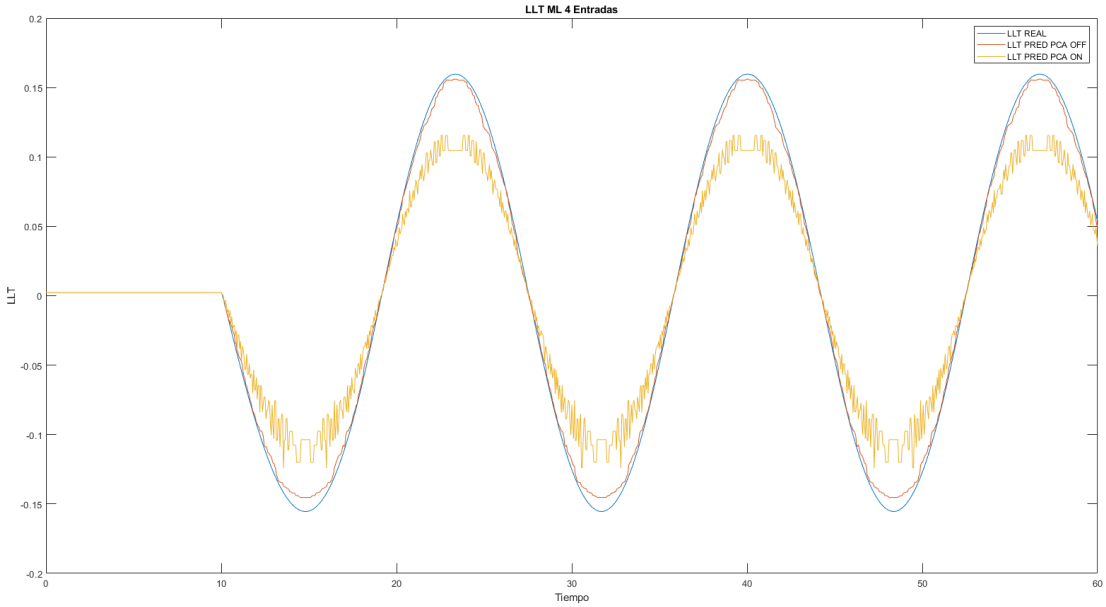


Ilustración 55: Comparativa para 4 datos de entrada con Machine Learning con la maniobra eslalon

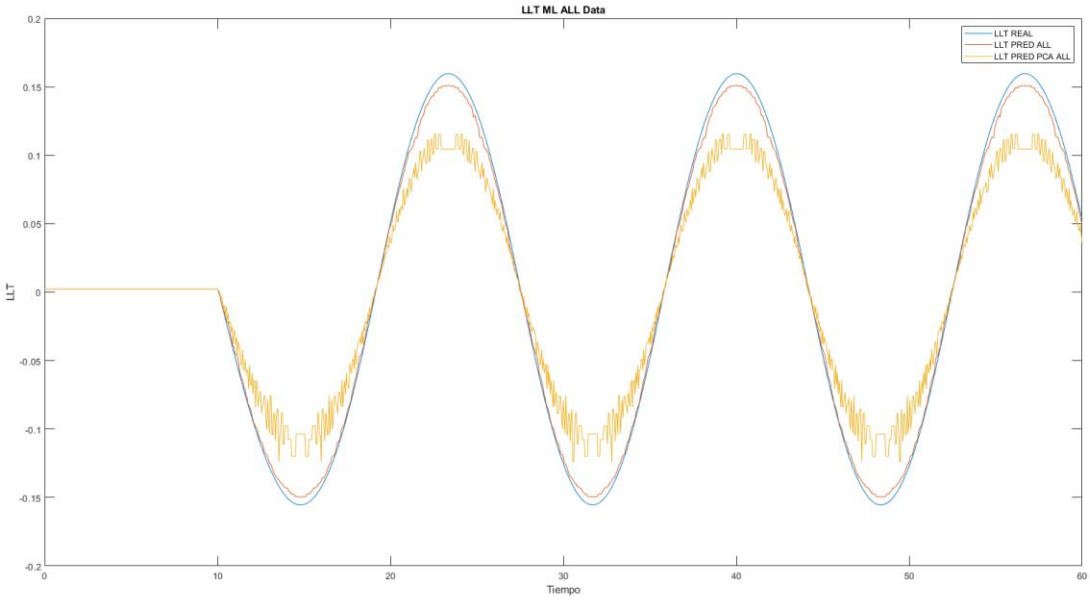


Ilustración 56: Comparativa para 16 datos de entrada con Machine Learning con la maniobra eslalon

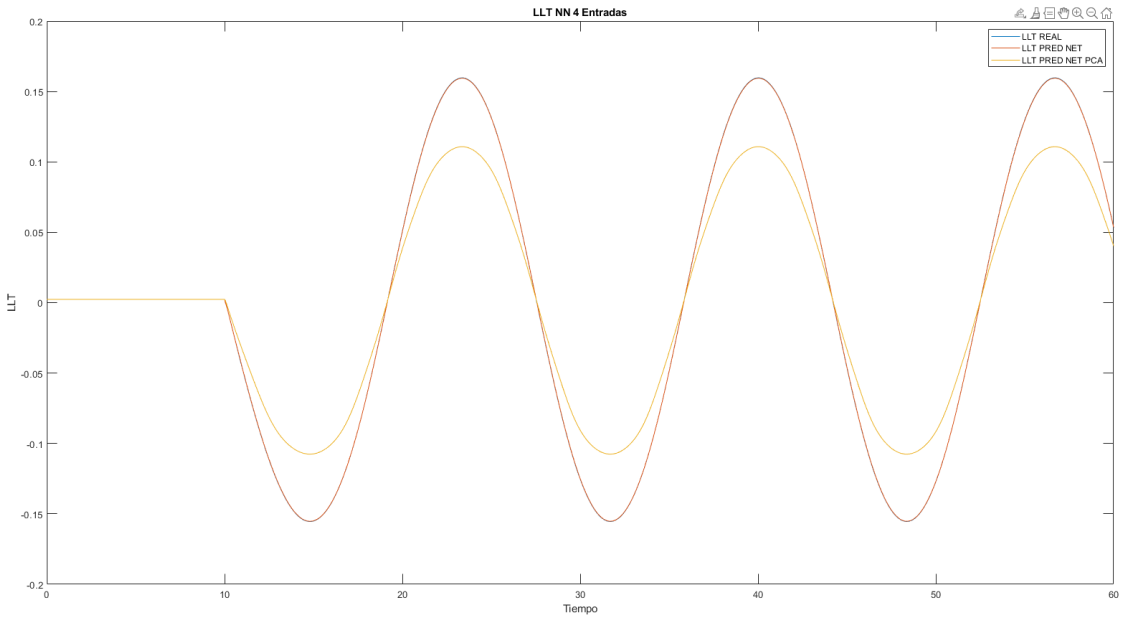


Ilustración 57: Comparativa para 4 datos de entrada con NN con la maniobra eslalon

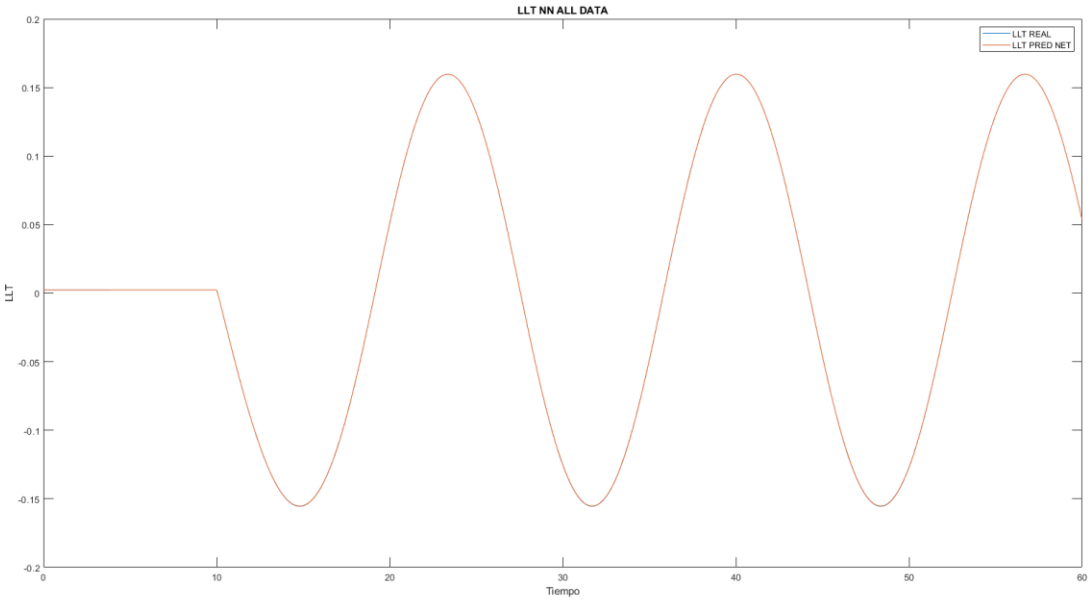


Ilustración 58: Comparativa para 16 datos de entrada con NN con la maniobra eslalon

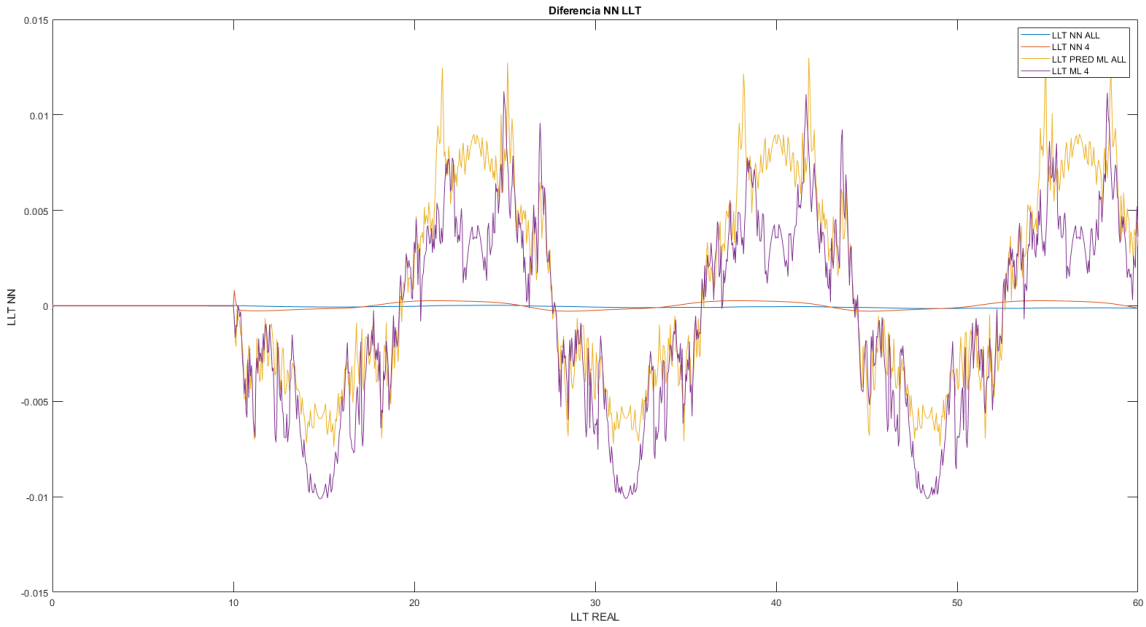


Ilustración 59: Comparativa de errores para maniobra eslalon

4.1.2.4 Prueba uniendo todas las maniobras (Cambio de carril, rampa y eslalon)

En este apartado, y una vez analizado los tres casos (Single Line change, rampa, eslalon), se continúa entrenando con todos los datos obtenidos y realizando el test con diferentes casos de los tres tipos de maniobras analizadas. De esta forma, se realizarán los mismos cálculos, pero con tres tipos de entrada de datos para el entrenamiento.

Este último tipo de prueba se realiza para estudiar cómo se comportan los diferentes algoritmos a la hora de introducir datos de maniobras diferentes.

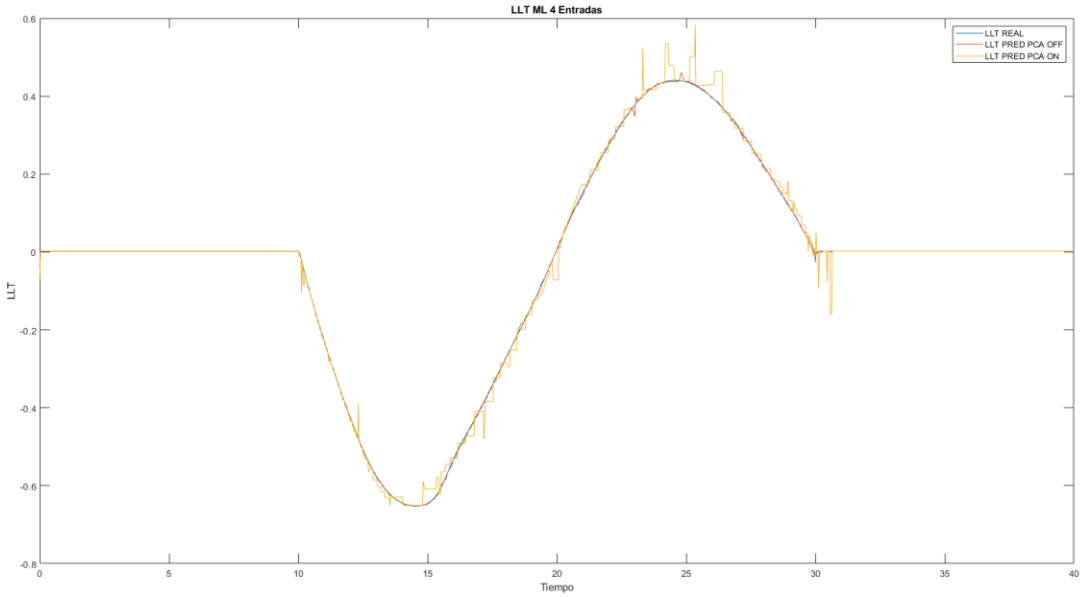


Ilustración 60: Comparativa para todos los datos con Machine Learning con todas las maniobras

Unir las diferentes maniobras ha provocado que los resultados mejoren de forma general. En todos los casos se han producido mejoras. En el caso de los algoritmos tradicionales los resultados han mejorado en mayor proporción, aunque anteriormente habían sido peores.

Por otro lado, se han producido algunos fenómenos que se muestran en las ilustraciones 60-62. En primer lugar, en la ilustración 60 se muestra la mejoría de los resultados ofrecidos por los algoritmos tradicionales. Mientras que en el caso en que solo se utilizan datos de la misma maniobra los datos eran pobres, estos son muy similares a los datos de referencia incluido cuando se utiliza la reducción de dimensiones.

En la ilustración 61 se muestra la misma problemática que había sucedido en el ensayo que solo se utilizaban los datos de la maniobra rampa, se produce un error continuo al inicio de la respuesta. Como se comentó anteriormente, la problemática es debida a la propia maniobra y al no procesar los datos previamente al entrenamiento del sistema.

Por último, en la ilustración 62 se vuelve a hacer referencia a la misma problemática. Es de nuevo significativa porque hasta ahora solo había aparecido a la hora de utilizar redes neuronales, pero esta vez se reproducen en el caso de algoritmos tradicionales de Machine Learning.

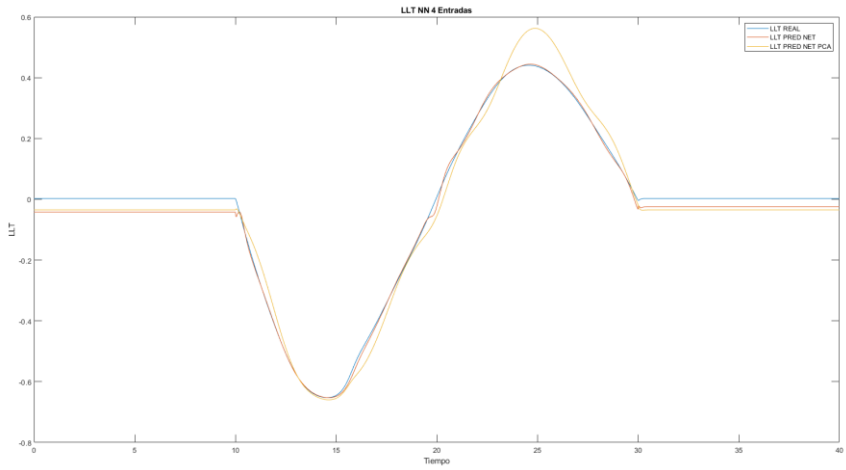


Ilustración 61: Comparativa para 4 datos con NN con todas las maniobras

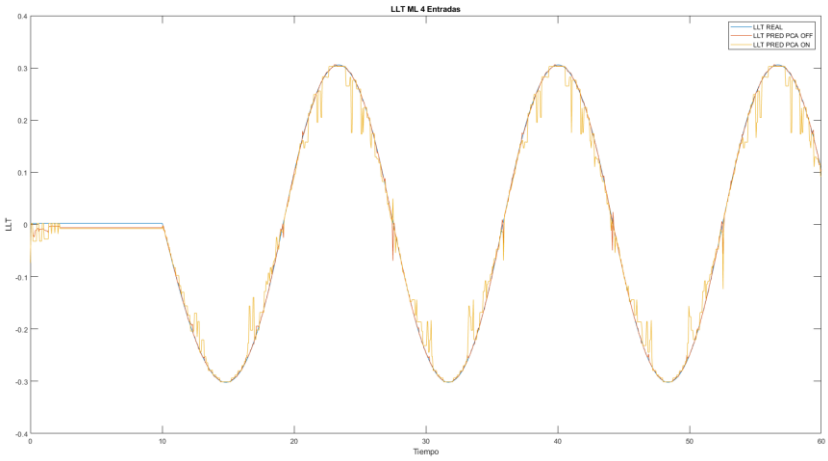


Ilustración 62: Comparativa para 4 datos con Machine Learning con todas las maniobras

4.1.2.5 Conclusiones de la comparación entre algoritmos tradicionales y redes neuronales

Durante este apartado se ha mostrado el procedimiento y los resultados obtenidos de la comparación entre dos formas diferentes de afrontar un problema mediante una metodología de entrenamiento automático.

Tras el estudio de los resultados ofrecidos por ambos procedimientos se decide continuar la implementación del aprendizaje automático mediante redes neuronales debido a:

- Los resultados ofrecidos por las redes neuronales generalmente presentan un menor error. En todos los ensayos que se han realizado, las redes neuronales ofrecían unos mejores resultados, acercándose más a la realidad. Además, se obtenían resultados con una mayor continuidad y no ofrecían discontinuidades.

  Aunque en algunos casos, como con la maniobra de la rampa, se producen errores de discontinuidad en los intervalos iniciales, se conoce la procedencia de este error. Además, el resultado en la zona donde el vehículo experimentaba cambios en la transferencia de carga seguía siendo mejor.

- Las redes neuronales admiten una mayor personalización para el problema a estudiar. Las redes neuronales brindan la oportunidad de modificar el número de neuronas, el número de capas, el número de neuronas por capa ofreciendo una gran ventaja a la hora de simular diferentes escenarios con un numero de datos diferente.

- El análisis PCA utilizado en este apartado muestra que, aunque reduce el número de variables del problema, no obtiene unos resultados acordes a los esperados. Además, de esta manera se está perdiendo totalmente el sentido físico del problema provocando que sea más difícil diseñar una respuesta para evitar las inestabilidades producidas con la transferencia de carga excesivas.

Por lo tanto, en los siguientes apartados se abordará el preprocesamiento de los datos para una mejora de la repuesta inicial junto con un estudio para la viabilidad de la reducción de dimensiones utilizando redes neuronales. Además, como se utilizan redes neuronales a partir de ahora, se analizará como se puede ajustar cada red al problema de este proyecto en cada caso. También se profundizará en el uso del análisis PCA enfocándolo desde otro punto de vista.

4.2 Implementación del problema mediante redes neuronales

A continuación, el proyecto se centra exclusivamente en el estudio del vuelco de vehículos mediante redes neuronales. En primer lugar, se procede a un preprocesado de los datos que se introducirán en el sistema de redes neuronales para mejorar la respuesta. El preprocesado se centra principalmente en tres aspectos:

- Eliminar datos redundantes
- Normalizar valores
- Referenciar valores a datos iniciales

Este conjunto de técnicas se intenta implementar para mejorar la respuesta, tanto computacionalmente como para mejorar la respuesta del resultado esperado, el LLT.

4.2.1 Preprocesado de datos

4.2.1.1 Eliminación de datos redundantes

La base de datos que se están utilizando hasta ahora es extraídas directamente desde el software de simulación y se trabaja con todos los datos. Se presenta una sobreexposición de datos similares donde no se está aportando valor añadido al aprendizaje, como pueden ser los instantes donde el vehículo está varios segundos yendo en línea recta hasta que da comienzo un giro.

Al introducir este tipo de datos en el aprendizaje se aumenta la probabilidad de que se tomen decisiones teniendo en cuenta el "ruido" de la base de datos. A este problema se le suele llamar sobreentrenamiento y reduce la precisión en la respuesta al igual que aumenta el tiempo de entrenamiento, es decir, aumentando el número de datos no significativos el algoritmo aprende más lento y puede llegar a conclusiones erróneas.

En este caso se intenta resolver el problema añadiendo varias funciones dentro del desarrollo en Matlab:

- En primer lugar se eliminan cualquier fila duplicada en la base de datos mediante el comando *unique()* de Matlab. Mediante este comando, se comparan todas las filas de la base de datos y se tan solo se queda con una de las filas que aportan información
- En segundo lugar, se quiere identificar las filas de la base de datos donde el vehículo circula en línea recta y sin aportar información sobre las aceleraciones y las fuerzas. Para esto, simplemente se eliminan todas las filas donde la aceleración lateral, uno de los parámetros más determinantes en el vuelco, tiene valores extremadamente pequeños (Ilustración 63).
- En tercer lugar, teniendo en cuenta la influencia de la aceleración de balanceo, se identifica un rango general donde los valores de dicha aceleración fluctúan. Se realiza este preprocesado debido a que el software de simulación en determinadas situaciones tiene picos en los valores, como pueden ser los momentos iniciales o cuando el vehículo no tiene contacto con todas las ruedas. Para evitar este problema se delimita un rango de aceleración y se eliminan todas las filas de la base de datos que estén fuera de este rango de valores (Ilustración 64).
- Por último, debido al problema de inicio del software de simulación que lanza el vehículo hasta que tiene contacto con el suelo ficticio, se introducen valores iniciales no acordes a la simulación. Para eliminar este problema, se elimina el primer segundo de cada simulación que se utiliza para entrenar el algoritmo.

Preprocesado de datos

Eliminar ceros

```
flag_1=0;
for aux_1=2:(height(dataTrain))
    if dataTrain.chassis_accelerations_lateral(aux_1)<0.009
       if dataTrain.chassis_accelerations_lateral(aux_1)>-0.009
           flag_1=flag_1+1;
        Vector_aux(flag_1)=aux_1;
       end
    end
end
dataTrain([Vector_aux],:)=[];
```

Ilustración 63: Extracto de programación: Eliminar ceros

Eliminar Roll acceleration

```
flag_3=0;
Vector_aux_3=0;
for aux_3=2:(height(dataTrain))
    if dataTrain.chassis_accelerations_roll(aux_3)>1.2
        flag_3=flag_3+1;
        Vector_aux_3(flag_3)=aux_3;
    elseif dataTrain.chassis_accelerations_roll(aux_3)<-1.2
           flag_3=flag_3+1;
           Vector_aux_3(flag_3)=aux_3;
    end

end
if Vector_aux_3~=0
    dataTrain([Vector_aux_3],:)=[];
end
```

Ilustración 64: Extracto de programación: Eliminar aceleración de balanceo

4.2.1.2 Normalización de datos

Una de las técnicas más utilizadas para optimizar la respuesta en Machine Learning es la normalización de datos. Se utiliza para cambiar los valores de las columnas de la base datos para usar una escala común. La normalización consiste en comprimir o extender los valores de las entradas para que se engloben dentro de un rango.

En este caso se va a utilizar un escalado de variables donde los datos serán normalizados entre los valores máximo y mínimos de las variables. El posible problema de este tipo de normalización es que si existe ruido dentro de la base de datos se ve amplificado.

La normalización de los datos se utiliza en todas las variables que se han introducido como entrada del sistema y en el LLT, se compara con los resultados que se han obtenido anteriormente para poder comparar los resultados (Ilustración 65).

Normalización de datos

```
[Filas,Columnas]=size(dataTrain);

for  aux_4=2:Columnas
max_1=max(dataTrain{:,aux_4});
min_1=min(dataTrain{:,aux_4});
     if abs(max_1)>abs(min_1)
         for aux_5=2:Filas
              dataTrain{aux_5,aux_4}=dataTrain{aux_5,aux_4}/max_1;
         end
     else
         for aux_5=2:Filas
              dataTrain{aux_5,aux_4}=dataTrain{aux_5,aux_4}/min_1;
         end
     end
end

[Filas,Columnas]=size(dataTest);

for  aux_4=2:Columnas
max_1=max(dataTest{:,aux_4});
min_1=min(dataTest{:,aux_4});
     if abs(max_1)>abs(min_1)
         for aux_5=2:Filas
              dataTest{aux_5,aux_4}=dataTest{aux_5,aux_4}/max_1;
         end
     else
         for aux_5=2:Filas
              dataTest{aux_5,aux_4}=dataTest{aux_5,aux_4}/min_1;
         end
     end
end
```

Ilustración 65: Extracto de programación: Normalización de datos

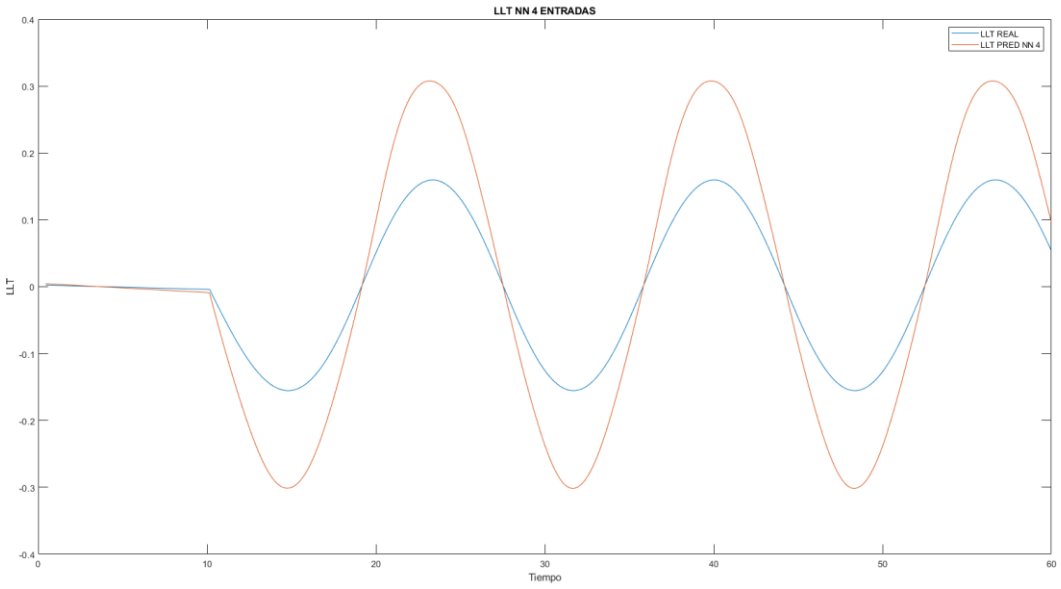


Ilustración 66: Normalización de datos

En la ilustración 66 se aprecia el empeoramiento de los resultados es general, se muestra como ejemplo significativo el ensayo con 4 entradas, que suele ofrecer una mayor estabilidad para entrenar la red neuronal. Se producen grandes errores en la predicción y se desestima la opción de normalizar los datos para intentar mejorar el procesado de los datos por parte de las redes neuronales.

4.2.1.3 Referenciar valores a datos iniciales

Tras desestimar la normalización de los datos, también se intenta mejorar la respuesta del sistema filtrando los datos, está vez centrándose en los datos del LLT que se introducen al sistema para poder realizar el entrenamiento supervisado.

Se observa que al inicio de las simulaciones la transferencia de carga lateral no es siempre igual. Al inicio de la simulación el software lanza al vehículo y produce que en los momentos iniciales el LLT adquiera valores distintos de cero. Además, transcurrido un periodo de tiempo prudencial se sigue apreciando unos resultados del LLT distintos de 0 en ensayos en línea recta.

Para solventar este problema, se intenta referenciar todos los valores del LLT a un valor inicial (25) que se considerará el valor neutro de esta variable. Este proceso se realiza en la función con la que se calcula el LLT:

$$\mathrm{LLT} = \left(\frac{F_{n2} - F_{n1}}{F_{n2} + F_{n1}}\right) - \left(\frac{F_{n2}(1) - F_{n1}(1)}{F_{n2}(1) + F_{n1}(1)}\right) \qquad (25)$$

El proceso utilizado para analizar si este tipo de tratamiento de datos es válido, es el mismo que en el caso anterior. Se ha calculado con los mismos datos, con el mismo número de capas y neuronas modificando la forma en la que se obtiene el LLT.

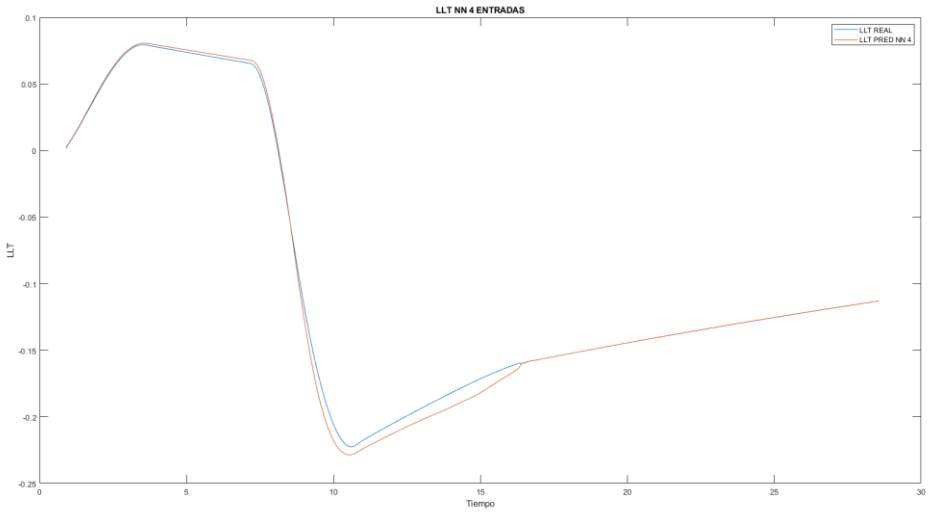


Ilustración 67: Referenciar datos LLT sin preprocesado

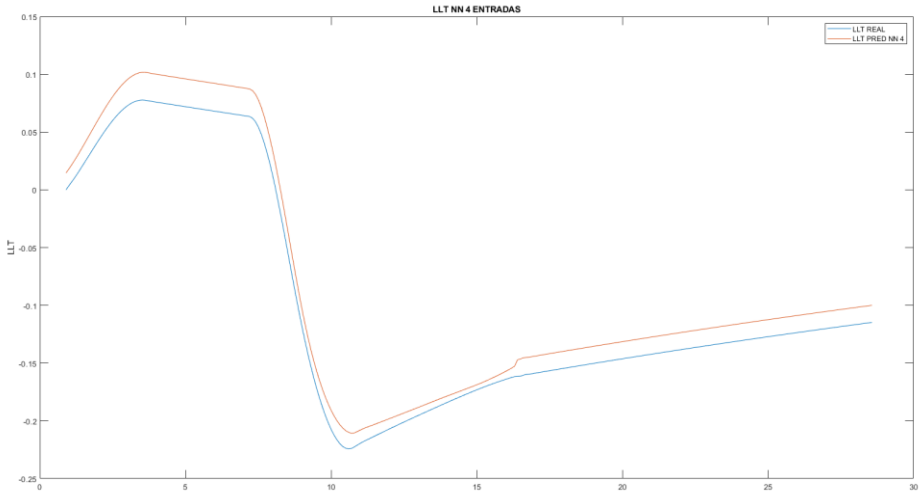


Ilustración 68: Referenciar datos LLT con preprocesado

Los resultados (Ilustraciones 67 y 68) vuelven a mostrar que este tipo de preprocesado de datos no mejora en este determinado caso los resultados obtenidos, introduciendo variaciones que empeoran el resultado de la predicción. Por lo tanto, se decide desestimar también esta opción debido a que los resultados previos han sido mejores.

4.2.2 Consideraciones iniciales de redes neuronales

Después de haber determinado como se va a realizar el tratamiento previo de los datos surgen otras incógnitas como son el número de datos necesarios para entrenar de una forma eficaz y eficiente a la red neuronal. También surge la incógnita del número idóneo de neuronas, capas y neuronas por capas necesario para que el sistema funcione de la mejor forma posible, optimizando los recursos computacionales y la respuesta del sistema.

Además, uno de los problemas más comunes que se producen al trabajar con redes neuronales es el sobreajuste (*overfitting*). Consiste en el sobreentrenamiento de un algoritmo de aprendizaje automático supervisado. Estos algoritmos deben predecir un resultado en función de lo aprendido con unos datos de entrenamiento. El problema surge cuando se utilizan datos extraños, con información confusa o se sobreentrena el algoritmo. Se denomina sobreajuste cuando un modelo de aprendizaje automático se ajusta tanto a los datos del entrenamiento que pierde la propiedad de generalizar para la predicción de datos generales(Ilustración 69).

El sobreajuste se produce principalmente por dos causas: cantidad y/o calidad de los datos y cantidad y/o disposición de las neuronas. Es por ello que tener en cuenta estos aspectos en fundamental para tener un algoritmo que funcione de manera correcta.

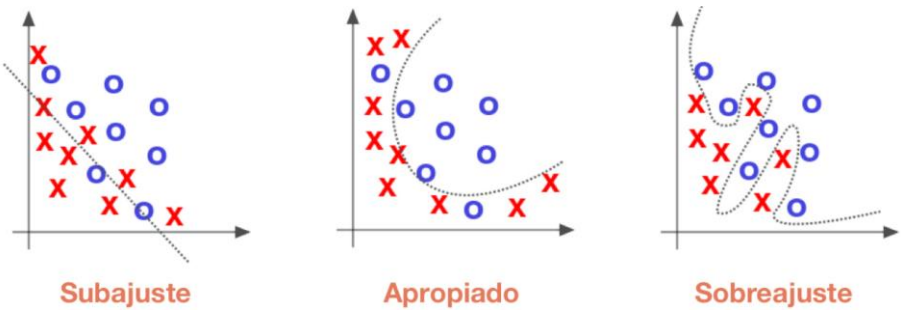


Ilustración 69: Representación subajuste y sobreajuste

4.2.2.1 Número de neuronas y capas para la red neuronal

Una de las principales incógnitas a determinar para cualquier proyecto de aprendizaje automático con redes neuronales es cuantas neuronas se necesitan y como se van a organizar. Esta cuestión es muy importante porque condicionará el resultado y el tiempo necesario para obtenerlo.

Una red multicapa puede presentar muchas disposiciones, habiendo una o varias capas ocultas entre la capa de entrada y la capa de salida. El número de capas y el número de neuronas por cada capa oculta definirá el tamaño de la red.

Las recomendaciones generales para la creación de una red neuronal para un problema no complejo son las de usar una sola capa oculta, usar el menor número posible de neuronas y realizar multitud de ensayos. La utilización de una sola capa oculta se recomienda debido a que según el Teorema de Aproximación Universal (TAU) se dice que bajo condiciones determinadas una red neuronal puede aproximar cualquier función continua con una sola capa oculta, aunque puede llegar a tener un número muy elevado de neuronas. Solo se recomienda el uso de más capas ocultas cuando se esté aumentando progresivamente el número de neuronas y el error obtenido siga siendo menor.

En una primera instancia, se suele usar la regla de la pirámide geométrica (25) que se utiliza para seleccionar el número de neuronas en las capas ocultas. El número de neuronas sigue una forma piramidal, siendo la mayor concentración de neuronas a la entrada de la red, y el menor a la salida. Para redes comunes donde solo haya una capa oculta:

$$h = m * n \tag{25}$$

donde:

- n: neuronas de entrada
- m: neuronas de salida
- h: número de neuronas de la capa oculta

La regla de la pirámide geométrica es una aproximación del tamaño de capas oculta. Se recomienda generalmente iniciar con un número pequeño de neuronas en la capa oculta y seleccionando un criterio, entrenar la red aumentando el número de neuronas hasta encontrar el equilibrio deseado entre tiempo computacional y resultados. A esta técnica se la conoce como curva de aprendizaje.

La curva de aprendizaje se denomina al diagrama que representa en el eje horizontal el transcurso del tiempo y en el eje vertical el número de éxitos. Esta definición adaptada a las redes neuronales se define como el diagrama donde en el eje horizontal se representan el número de neuronas y en el eje vertical se representan los valores de los errores cometidos en la predicción de la red.

Se decide utilizar este tipo de metodología debido a que se está utilizando un software de programación que permite la opción de realizar un cálculo secuencial variando el número de neuronas en la capa oculta. Además, se puede fijar un valor que haga que la red neuronal funcione de una mejor manera que con una aproximación empírica como es el teorema de la pirámide geométrica.

Por la propia definición de la opción que se decide elegir para el cálculo del número de neuronas en la capa oculta, no se define un número fijo de neuronas para el problema, sino que cada simulación con su número determinado de datos de entrada tendrá un número de neuronas asociado. Este número de neuronas siempre ira asociado a minimizar el error de la predicción que se realiza con la red neuronal. Se puede observar su programación en la ilustración 70.

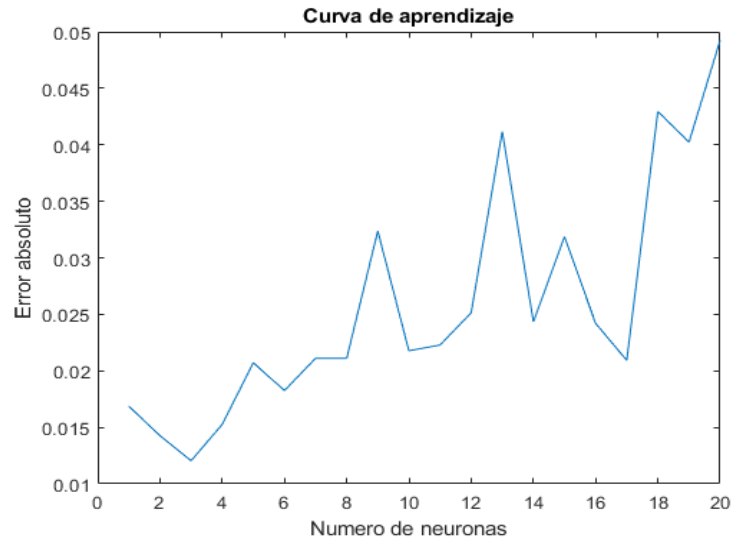
```
dataTrain_NN=dataTrain{:,[11 12 16 6 13]};
dataTest_NN=dataTest{:,[11 12 16 6 13]};
Rango_num_Neuronas=(1:20);
for aux_4=Rango_num_Neuronas
NN_4=fitnet(aux_4);
NN_4.divideParam.trainRatio=70/100;
NN_4.divideParam.valRatio=15/100;
NN_4.divideParam.testRatio=15/100;

[NN_4,~]=train(NN_4,dataTrain_NN',LLT_Del');
LLT_Del_NN=NN_4(dataTest_NN');

Error_prediccion(aux_4)=mean((abs(LLT_Del_Test-LLT_Del_NN')));

end

plot(Rango_num_Neuronas,Error_prediccion)
title('Curva de aprendizaje')
xlabel('Numero de neuronas')
ylabel('Error absoluto')
```

Ilustración 70: Extracto de programación: curva de aprendizaje

En primer lugar, se definen los datos que se van a utilizar y el rango de valores para el número de neuronas. Una vez definido este rango, se entra en un bucle donde se van a entrenar todas las posibles redes neuronales. A continuación, y sin salir del bucle, se utilizan los datos para lo comprobación, o los llamados datos de test. Con estos datos se puede obtener el error medio absoluto de la predicción para cada red.

Tras haber acabado el bucle descrito, se obtienen los errores medios para el rango de valores que puede tomar el número de neuronas. De esta forma, comparando los valores de los errores tan solo es necesario seleccionar el número de neuronas para el cual, el error se minimiza.

Teóricamente, si la relación entre número de neuronas y error fuese una relación directa, se debería obtener una curva donde al aumentar el número de neuronas se debería reducir el error, pero en el caso de este proyecto, y al ser un problema con correlaciones altas se produce el fenómeno contrario. Se crea una curva donde los errores menores se dan para un número pequeño de neuronas (Ilustración 71). Al aumentar el número de neuronas se produce un aumento del error medio debido al overfitting producido por un exceso de neuronas.




```
Error_abs_prediccion = 0.0120
Numero_Neuronas = 3
```

Ilustración 71: Ejemplo de curva de aprendizaje de red neuronal

4.2.2.2 Tamaño de base de datos

Los problemas que se presentan en referencia a los datos son la cantidad y la calidad de la base de datos que se utilizará para entrenar la red neuronal. Este apartado se centra en la cantidad de los datos mientras la calidad de los datos se ha abordado anteriormente en el apartado de preprocesado de datos.

Hay varios factores que afectan a la cantidad de datos necesarios para el modelo:

- En primer lugar, es el número de variables no correlacionadas o débilmente correlacionadas en el conjunto de datos. El aprendizaje automático se basa en las ideas de correlación, por lo tanto, si se aumenta el número de variables relacionadas entre sí, se disminuirá la incertidumbre del modelo
- En segundo lugar, es la complejidad del sistema que se quiere simular. La complejidad de un modelo suele ir asociada al número de parámetros correlacionados y no correlacionados. A mayor número de variables no correlacionadas la complejidad del sistema aumenta.
- Por último, el intervalo en la recogida de los datos es un factor crítico. Consiste en obtener la cantidad de datos necesarias en un intervalo de tiempo para ser capaz de representar la realidad. Por ejemplo, en el caso de maniobras lentas, se podrá obtener un buen resultado con una toma de datos menor que en el caso de maniobras rápidas, donde los cambios se hacen de forma más repentina y si el intervalo es demasiado grande se puede llegar a perder información. En el ejemplo de la ilustración 76 se muestra la diferencia entre dividir en dos o en tres el intervalo.

En el problema que se plantea en este proyecto se conoce que las variables están relacionadas entre sí, de hecho, unos de los objetivos de este proyecto es encontrar las variables, correlacionadas entre sí para obtener una respuesta fiable con el mínimo número de variables. También se sabe que el sistema no tiene una complejidad exagerada debido a que es una primera aproximación al cálculo del vuelco mediante redes neuronales y, por tanto, tampoco será de excesiva importancia los intervalos que se tomen porque se utilizaran maniobras donde la diferencia se mida en tiempos mayores que los intervalos que se utilizan.

Respecto al sobreajuste por exceso de datos, que provocan un ajuste no generalizado del problema, se tendrá que tener muy en cuenta el número total de datos con los que se entrenará la red neuronal. Debido a que no existe una forma exacta de estimar la cantidad de datos necesarias para entrenar una red neuronal, ni tampoco existe la forma exacta de determinar a partir de que cantidad de datos se corre el riesgo de sobre ajustar el sistema, generalmente se utilizan reglas experimentales para tener un valor aproximado del número de datos necesarios.

Una de las reglas experimentales más extendidas por su uso es (26) (Sheela and Deepa, 2013) :

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))} \qquad (26)$$

Donde:

$N_h = Numero\ de\ neuronas\ en\ la\ capa\ oculta$

$N_s = Número\ de\ ejemplos\ de\ la\ base\ de\ datos$

$N_i = Número\ de\ neuronas\ en\ la\ capa\ de\ entrada$

$N_o = Número\ de\ neuronas\ en\ la\ capa\ de\ salida$

$\alpha = Factor\ de\ escala\ entre\ 2 - 10$

Se determina, tomando el peor caso con $\alpha = 10$:

N neuronas	1	2	3	4	5	6
N output	1	1	1	1	1	1
N input	5	5	5	5	5	5
N samples	60	120	180	240	300	360

Tabla 3: Relación de número de datos y neuronas

En la tabla 3 se muestran el número de datos necesario en relación a las neuronas de la capa oculta. En el caso de este proyecto esta ecuación es útil porque se va a utilizar una capa oculta de neuronas. Esto es debido a que el problema no tiene una complejidad alta y con una capa será suficiente para determinar el propósito del proyecto.

4.2.3 Análisis PCA en redes neuronales

Una de las dificultades en cualquier problema es la definición de las variables que lo definen. En el caso de este proyecto, se están obteniendo aceleraciones, velocidades y desplazamiento de diferentes direcciones. Esto provoca que se trabaje con un número elevado de variables y que dificulte el tratamiento de los datos.

En primer lugar, el objetivo principal de la aplicación de análisis PCA a este problema es el de reducir las variables de entrada en la red neuronal sin sacrificar en el proceso la calidad de los resultados. El análisis de componentes principales es la técnica mediante la que se puede reducir la dimensión de un conjunto de datos. Se realiza mediante la proyección con la que mejor se representa los datos, perdiendo la menor información posible.

De forma intuitiva, el análisis PCA determina las componentes principales de la base de datos. Cada componente se obtiene por combinación lineal de las variables originales, pero estas nuevas variables serán independientes entre sí. Además, el análisis PCA se realiza dando mayor peso a donde se produce una mayor variabilidad, o lo que es lo mismo, una mayor variación que será capaz de identificar mejor los patrones.

Una vez realizado este análisis, se tendrá ordenado de mejor a peor las variables predictoras, es decir, las direcciones principales formadas por las antiguas variables, que mejor definen el problema. Como estas variables predictoras, o componentes principales, que son combinación lineal de las antiguas variables y están en orden de definición del problema, se puede elegir el número de variables predictoras para definir el problema sacrificando una cantidad conocida de información, pero eliminando un número de variables considerables:

```
X=dataTrain_NN_4;
[pcaCoefficients, ~, ~, ~, explained, ~] = pca(X);
explainedVarianceToKeepAsFraction = 98/100;
numComponentsToKeep = find(cumsum(explained)/sum(explained) >= explainedVarianceToKeepAsFraction, 1);
pcaCoefficients_2 = pcaCoefficients(:,1:numComponentsToKeep);

for i=1:numComponentsToKeep
    [F,idxi]=max(pcaCoefficients_2(:,i));
    I_2(i)=idxi;
end
```

Ilustración 72: Extracto de programación: PCA

Para realizar este proceso Matlab (Ilustración 72) tiene predefinida una función denominada “pca”, que directamente extrae los autovectores que definen las direcciones principales (“pcaCoefficients”) y una variable (“explained”) que recoge la aportación de cada dirección principal definida mediante el análisis PCA.

Para entender la variable "explained" se muestra la ilustración 73 donde en el eje horizontal se muestra la dirección principal y en el eje vertical se representan dos valores. Representado mediante columnas se muestra el rango de confianza que se puede llegar a obtener mediante esa dirección principal y por otro lado, representado por una línea, se muestra el acumulado del intervalo de confianza. En la gráfica se puede ver como con dos direcciones principales, o nuevas variables, se pueden obtener los mismos resultados con casi un 100% de fiabilidad.

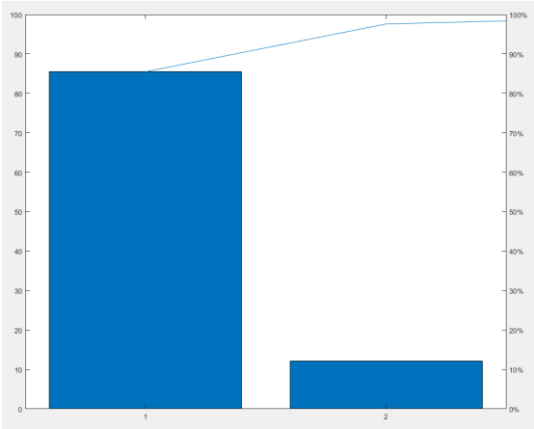


Ilustración 73: Representación "explained"

En el procedimiento seguido en Matlab, se otorga el valor de confiabilidad deseado mediante la variable "explainedVarianceToKeepFraction" y el resto de la programación consiste en determinar el número de direcciones principales, o nuevas variables, que se van a utilizar para entrenar la red neuronal.

Una vez que se determina qué nuevas variables se van a utilizar, hay que transformar la base de datos actual en la base de datos donde se recojan los valores de estas nuevas variables (Ilustración 74). Se multiplica la matriz de datos por la matriz obtenida de los vectores principales que componen la variable "pcaCoefficients", obteniendo la nueva base de datos al proyectar los datos en las direcciones principales.

Tras la obtención de la nueva base de datos solo queda repetir el proceso de creación de una red neuronal pero esta vez, con los datos obtenidos mediante el análisis PCA.

```
dataTest_NN_PCA_4=dataTest_NN_4*pcaCoefficients_2;
dataTrain_NN_PCA_4=dataTrain_NN_4*pcaCoefficients_2;

NN_PCA_4=fitnet(4);
NN_PCA_4.divideParam.trainRatio=70/100;
NN_PCA_4.divideParam.valRatio=15/100;
NN_PCA_4.divideParam.testRatio=15/100;

[NN_PCA_4,~]=train(NN_PCA_4,dataTrain_NN_PCA_4',LLT_Del');
LLT_Del_PRED_NN_PCA_4=NN_PCA_4(dataTest_NN_PCA_4');
```

Ilustración 74: Extracto de programación: transformar base de datos PCA

Realizar este procedimiento conlleva un gran inconveniente, se pierde el sentido físico del problema. Al crear nuevas variables que son combinación lineal de las variables físicas que se han obtenido de la simulación, estas nuevas variables carecen de sentido físico y aunque se haya reducido el número de variables necesarias para entrenar la red neuronal, sigue siendo necesario conocer todas las variables de antemano.

Como solución a este problema, se presenta una solución para no perder el sentido físico. Esta alternativa consiste en estudiar los resultados obtenidos al realizar el análisis PCA y determinar las variables con las que se puede reducir las dimensiones del problema.

Los resultados obtenidos del análisis PCA (Tabla 4), son la matriz de vectores principales que definen la dirección de las direcciones principales de la nueva base ortogonal para los datos y la capacidad de definición del problema, que es capaz de explicar cada dirección al proyectas las variables antiguas. Además, se obtienen los valores que aporta cada dirección principal mediante la variable "explained2":

04_11_2020_2.mlx | pcaCoefficients_ALL = 12×12

	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.0001	-0.0003	0.0012	-0.0011	-0.0290	-0.0003	-0.3067	-0.0965	-0.4160	0.5046	0.6761	-0.1049
2	-0.0002	-0.0036	-0.0386	0.0010	-0.0609	0.0158	0.1934	0.0740	-0.4618	0.5910	-0.6075	0.1416
3	-0.0000	-0.0002	-0.0009	0.0003	-0.0049	-0.0007	0.0351	-0.0025	0.0126	-0.0320	0.1989	0.9788
4	0.0001	0.0002	0.0077	0.0076	0.1403	0.9818	-0.0558	0.1135	0.0146	0.0058	0.0024	0.0032
5	-0.0012	0.0755	-0.0959	0.9892	0.0424	-0.0213	-0.0186	0.0626	-0.0022	-0.0018	0.0040	-0.0002
6	-0.0003	-0.0074	-0.0756	0.0060	-0.1346	0.0607	0.7190	0.0773	-0.4947	-0.3492	0.2743	-0.0871
7	1.0000	0.0028	-0.0042	0.0006	-0.0007	0.0001	-0.0002	-0.0001	-0.0000	0.0000	0.0000	0.0000
8	-0.0026	0.9940	0.0549	-0.0678	-0.0643	0.0088	0.0043	0.0050	0.0006	0.0011	-0.0002	-0.0001
9	0.0010	0.0560	0.0943	-0.0362	0.9683	-0.1279	0.0843	-0.0314	-0.1538	-0.0308	0.0037	0.0020
10	0.0002	0.0004	-0.0010	0.0017	0.0635	-0.0027	0.5586	0.1135	0.5845	0.5208	0.2342	-0.0575
11	0.0002	-0.0067	0.0059	-0.0673	0.0187	-0.1230	-0.1574	0.9739	-0.0411	-0.0295	0.0642	-0.0053
12	0.0042	-0.0540	0.9857	0.1043	-0.0987	0.0080	0.0549	0.0112	-0.0404	-0.0009	-0.0035	-0.0004

```
explained2 =

   94.2551
    5.2131
    0.4073
    0.1197
    0.0046
    0.0002
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

Tabla 4: Matriz de vectores principales y aportación de cada uno (Explained)

Las columnas de la matriz son las direcciones principales representadas por vectores unitarios. Cada fila se identifica con una variable:

1. Aceleración longitudinal
2. Aceleración lateral
3. Aceleración vertical
4. Aceleración de balanceo
5. Aceleración de guiñado
6. Balanceo
7. Guiñado
8. Velocidad longitudinal
9. Velocidad lateral
10. Velocidad vertical
11. Velocidad de balanceo
12. Velocidad de guiñado

Analizando la matriz se puede concluir:

- Hasta la sexta columna, los vectores se definen mayoritariamente por una sola variable. A su vez, analizando el valor de la variable "explained2" se puede decir que con las 5 primeras nuevas direcciones se explica el 100%.

  Por lo tanto, utilizando las 5 variables antiguas que definen los vectores de la dirección principal se podría reducir a la mitad el número de variables de entradas en la red neuronal sin perder información significativamente.

- Las variables que se obtienen del análisis PCA que deberían ser capaces de definir el problema son:
    - Guiñado
    - Velocidad longitudinal
    - Velocidad de guiñado
    - Aceleración de guiñado
    - Velocidad Lateral

- Se analizará mediante ensayos la validación de esta hipótesis y se compararan los resultados con las variables que se definen en (García Guzmán et al., 2018)
- Al realizar el análisis estadístico PCA también se llegó a la conclusión de que ciertas variables extraídas de la simulación tan solo aportaban ruido. Se determina que las variables de desplazamientos que se utilizaban en los primeros apartados de la aplicación del Machine Learning al vuelco no aportaban información para calcular el índice del vuelco. Sin embargo, aportaban un gran ruido al realizar los ensayos con PCA porque eran variables que tenían una gran variabilidad, debido a que su referencia era el punto inicial en el espacio y no el vehículo. En los ensayos posteriores estas variables serán eliminadas de las variables de entradas de las redes neuronales, pasando a tener un máximo de 12 variables en vez de 16.

4.2.4 Ensayos con redes neuronales

El objetivo de este apartado consiste en poner en práctica lo analizado en el punto 4 de este proyecto. Se realizarán ensayos con redes neuronales donde se estudiarán las diferentes opciones que se presentan con el número de variables que se han tratado, las diferentes maniobras y principalmente se buscará la comparación de los resultados con las variables que se utilizan en el artículo (García Guzmán et al., 2018)

Antes de continuar con el desarrollo de estos temas, se va a seleccionar un nuevo vehículo para las simulaciones. El nuevo vehículo reemplazará al vehículo deportivo para favorecer que los cambios que se produzcan en la transferencia de carga lateral (LLT) se hagan más visible.

El deportivo, al ser un vehículo con un centro de gravedad muy bajo, un ancho de vías considerablemente elevado y una amortiguación rígida provoca que los desplazamientos laterales, incluyendo el balanceo y el guiñado tengan variaciones muy pequeñas en comparación a la variación de fuerza entre ambos lados del vehículo.

La opción que se ha elegido es la de un vehículo tipo crossover (ilustración 75). Este tipo de vehículos son más pesados, altos y con suspensiones más blandas, provocando el efecto contrario al vehículo deportivo. Se favorece la transferencia de carga de un lado a otro, acompañado de desplazamientos y giros más pronunciados.

Ilustración 75: Vehículo Crossover

Por otro lado, a la hora de las simulaciones en el software Adams Cars, se recogen datos de tres tipos de maniobras, denominadas en el simulador como “Drift” “ISO Lane Change” y “Fish hook” (Ilustración 76). La elección de estas tres maniobras se realiza en primer lugar porque son maniobras donde se producen varias transferencias de carga entre los dos lados del vehículo. También se seleccionan estas maniobras porque son fácilmente programables en el software a la vez que conllevan una complejidad de giros y cambios de trayectorias que harán variar el índice de transferencia de carga acorde a lo que se quiere analizar.

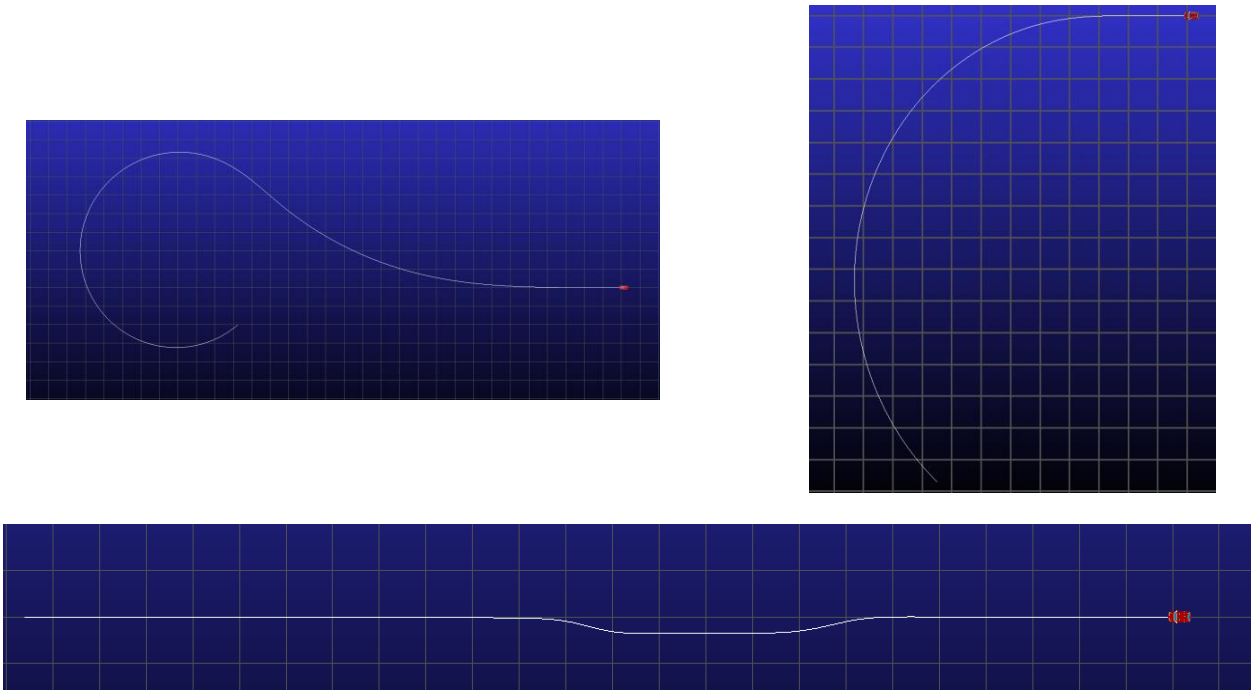


Ilustración 76 : Maniobras “Fish hook” “Drift” e “ISO Lane Change”

Mientras que estos datos serán utilizados para el entrenamiento de la red neuronal también será necesario la toma de datos para realizar la verificación y prueba de la predicción que ofrece la red neuronal. Para que los resultados tengan validez no se deben utilizar trazadas que se utilicen para el entrenamiento ya que un determinado overfitting podría devolver resultados erróneos.

Se selecciona un circuito completo (Ilustración 77), donde el vehículo describe diferentes trayectorias, desde partes rectas y curvas rápidas a zonas donde el trazado se vuelve más enrevesado y provoca zonas de muy baja velocidad y giros lentos. Se están generando diferentes situaciones donde se podrá valorar el funcionamiento de la red neuronal y valorar los resultados con mayor validez.

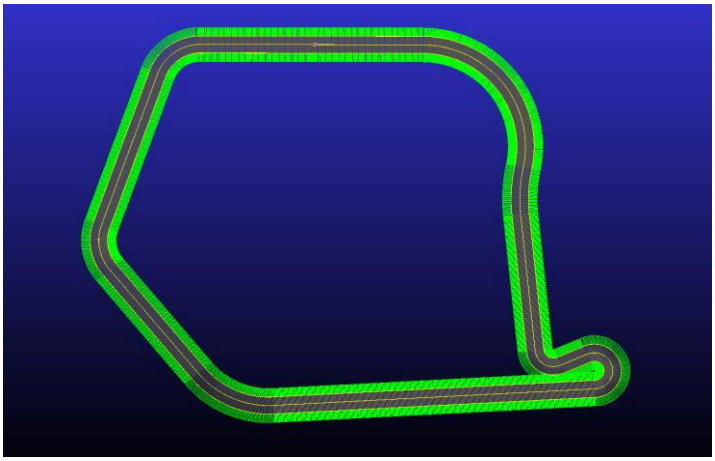


Ilustración 77: Circuito para datos de test

El ensayo con estas tres maniobras se realiza en tres fases.

- 1º Fase: Ensayos a velocidad baja.
- 2º Fase: Ensayos a velocidad alta
- 3º Fase: Se utilizará todo el conjunto de datos, a velocidades altas y bajas.

El objetivo de dividir el ensayo en tres partes es el de estudiar la respuesta ante las diferentes entradas de la red neuronal y del programa desarrollado para el ensayo. Para estudiar la respuesta ante los diferentes tipos de datos, la comparación se centrará en el estudio del error absoluto medio.

Al variar la velocidad de los ensayos, no solo se modifican las fuerzas y aceleraciones que experimenta el vehículo, sino que se consigue una reducción de datos debido a que se realiza la misma maniobra con la misma distancia recorrida modificando la velocidad. Se consigue variar también la cantidad de datos que se introducen en la red.

Además de esta división, el objetivo es la comparación entre la elección de las variables. Se va a utilizar como entrada las 12 variables que se obtienen de la simulación y que se han descrito anteriormente. A este grupo de variables se le aplicara el análisis PCA y se determinan 5 variables con las que se entrena otra red.

Por último, se utilizarán las variables descritas en (García Guzmán et al., 2018). Se persigue la comparación entre estas tres disposiciones de entradas de datos para obtener el mejor resultado con el menor número de variables posibles.

Teóricamente se deberían obtener mejores resultados con la red entrenada mediante 12 entradas o variables. Al tener una cantidad de información mayor, el error debería ser menor que en los otros dos casos. En contraposición, al introducir un número mayor de variables se aumenta el tiempo de computación a la vez que se aumenta la dificultad de llevar a cabo físicamente este proyecto, además de correr un mayor riesgo de overfitting. De todas formas, y a modo de comparación con el resultado de las demás redes neuronales, se estudiarán los resultados y se analizarán en diferentes situaciones.

4.2.4.1 Ensayos a velocidad baja

Se simulan las tres maniobras seleccionadas a una velocidad de 40 km/h. Se obtienen tres conjuntos de datos que posteriormente se convertirán en la base de datos para entrenar la red neuronal.

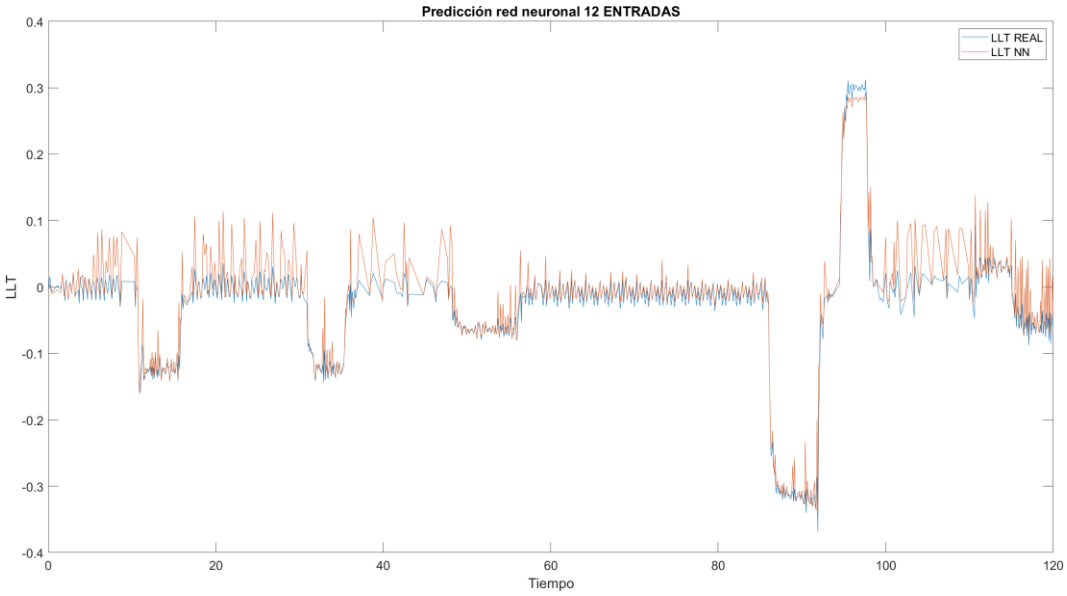


Ilustración 78: Predicción con 12 entradas a velocidad baja

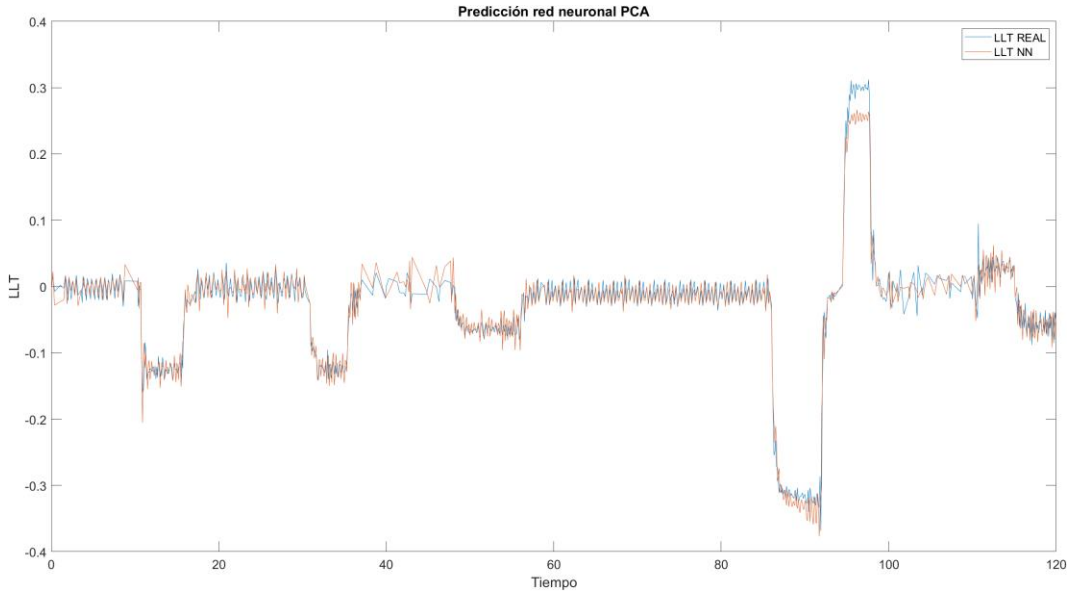


Ilustración 79: Predicción aplicando PCA a velocidad baja

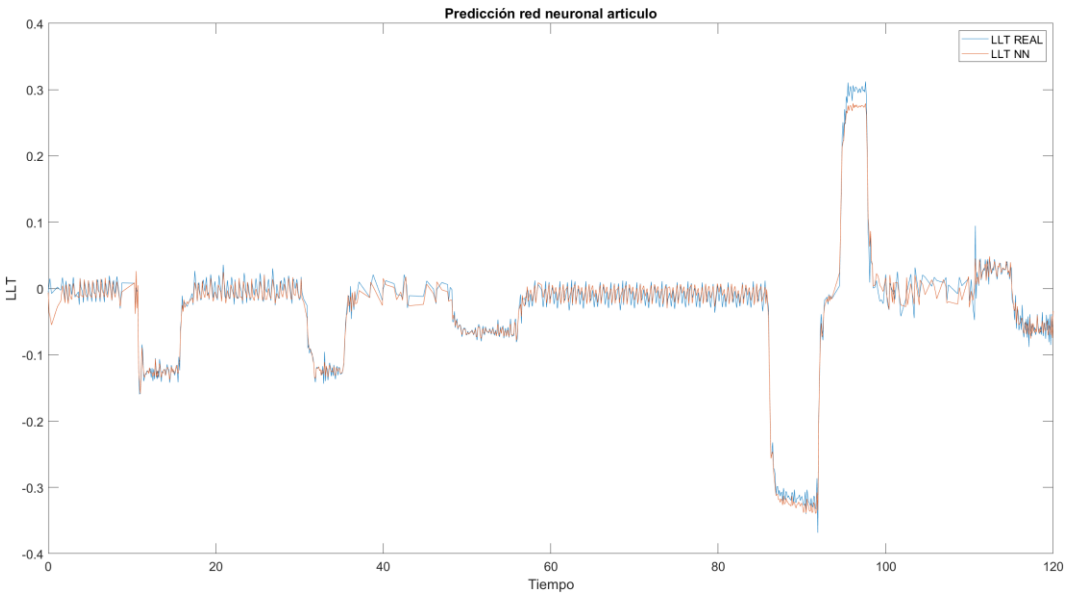


Ilustración 80: Predicción con variables del artículo guía a velocidad baja

	Numero de neuronas	Error absoluto medio
12 entradas	2	0.0048
Aplicando PCA	3	0.0128
Articulo	3	0.0076

Tabla 5 : Error absoluto medio a baja velocidad

El resultado obtenido con la introducción de datos a baja velocidad muestra:

- El error absoluto medido en los diferentes escenarios es muy bajo, pero presenta tres rangos (Tabla 5). En primer lugar, se tiene el resultado obtenido mediante el entrenamiento de la red neuronal con 12 entradas que, como ya se anticipaba, es el que menor error medio tiene. En segundo lugar, se tiene los resultados obtenidos con las variables que se muestran en el artículo y por último el resultado obtenido mediante el análisis PCA propuesto en este trabajo. La diferencia entre rangos es, aunque en valor muy pequeña, casi el doble entre unos ensayos y otros.

  Otro dato significativo es que, aunque se han utilizado un número mayor de entradas en el primer caso, se ha obtenido un mejor resultado con dos neuronas en la capa oculta, siendo 3 en los otros dos.

- En la ilustración 78,79 y 80 se muestra el resultado de la predicción con 12 entradas, con la aplicación del análisis PCA y con las variables obtenidas del articulo respectivamente. En la ilustración 78 se muestran los resultados con un menor error medio a pesar de las irregularidades que se muestran gráficamente. Estas irregularidades se acentúan al inicio, entre los segundos 0 y 40, y al final de la simulación, entre los valores de 100 y 120 segundos. En cambio, en la zona donde se produce una transferencia de carga lateral mayor, que se sitúa en la curva de 180º del circuito, la predicción es muy buena y se comenten muy pocas variaciones respecto a la realidad.

  En la ilustración 79, utilizando las variables obtenidas del análisis PCA previo se reducen la diferencia e irregularidades en la mayor parte del ensayo, pero el resultado final es el error medio más alto. Esto es debido a que gráficamente en las zonas donde se producen pocas variaciones las curvas son similares porque muestran menos irregularidad o picos, pero generalmente el error es mayor. Además, en la zona más revirada del circuito se produce un desajuste pronunciado al finalizar la curva. Aun así, el resultado es bastante satisfactorio y representa bien la realidad.

Por último, en la ilustración 80 se muestra los resultados de la red entrenada con las variables de entrada que se muestran en el artículo. Gráficamente la respuesta es más continua y se copia la curva real en todo el ejemplo. En las zonas con menos transferencia de carga la curva se vuelve menos irregular al igual que en la zona más revirada.

- Teniendo en cuenta que el número de entradas para el caso con menor error, 12 entradas, es más del doble que las variables utilizadas en el artículo y que el aumento del error es menor del doble, se determina que la mejor solución en este caso sería optar por las variables que se utilizan en el artículo. Se obtiene unos resultados lo suficientemente buenos en la mayoría de casos con un error mínimo.
- El tiempo computacional es superior en el primer caso, con 12 entradas, aunque la mayor parte del tiempo es imputado a la creación de la curva de aprendizaje. Además, los mayores tiempos de computación se producen en los cálculos con un número elevado de neuronas en la capa oculta, los cuales no son representativos ya que la solución óptima son 2-3 neuronas en la capa oculta.

4.2.4.2 Ensayos a velocidad alta

Tras la simulación a una velocidad baja se simulan las tres maniobras, en este caso a una velocidad de 100 km/h. Al igual que en el apartado anterior se unen los tres conjuntos de datos en la base de datos que servirá para entrenar la red neuronal.

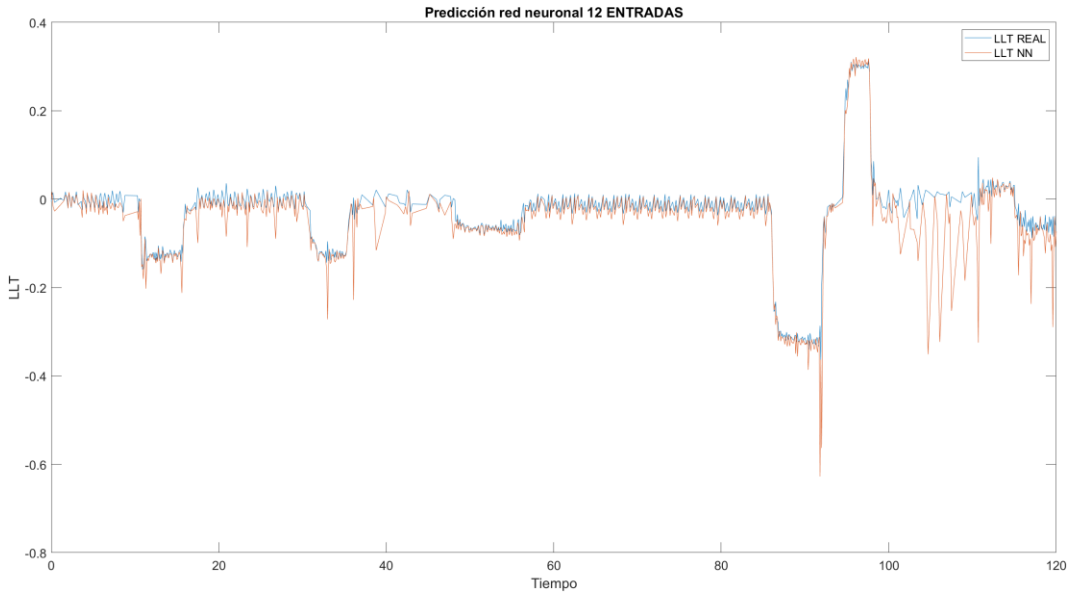


Ilustración 81: Predicción con 12 entradas a velocidad alta

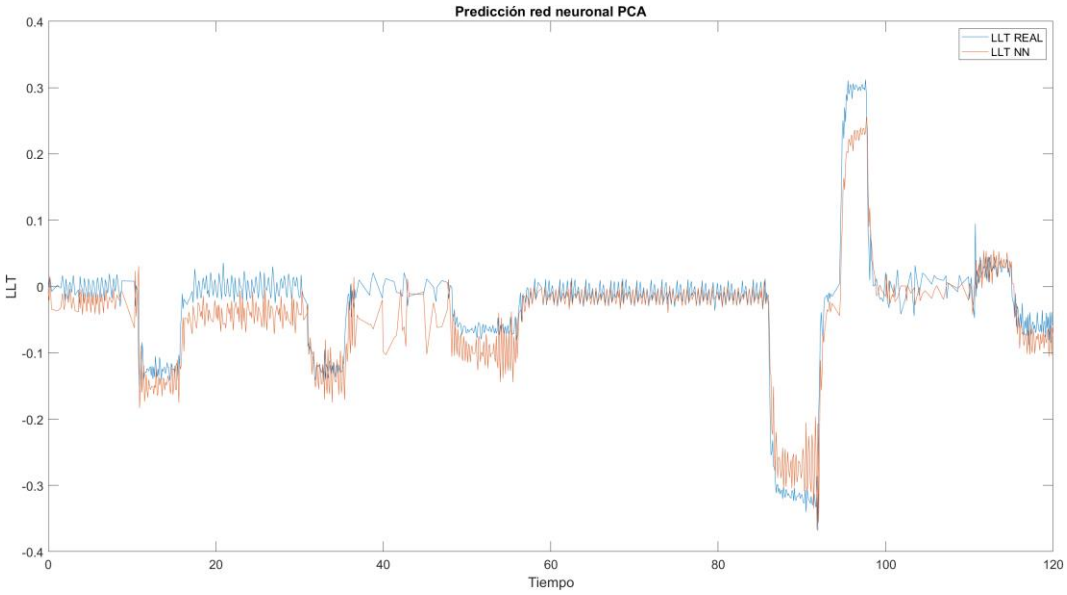


Ilustración 82: Predicción aplicando PCA a velocidad alta

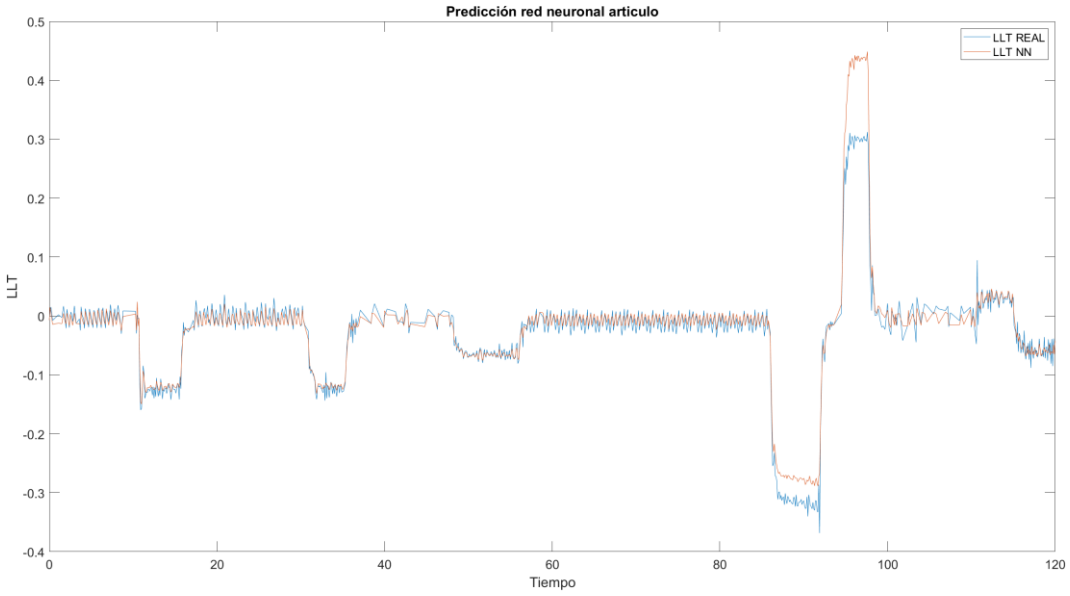


Ilustración 83: Predicción con variables del artículo guía a velocidad alta

	Numero de neuronas	Error absoluto medio
12 entradas	3	0.0035
Aplicando PCA	3	0.0219
Articulo	3	0.0083

Tabla 6: Error absoluto medio a alta velocidad

En este segundo ensayo donde se utiliza una velocidad más elevada los resultados muestran:

- El número de neuronas en la capa oculta es el mismo para cada forma de entrenar la red (Tabla 6).Al igual que con las predicciones a baja velocidad el error absoluto es mayor en el caso donde se han utilizado las variables obtenidas del análisis PCA y menor en el caso que se utilizan 12 entradas para entrenar la red. De nuevo, el punto intermedio es la red entrenada mediante las variables del artículo. En este caso, las diferencias son más notables siendo los escalones entre unos valores y otros superiores a dos veces el valor del anterior.
- En la ilustración 81, que representa la predicción utilizando una red entrenada con 12 entradas, se produce un fenómeno similar al ejemplo anterior. Se producen saltos en los valores, pero está vez es mucho más significativo. En este caso el error medio, aunque se ha reducido, se puede apreciar gráficamente que los errores puntuales difieren más de un 50% de la realidad. Sin embargo, exceptuando los casos puntuales donde los valores difieren tanto, el resto se acerca más a la realidad.
- En la ilustración 82, se representas los resultados de las variables obtenidas mediante el análisis PCA. En este caso los valores son más continuos pero el error medio que presenta es el más alto de todos. Se comprueba como en las zonas más rápidas y con menor transferencia de carga la predicción es muy exacta, pero en el resto de la simulación el error es evidente.
- Por último, en la ilustración 83 se muestran los resultados utilizando las variables del artículo. Aunque en la gran parte de la simulación la predicción se acerca muchísimo a la realidad, en la zona más revirada se producen predicciones erróneas, haciendo que el error medio absoluto crezca. A pesar de estos errores, el resto de la simulación es la más continua y estable de todas.

4.2.4.3 Unión de todos los datos

Por último, se une el conjunto de datos para entrenar las redes neuronales. A priori, este ensayo debería mostrar los mejores resultados ya que ofrece una variedad de datos mayor, aportando datos de diferentes situaciones a diferentes velocidades. Sin embargo, se corre el riesgo de que estas redes presenten el fenómeno de overfitting.

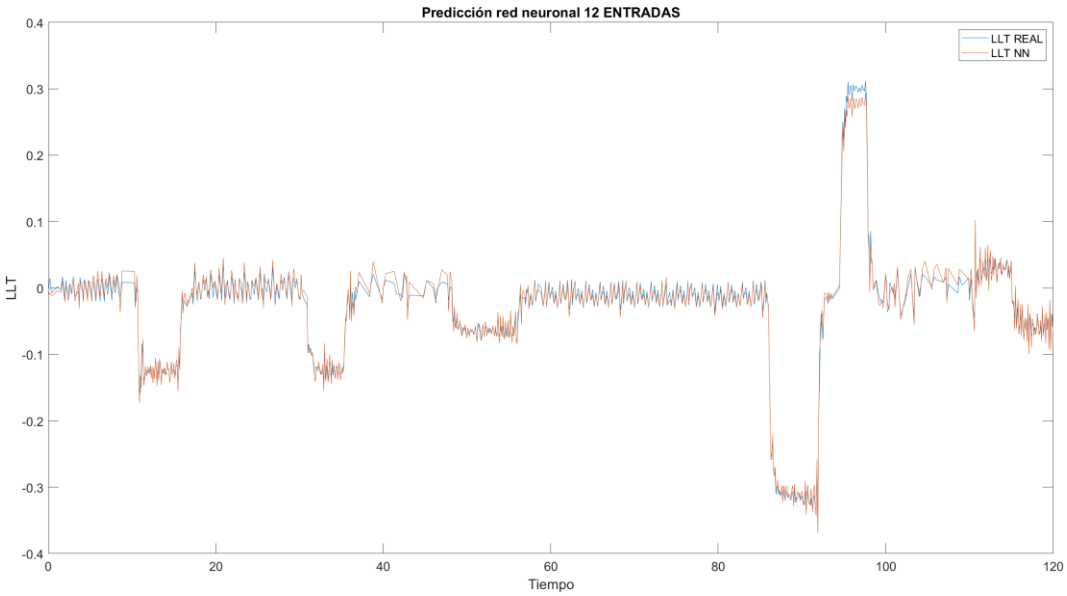


Ilustración 84: Predicción con 12 entradas con todos los datos

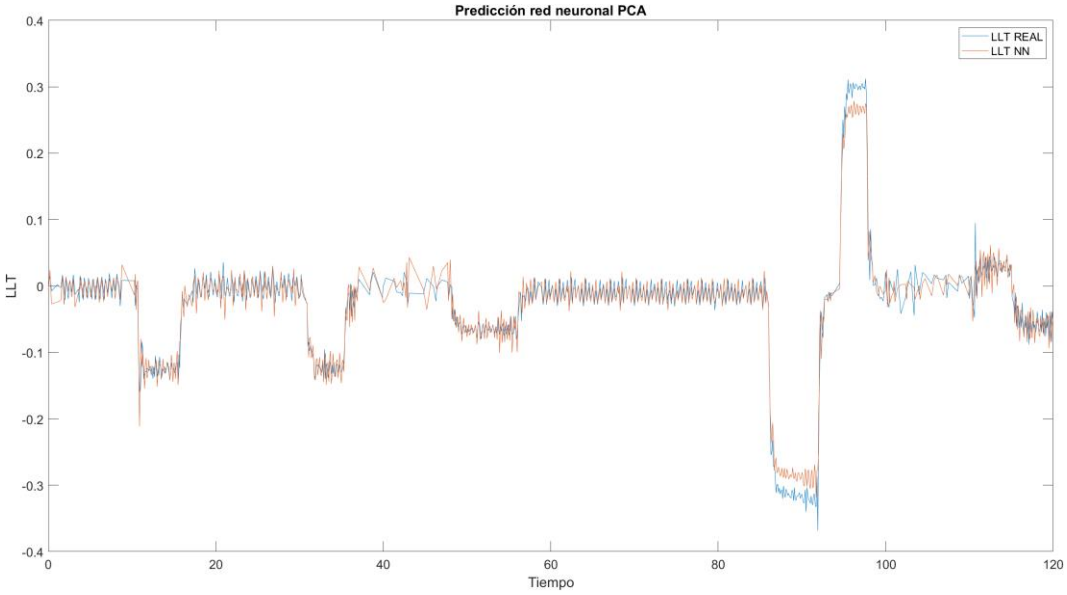


Ilustración 85: Predicción aplicando PCA con todos los datos

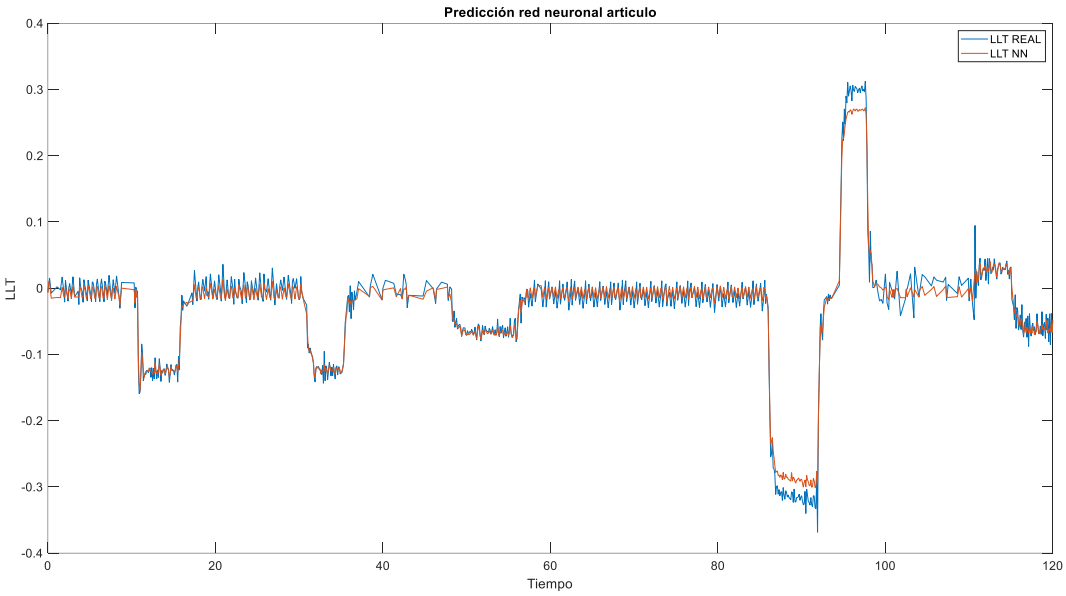


Ilustración 86: Predicción con variables del artículo guía con todos los datos

	Numero de neuronas	Error absoluto medio
12 entradas	1	0.0074
Aplicando PCA	6	0.0123
Articulo	4	0.0082

Tabla 7: Error absoluto medio uniendo los datos a baja y alta velocidad.

El resultado obtenido utilizando todo el conjunto de datos muestra:

- El error absoluto medio, en contra de lo esperado, ha aumentado en el entrenamiento de la red con 12 entradas y cuando se utilizan las variables del artículo. Sin embargo, en el caso de las variables que se han obtenido mediante el análisis previo de PCA se ha reducido el error, aunque de forma poco significativa.
- El número de neuronas en la capa oculta aumenta en dos de los tres tipos de entrenamiento, lo que se corresponde con el mayor número de datos introducidos en la red. Sin embargo, en el caso donde más datos se introducen, la red de 12 entradas, se produce el caso contrario. Aumenta sustancialmente el error en comparación con los ejemplos anteriores con las mismas entradas, lo que podría indicar un overfitting al introducir un número muy grande de datos.
- A pesar de los resultados del error absoluto medio, en la ilustración 84 se muestra la predicción de la red entrenada con 12 entradas y el resultado es muy bueno. Los saltos en algunas zonas se han disminuido en comparación con otros ejemplos con las mismas entradas y la respuesta en la zona más revirada del circuito es muy buena.
- En la ilustración 85 se muestra la predicción realizada por la red entrenada con variables obtenidas del análisis PCA. Se producen algunos saltos en los resultados, pero la curva se aproxima mucho a la realidad. En la zona donde se produce una mayor transferencia de carga es donde mayor error continuo se produce siendo más significativo que en el caso de 12 entradas.
- En la ilustración 86 se muestra la predicción utilizando las variables que presenta el articulo guía y los resultados son muy parecidos a cuando se utilizan las variables que presenta el análisis PCA. Lo que reduce el error en este caso es la mayor continuidad en la curva, sin tantas variaciones en el conjunto completo.

4.2.4.4 Conclusiones

En términos de valor de error absoluto medio, los mejores resultados se han obtenido al introducir tan solo los datos a una mayor velocidad. En cambio, también se han obtenido los peores resultados en función de la forma en la que se entrenaran las redes. Los mejores resultados en general de error medio se obtienen en el primer y último caso donde el error absoluto medio promedio en los tres ensayos está por debajo del 1%

Sin embargo, gráficamente el resultado no ha sido exactamente el mismo. Los resultados mejoran en el último ensayo donde se han utilizado el conjunto total de datos que se tiene. Con estos datos, las gráficas son más estables a la vez que copian mejor la curva real. En concreto, utilizando las variables del articulo para entrenar la red se obtienen los mejores resultados generales.

El valor absoluto medio no define totalmente en estos casos la realidad gráfica de la proximidad de la predicción realizada mediante una red neuronal y la realidad. Teóricamente se sabe que este valor si debe definir la calidad de la predicción, por lo que variaciones excesivas en las curvas en zonas delimitadas pueden estar introduciendo variaciones en el error absoluto medio.

De hecho, este fenómeno se produce y se ha expuesto en los ejemplos anteriores. Donde en la zona más revirada del circuito se producían diferencias muy sustanciales en comparación con el resto de la simulación. Este fenómeno podría ser explicado a su vez con otra hipótesis:

Las simulaciones que se han realizado son relativamente extensas en duración. Para la obtención de datos con la menor cantidad de ruido y oscilaciones se han utilizado tiempos y espacios de simulación lo suficientemente grandes para evitar este efecto. Sin embargo, solucionar ese problema ha podido generar que los datos introducidos tengan pocas variaciones en la transferencia de carga lateral en un periodo corto de tiempo, produciendo un sobreajuste para las zonas con variaciones de transferencia de carga lateral en un periodo largo.

El ejemplo se muestra en la mayoría de las ilustraciones entre la 84 y la 90 en la zona revirada del circuito. Esta zona se caracteriza por un giro de 180 grados seguido de un giro de 90 en un espacio corto, donde se producen las mayores transferencias de carga en un tiempo limitado.

La forma de demostrar dichas hipótesis conjuntas es realizar las mismas simulaciones con las mismas maniobras, pero en tiempos y espacios más reducidos. Aunque también se podrían incluir maniobras con las características de la zona con mayor transferencia de carga lateral, produciría un aumento en la cantidad de datos totales pudiendo a su vez provocar overfitting.

Por lo tanto, se realizarán simulaciones con recorridos más reducidos para favorecer los giros y cambios en la transferencia de carga de forma más brusca, buscando obtener una relación más directa entre los resultados gráficos y los resultados que presenta el error absoluto medio.

5 ENSAYOS CORREGIDOS CON REDES NEURONALES

En este apartado se va a presentar el análisis de las hipótesis que se han formulado en el apartado anterior. El resultado obtenido del cálculo de la transferencia de carga lateral de un vehículo mediante redes neuronales, aunque satisfactorio, es mejorable. Mediante la comprobación de estas hipótesis se intentará mejorar dicho resultado.

Para probar ambas hipótesis se realizarán nuevas simulaciones de las mismas maniobras. Las maniobras utilizadas recogen un conjunto de elementos suficientes para abarcar un amplio abanico de posibilidades y situaciones donde se puedan producir transferencias de carga lateral significativa. Sin embargo, la forma es la que se realizan estas simulaciones pueden variar el resultado obtenido al utilizar redes neuronales.

Se procede a realizar las nuevas maniobras con aproximadamente los mismos ángulos de giro reduciendo la distancia recorrida. Se favorecen así los intercambios de carga entre el lado derecho e izquierdo del vehículo y el aumento de estas variaciones.

Para la simulación de la maniobra denominada “Fish hook” se mantienen los ángulos de giro variando el periodo de tiempo en el que el vehículo debe realizando. Se muestra en la ilustración 87 el acortamiento de la maniobra en espacio. A la izquierda se muestra la nueva maniobra y a la derecha la maniobra realizada para la obtención de los datos en el apartado anterior. Además, se reduce el tiempo de la simulación porque la velocidad del vehículo se va a mantener en 40 km/h y 100 km/h para las simulaciones.

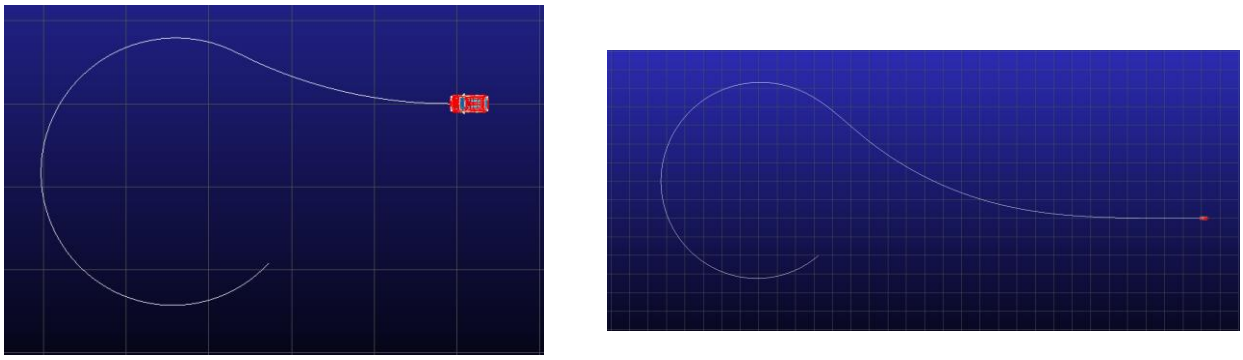


Ilustración 87: Maniobra Fish Hook ajustada

En el caso de la maniobra denominada “Drift” se modifica la rampa de aceleración para provocar que el vehículo realice el giro en un tiempo más reducido. También se modifica el ángulo de giro transcurridos 5 segundos para obligar a que el radio de giro sea menor y provocar un giro más cerrado. En la ilustración 88 se muestra la diferencia entre el formato de la maniobra usada anteriormente, posicionada en el lado derecho y la maniobra que se usará en este apartado, en la parte izquierda.

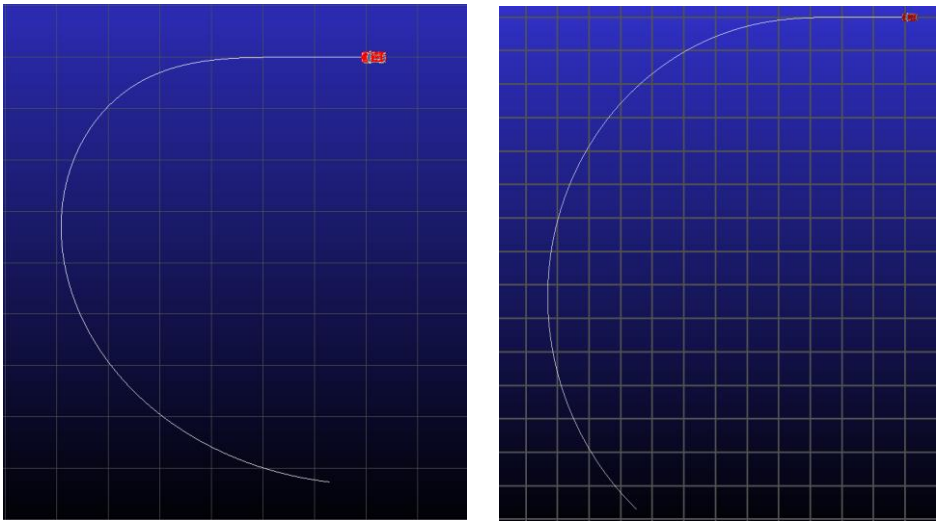


Ilustración 88: Maniobra Drift ajustada

Por último, habría que modificar la maniobra denominada "ISO Lane change". Está maniobra se simula mediante la introducción de un solo parámetro que es la velocidad inicial. La modificación de la velocidad no será un factor determinante porque ya se están introduciendo la misma maniobra a velocidad alta y baja y variar un porcentaje este rango no produciría un cambio en la dirección deseada.

Se decide continuar, para la maniobra del cambio de carril, con los datos que ya se habían obtenido anteriormente y modificar los datos de las otras dos, Fish hook y Drift.

Tras la obtención de los nuevos datos se introducen en el programa creado en Matlab para realizar el mismo proceso que se ha realizado anteriormente en el apartado 4, para obtener resultados entrenando la red neuronal de diferente forma y comparar los resultados.

5.1 Resultados

A continuación, se exponen los resultados obtenidos de entrenar tres redes neuronales con los datos obtenidos de tres maniobras diferentes con una velocidad de 40 km/h y 100km/h cada una. Se obtienen 6 archivos que conjuntamente forman la base de datos que se utilizará posteriormente para el entrenamiento de las redes.

Como se ha realizado anteriormente, se entrenarán con estos datos tres redes neuronales variando el número total de entradas y las variables asignadas a dichas entradas. El primer caso corresponde a la introducción de 12 entradas, incluyendo velocidades y aceleraciones que comprenden todos los ejes del vehículo y desplazamientos de balanceo y cabeceo. El segundo, como anteriormente, corresponde a la red neuronal que se entrenara con las variables obtenidas del análisis PCA que son: guiñado, velocidad longitudinal, velocidad de guiñado, aceleración de guiñado, velocidad lateral, aceleración de balanceo. El último caso corresponde a la red neuronal con las variables que se utilizan del artículo: aceleración longitudinal, aceleración lateral, velocidad de guiñado y velocidad de balanceo.

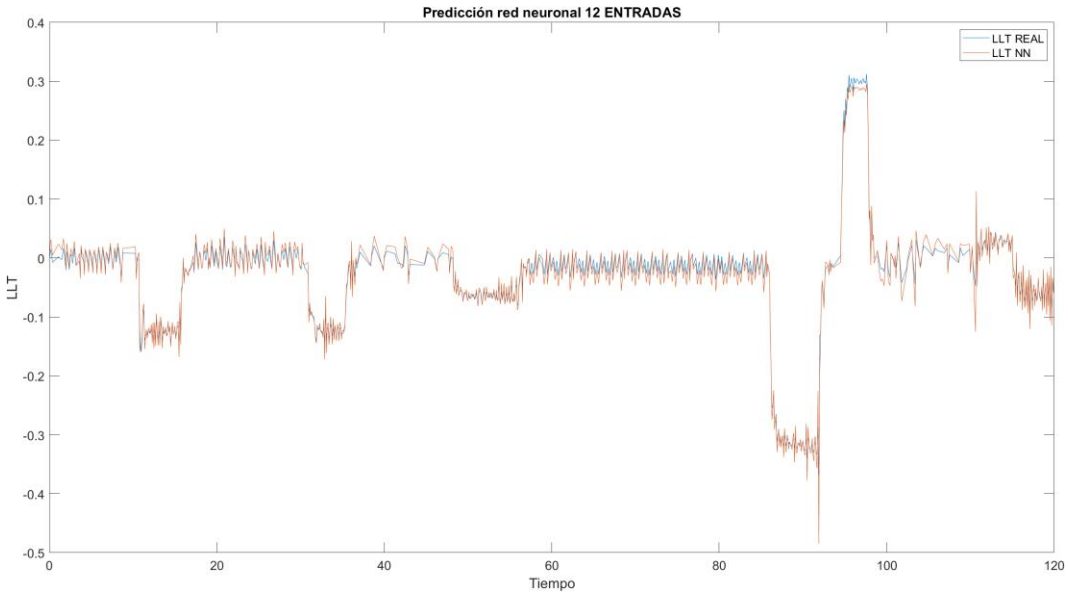


Ilustración 89: Predicción con datos corregidos con red neuronal de 12 entradas

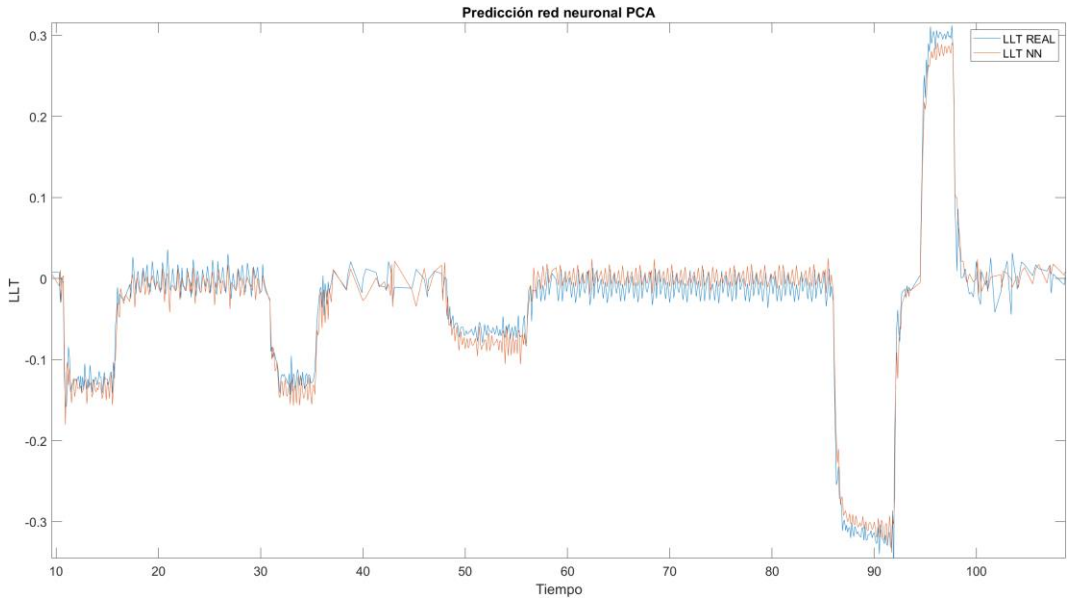


Ilustración 90: Predicción con datos corregidos con red neuronal utilizando análisis PCA

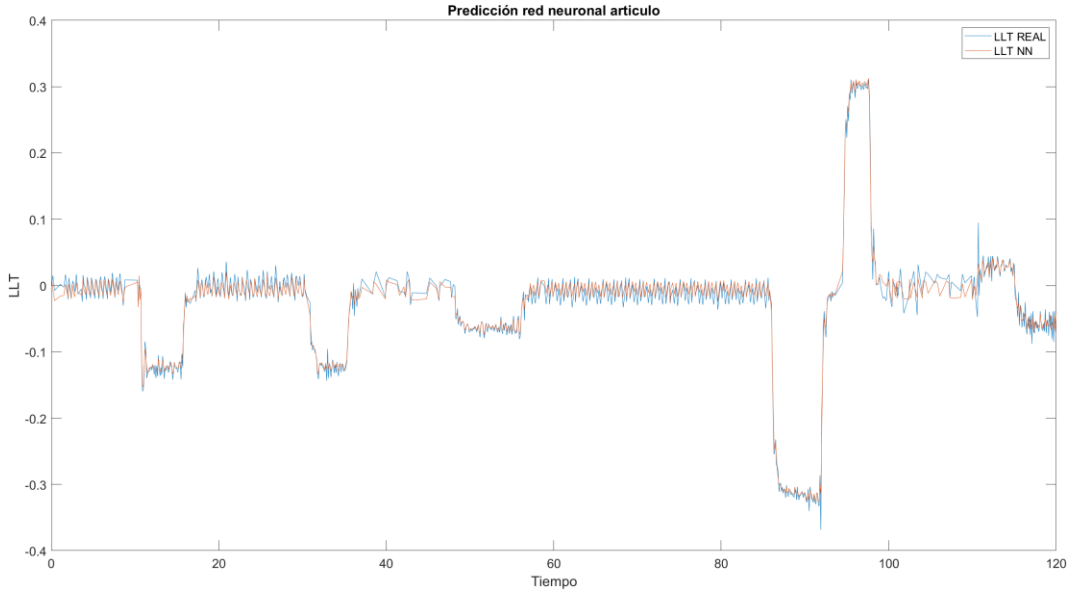


Ilustración 91: Predicción con datos corregidos con red neuronal entrenada mediante variables del artículo.

	Numero de neuronas	Error absoluto medio
12 entradas	3	0.0064
Aplicando PCA	7	0.0131
Articulo	2	0.0072

Tabla 8: Error absoluto medio con ensayo corregido

El resultado obtenido utilizando maniobras más bruscas en su conjunto muestra:

- El error absoluto medio del conjunto de ensayos se ha reducido de forma general (Tabla 8), especialmente en el caso donde se han utilizado las variables de entradas que se utilizan en el artículo. El mínimo error absoluto medio se produce en la red neuronal con un mayor número de datos de entrada, obteniendo un resultado de 0,0064. Sin embargo, el error que se produce en la red entrenada mediante las variables obtenidas del análisis PCA empeora.
- El número de neuronas en la capa oculta ha aumentado de forma general, llegando a 7 neuronas en el segundo caso, donde el error se hace mayor. En el primer y último caso, se utilizan 3 y 2 neuronas, respectivamente.
- Gráficamente se evidencia la mejora con respecto a la simulación anterior. Aunque el error ha disminuido, gráficamente se puede ver como los problemas que se tenían con la base de datos anterior se han solucionado en gran medida. En la ilustración 91 se muestra como la red entrenada con 5 variables de entradas definidas en el artículo, se ajusta en todo momento a la curva real. En la zona más revirada del circuito, donde se producía anteriormente mayor error continuo, ahora tenemos en todos los casos una curva mucho más ajustada y con mayor continuidad.

  En las ilustraciones 89 y 90 también se muestra la mejora que se ha producido al introducir datos de otras maniobras. Se mejora la repuesta global y además se mejora en la continuidad de la curva. En la ilustración 89, donde se introducen un mayor número de variables, persisten las discontinuidades y saltos que se experimentan anteriormente, pero de forma general el resultado ha mejorado.

5.2 Conclusiones

A pesar de disminuir el tiempo y el espacio, y por tanto el número total de datos, se ha favorecido la transferencia de carga entre ambos lados del vehículo. Se concluye entonces que los sistemas de redes neuronales funcionan mejor.

En el caso donde se utilizan las cuatro variables descritas en el artículo, que es el caso más regular en la mayoría de los ensayos, el error se reduce en un gran porcentaje. También se produce el mismo efecto con 12 entradas y aunque ocurre lo contrario en el caso de las variables obtenidas mediante PCA, la media general mejora.

De forma gráfica, se produce también una mejora en las representaciones por lo que se puede determinar que en términos generales realizar ensayos de esta manera mejora el rendimiento de las redes neuronales.

En los siguientes apartados se continuará utilizando tanto los datos obtenidos en este ejemplo como con la forma de realizar las simulaciones.

6 GENERALIZACIÓN DEL PROBLEMA

El núcleo de este proyecto se centra en demostrar la efectividad y la posibilidad de aplicar Machine Learning al estudio del vuelco de vehículo. Se ha utilizado un vehículo genérico con un centro de gravedad considerablemente alto para propiciar las transferencias de carga lateral y así, favorecer el análisis y la visualización del estudio. Sin embargo, uno de los objetivos de aplicar Machine Learning al vuelco de vehículos es también la posibilidad de generalizar el problema.

En los artículos y documentos que se han estudiado durante la realización del proyecto, al igual que en la realización del proyecto fin de grado previo, se utilizaban vehículos determinados, con geometrías conocidas o incluso definidas mediante la introducción manual de variables. Uno de los ejemplos más claros es el dispositivo Inclisafe, del cual nacen ambos proyectos. A este dispositivo era necesario introducir variables como la inercia de los vehículos y su geometría, teniendo en cuenta los accesorios extras que se le incluían a los vehículos agrícolas, como son los arados o excavadoras.

No ser capaz de generalizar el problema genera un déficit enorme en el potencial para usar este tipo de tecnología en vehículos. Se tendría que particularizar las redes neuronales con cada vehículo y con cada tipo de disposición del mismo.

En este apartado se realizará un análisis sobre la viabilidad de usar redes neuronales para generalizar el problema. Se medirán los errores que se producen al utilizar las redes creadas para otro tipo de vehículos, donde la altura del centro de gravedad y el ancho de vía varían. Por último, tras obtener las diferencias entre los diferentes vehículos se presentan dos métodos por los que se trata de generalizar el problema.

6.1 Resultados utilizando la red neuronal creada.

En primer lugar, se utilizarán las redes neuronales que se han creado mediante el proceso descrito anteriormente, específicamente las desarrolladas en el apartado 5.

En este proceso se entrenará la red neuronal con los datos del vehículo denominado Crossover. Una vez entrenada la red, se utilizará el circuito empleado para el test anteriormente, pero esta vez con vehículos diferentes. Se utilizará un vehículo tipo sedán, un vehículo tipo pick-up y un autobús.

Como los mejores resultados se han obtenido al utilizar las variables que utilizan en el artículo, se utilizara el resultado ofrecido por la red entrenada con dichas variables.

Las especificaciones geométricas se muestran en la tabla 9:

Parámetro / Vehículo	*Sedán*	*Crossover*	*Pick-Up*	*Autobús*
Longitud (mm)	4700	4680	5100	14300
Ancho de vías	1800(-15%)	2100	1800(-15%)	2600(+24%)
Altura cdg	500 (-9%)	550	600(+9%)	1100(+100%)
Masa (kg)	850	850	1700	10300

Tabla 9: Especificaciones geométricas de los vehículos

A priori, los resultados que se van a obtener, aunque similares, no deben ser unos buenos resultados. Con la relación que existe entre la variación de la masa, del ancho de vías y la altura del centro de gravedad con el vuelco no se deberían obtener buenas aproximaciones. Excepto en el caso del sedán que, además de tener la misma masa total, solo tienen una variación de la altura y el ancho de vías son de un 15 % y un 9%, respectivamente.

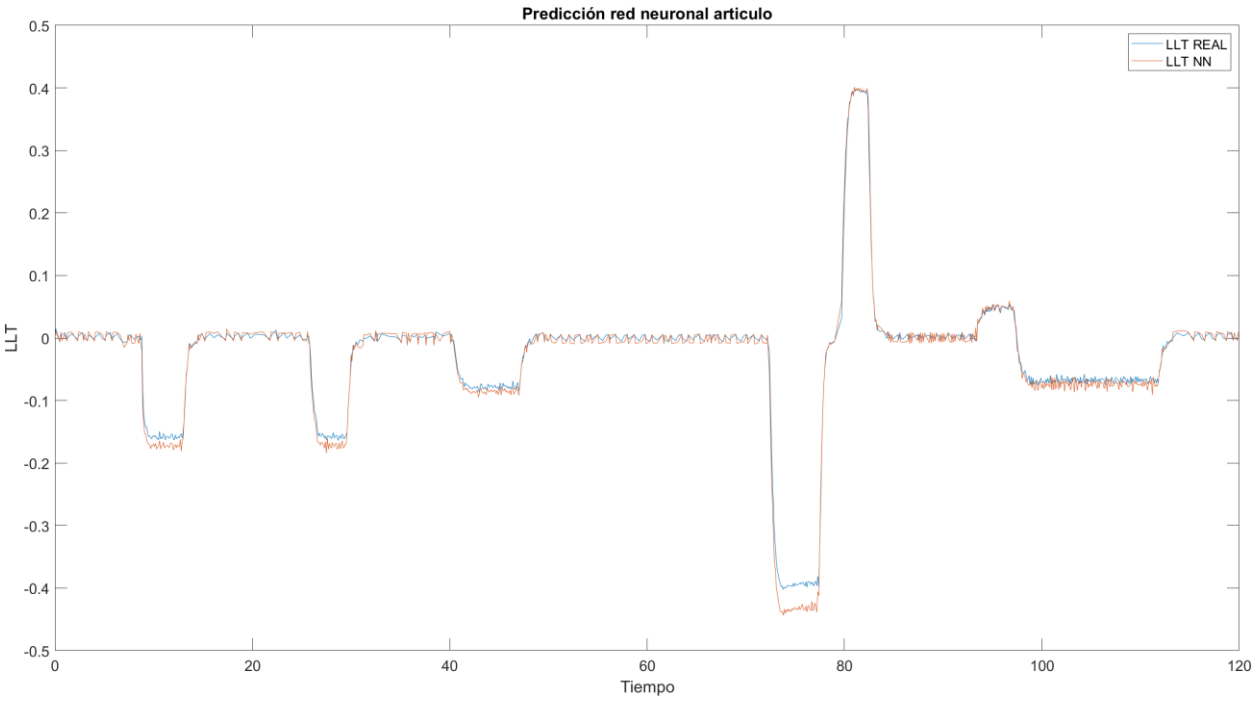


Ilustración 92: Predicción LLT sedán con red neuronal del Crossover

A pesar de la variación de la altura del centro de gravedad y del ancho de vías, los resultados que se obtienen son muy satisfactorios. Se puede apreciar en la ilustración 92 como la predicción realizada por la red neuronal es prácticamente idéntica en la mayoría de la simulación, salvo por un instante que corresponde a la primera curva en la zona revirada del circuito, donde se produce una variación entre los resultados reales y la predicción. Además, el error absoluto medio de la simulación tiene un valor de 0.0103, siendo un valor muy semejante a los obtenidos para el vehículo de los propios datos con los que se ha entrenado la red.

En cambio, en el caso de la predicción mediante la red neuronal entrenada con los datos del crossover y utilizando para la predicción los datos de una pickup, el resultado obtenido se acerca a la realidad acumulando un error sustancial durante la simulación. En la zona revirada (ilustración 93) el error absoluto en los puntos de mayor transferencia de carga se hace más evidente. También aumenta el valor del error absoluto medio llegando a un valor de 0,0314, aumentando considerablemente respecto al caso del sedán.

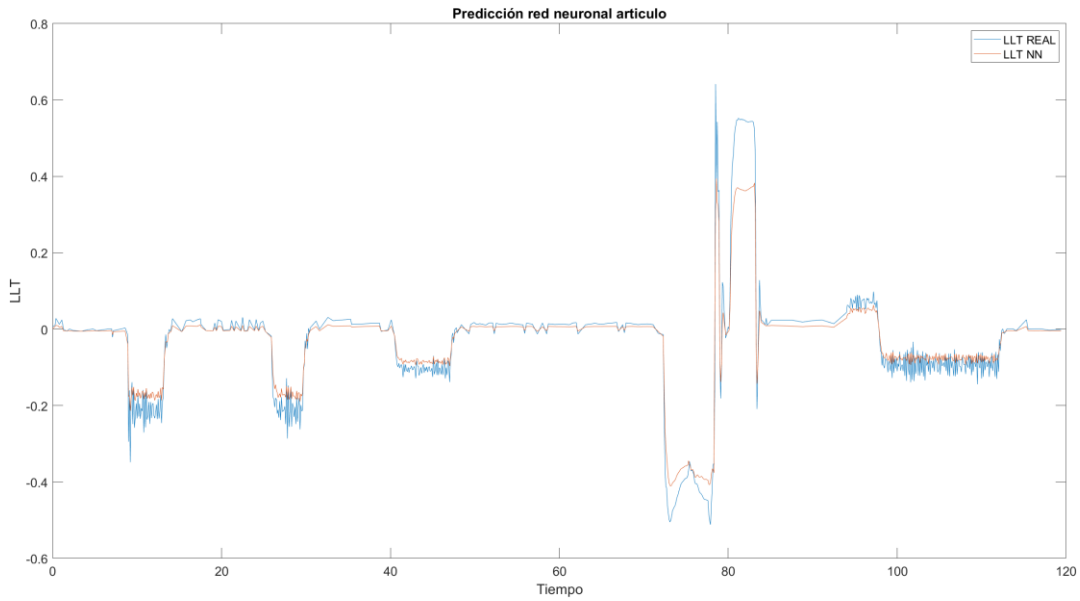


Ilustración 93: Predicción LLT Pick-up con red neuronal del Crossover

La disminución o aumento de la altura del centro de gravedad y la disminución del ancho de vías no parece ser el factor fundamental que provoca que la red comience a dar unos resultados con demasiado error. Sin embargo, el aumento de la masa total del vehículo (Tabla 9), al ser la mayor diferencia en comparación entre en sedan y la pick-up, parece producir un aumento significativo del valor del error absoluto media, provocando una respuesta más alejada de la realidad.

Por último, también se va a realizar la comparación de los datos con el autobús (Ilustración 94). Se entiende que, con los resultados obtenidos anteriormente, los resultados que será capaz de obtener la red neuronal entrenada con los datos del crossover no se acercaran lo suficiente a la realidad. El mayor problema que se presenta en esta simulación es el aumento exponencial de la masa del vehículo junto con mayores diferencias en el ancho de vías y en la altura del centro de gravedad. En este caso, se producen aumento de un 24% en el ancho de vías y la altura del centro de gravedad se multiplica por dos.

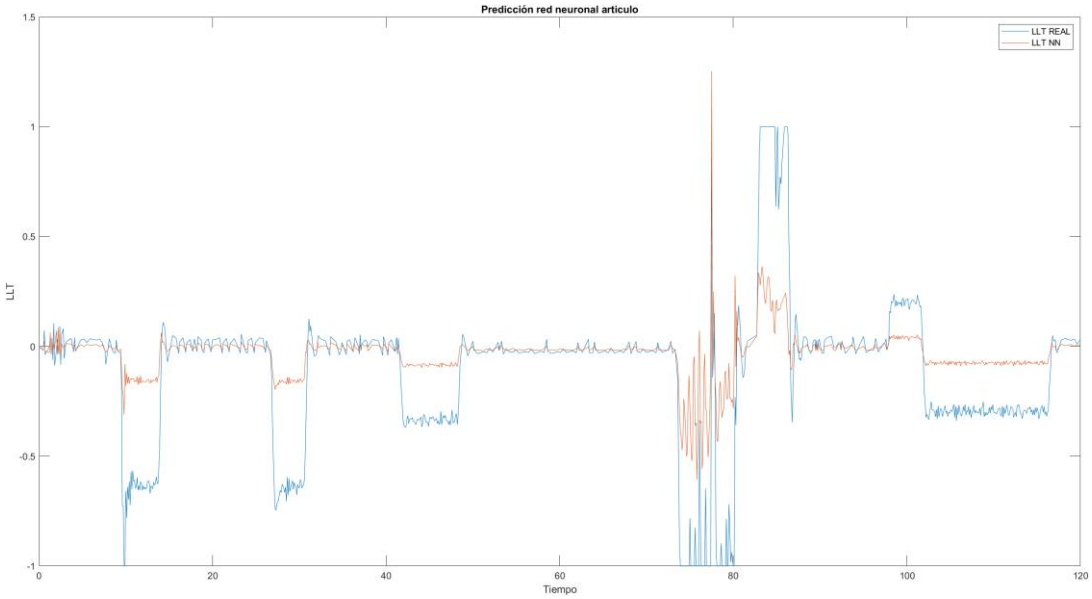


Ilustración 94: Predicción LLT Autobús con red neuronal del Crossover

Como se anticipaba, la predicción realizada por la red neuronal se aleja mucho de la transferencia de carga real que se produce en el autobús. Se ha aumentado excesivamente parámetros que afectan en gran medida a la transferencia de carga. El error absoluto medio ha aumentado hasta un valor de 0.22 siendo prácticamente 20 veces mayor que en el caso del sedán con la predicción de los datos del crossover. Debido a esta diferencia, se determina que no es posible la utilización de datos de vehículos tan dispares sin una clasificación previa de los vehículos en función de su masa, altura del centro de gravedad y ancho de vías.

Sin embargo, en la simulación realizada con un mayor número de variables de entrada para entrenar la red, los resultados que se presentan, sin ser un fiel reflejo de la realidad, mejoran notablemente el resultado anterior. En este caso el error aumento hasta un valor de 0.11, siendo 10 veces mayor que en las simulaciones previas, pero siendo la mitad del error que se obtenía con tan solo cinco variables de entradas.

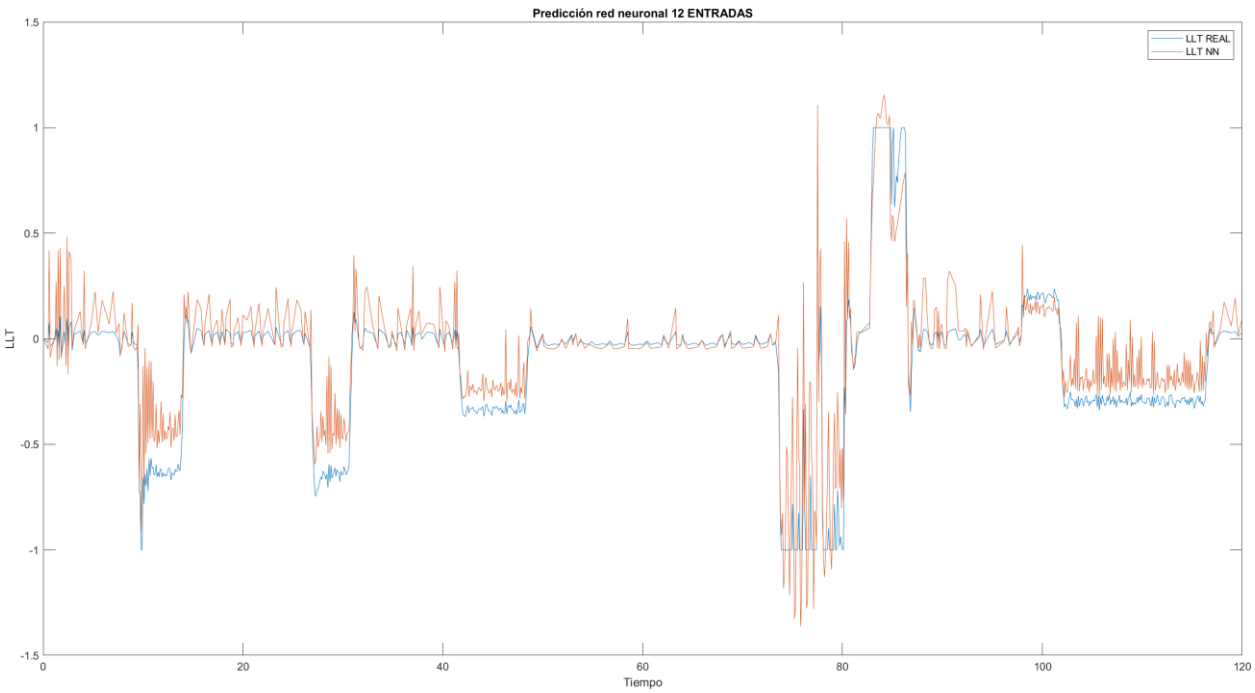


Ilustración 95: Predicción LLT Autobús con red neuronal del Crossover con 12 entradas

En la ilustración 95 se muestra el resultado obtenido utilizando un número mayor de variables de entradas para entrenar la red, y por consiguiente para realizar la predicción. En este caso, como en la mayoría de los casos, los resultados mejoran respecto al uso de un número menor de variables de entradas. Anteriormente se descarto está opción porque se utilizaban muchos recursos para la obtención de resultados prácticamente idénticos. Sin embargo, en este caso se muestra la mejora que se podría obtener si se utilizara una red con un mayor número de variables. Incluyendo variables geométricas se podría de predecir de forma correcta la transferencia de carga lateral sin tener que realizar una clasificación previa del vehículo.

6.2 Resultados con red neuronal con datos geométricos.

A continuación, se va a estudiar la posibilidad de utilizar redes neuronales con variables de entradas que caracterizan a los vehículos de forma geométrica. Se busca una generalización total mediante una sola red, la cual sea capaz de predecir la transferencia de carga lateral de una forma óptima para cualquier tipo de vehículo. Esta red se entrenaría previamente con diferentes tipos de vehículos, pero sin tener que particularizar uno a uno todos los vehículos que existen para el buen funcionamiento del sistema.

Para desarrollar este análisis se va a recurrir a la modificación de las redes que ya se habían programado anteriormente. Se introducirá en las bases de datos existente unas nuevas columnas de datos. Estos serán la altura del centro de gravedad, el ancho de vías, longitud y la masa del vehículo.

Se eligen estas variables como las características principales para diferenciar los vehículos por los estudios presentados en el apartado uno y dos de este proyecto. Estas variables son las más influyentes en el vuelco de vehículos, además de utilizarse en la primera aproximación matemática para la obtención de la aceleración lateral (cociente entre el ancho de vías y dos veces la altura del centro de gravedad).

Una vez se tenga la red neuronal preparada se realizará el mismo ensayo que anteriormente. En primer lugar, se entrenará la red con datos de todos los vehículos, con lo que se obtendrán la red entrenada lista para predecir resultados de otros vehículos. En segundo lugar, se introducirán en estas redes los datos obtenidos de las pruebas en el circuito de los mismos vehículos que en el apartado anterior: el sedán, la pickup y el autobús. Por último, se compararán los resultados obtenidos con la transferencia de carga real, al igual que se comparan los resultados con el apartado anterior. Se busca la opción de comparar ambos métodos de generalización, así mismo, se quiere probar la opción de una sola red capaz de predecir los resultados a la vez que clasificar los vehículos.

En primer lugar, se muestran los resultados obtenidos después de haber entrenado la red con datos de todos los vehículos e incluyendo los datos geométricos de cada uno. Inicialmente, el número de variables que se utilizan ha aumentado, lo que prácticamente fuerza a que la red neuronal vaya a necesitar un mayor número de neuronas en la capa oculta. Además, al introducir datos de diferentes tipos de vehículos también se aumenta ligeramente el número de datos utilizados para entrenar la red. Esto repercute en el tiempo de computación, que para este problema no es muy significante, pero se ha visto aumentado.

Al igual que anteriormente los mejores resultados se obtienen generalmente cuando se utiliza una red con un mayor número de variables de entradas, anteriormente se ha llamado red de “12 entradas”. Sin embargo, se obtienen unos resultados muy similares con las cuatro variables del artículo, reduciendo a más de la mitad el número de variables. Se continuará mostrando los resultados de este tipo de redes.

A continuación se presentan los resultados obtenidos tras predecir la transferencia de carga lateral del autobús. Anteriormente, este ensayo ha provocado las mayores diferencias entre la predicción y la transferencia de carga real, llegando a tener errores absolutos medios de 0,22. En cambio, utilizando este método el error se ha reducido casi 20 veces hasta un error medio absoluto de 0,0140.

En este caso se han utilizado las 4 variables geométricas de entradas y las 4 variables dinámicas que se han utilizado para este tipo de ensayos. La red neuronal al aumentar el número de variables también ha crecido hasta utilizar 11 neuronas en la capa oculta.

En la ilustración 96 se muestra el resultado gráfico y se puede apreciar como la diferencia entre la transferencia de carga real y la que se ha predicho son prácticamente iguales excepto en la zona revirada, donde se produce una transferencia de carga total entre un lado y otro, por lo que se podría asumir que es una zona de inestabilidad donde se podría haber producido el vuelco. Esto explicaría la diferencia entre ambas curvas y los saltos en esa zona. Aun así, la predicción es satisfactoria ya que la predicción es prácticamente igual a la curva real.

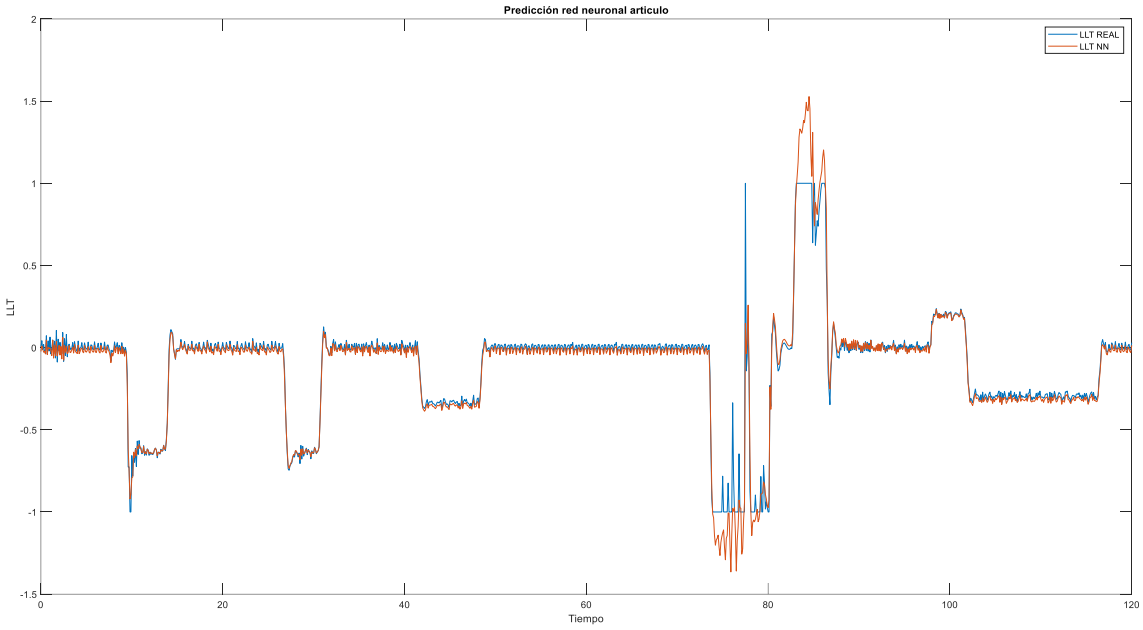


Ilustración 96: Predicción LLT autobús con red neuronal incorporando datos geométricos

El caso del ensayo utilizando los datos del sedán se obtiene un resultado, valorando el error absoluto medio, muy satisfactorio. Se obtiene un error absoluto medio de tan solo 0,0082, que se refleja de forma gráfica en la ilustración 97. En este caso la similitud en propiedades geométricas ha podido mejorar los resultados respecto al autobús, debido a que se utilizan más datos de vehículos similares como el crossover o la pickup para el entrenamiento de la red.

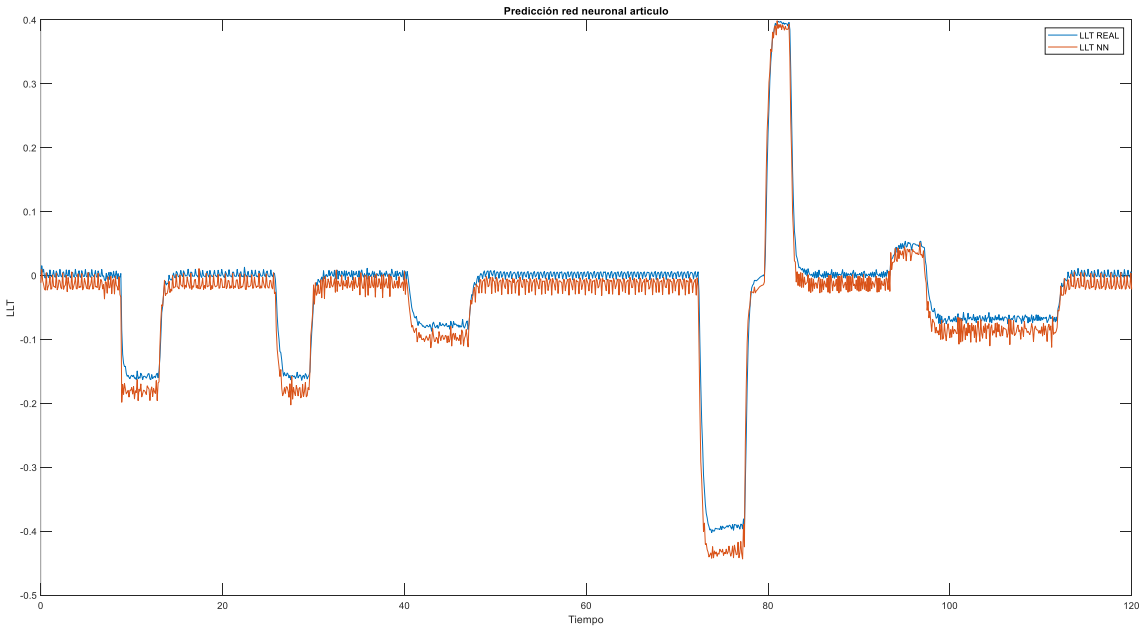


Ilustración 97: Predicción LLT sedán con red neuronal incorporando datos geométricos

Por último, los resultados obtenidos para el ensayo con la pickup (Ilustración 98) y con el crossover (Ilustración 98) muestran resultados parecidos. En el caso de la pickup se obtiene un error absoluto medio de 0,0116 y en el caso del crossover se obtiene un error de 0,0121, que se muestra en las ilustraciones 102 y 103, respectivamente.

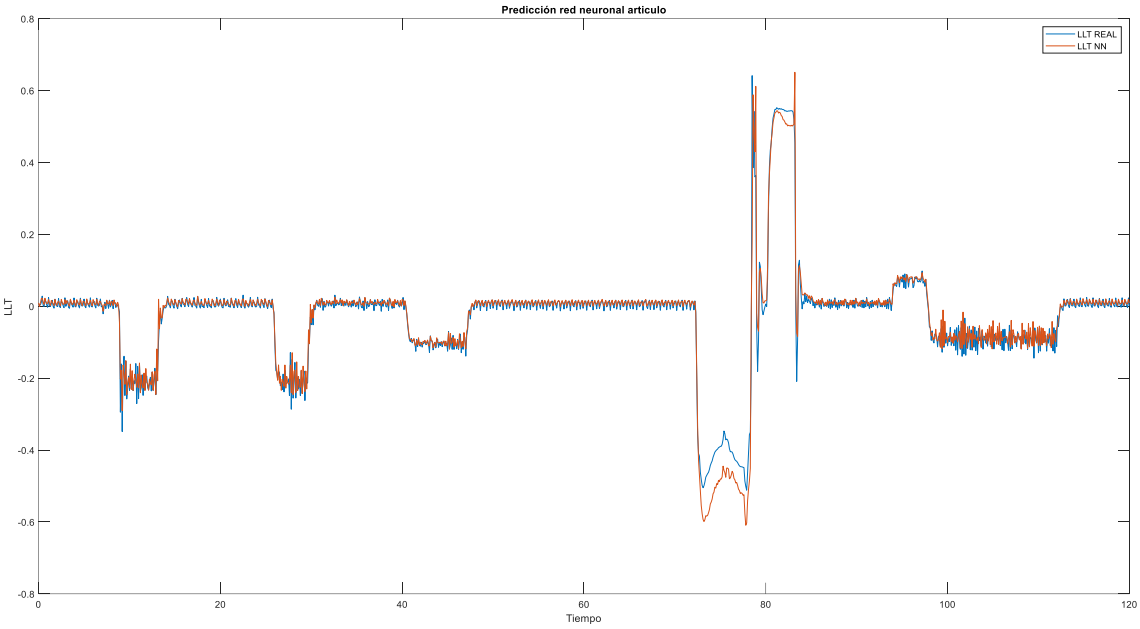


Ilustración 98: Predicción LLT pick-up con red neuronal incorporando datos geométricos

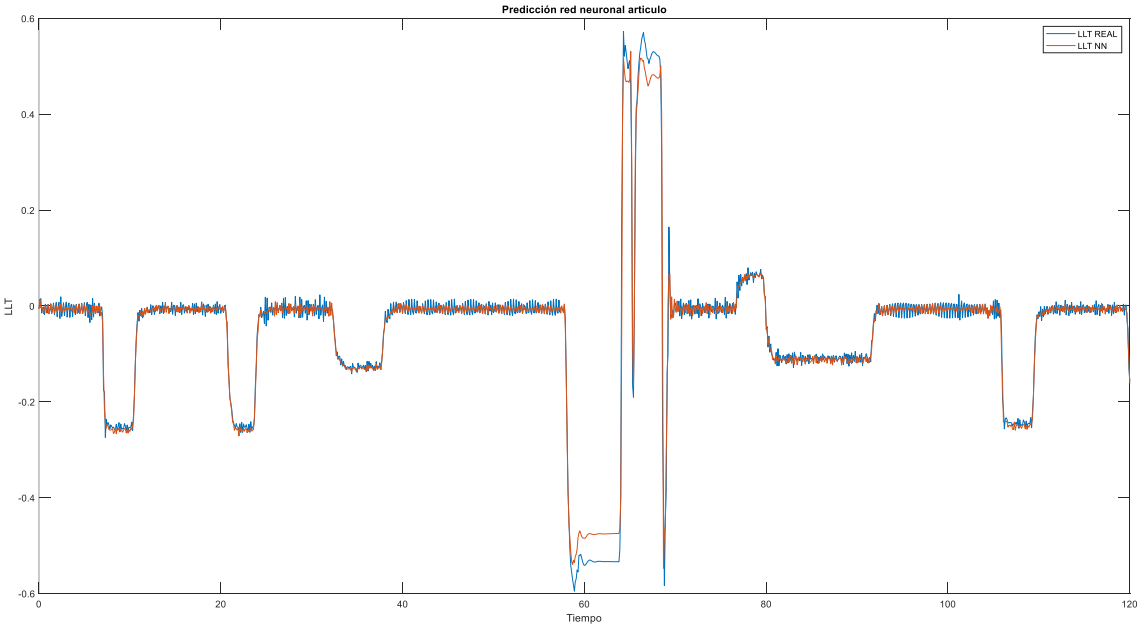


Ilustración 99: Predicción LLT crossover con red neuronal incorporando datos geométricos

6.3 Comparación de métodos de generalización del problema.

Inicialmente se presentaron dos hipótesis para generalizar el problema de obtener la transferencia de carga lateral de un vehículo mediante Machine Learning. La primera hipótesis consiste en utilizar una red neuronal para cada tipo de vehículos. Esto se consigue entrenando múltiples redes neuronales para distintos vehículos, en este caso un crossover, un sedán, una pickup y un autobús. Sin embargo, en este proyecto se ha entrenado una red para un tipo de vehículo y se ha estudiado la diferencia que habría con vehículos de distinta geometría para analizar el error que se podría cometer. De esta forma se puede estudiar la viabilidad de utilizar una red para cada tipo de vehículo y el error que se podría cometer con ciertas variaciones.

La segunda opción consiste en utilizar una red neuronal, entrenada con datos de varios tipos de vehículos, para predecir la respuesta de cualquier tipo de vehículo con cualquier tipo de geometría. En el proyecto se realiza una simulación de esta idea a pequeña escala. Se utilizan datos de todos los vehículos que se han nombrado anteriormente para entrenar la red. Posteriormente se utilizan los "datos de test" para evaluar el funcionamiento de la hipótesis.

La primera opción tiene una gran ventaja que es el buen funcionamiento para un determinado grupo de vehículos, aunque aceptando variaciones relativamente amplias. Por el contrario, si se quiere una gran generalización, que sea válida para cualquier vehículo, habría que tener muchísimas redes disponibles y bien catalogadas para utilizar en cada caso.

Sin embargo, con la segunda hipótesis se estaría en el caso contrario. En primer lugar, sería necesario tener una base de datos relativamente completa con variedad de vehículos para poder entrenar una red lo suficientemente capaz de predecir datos para cualquier tipo de vehículo. No obstante, una vez se tengan esos datos tendríamos una red neuronal capaz de predecir la transferencia de carga de cualquier vehículo. Comparándolo con la hipótesis anterior, con una cantidad de datos pequeña este método sería poco útil, sin embargo, al aumentar el número y variedad de datos, rápidamente se obtendrían mejores resultados con esta hipótesis con menor esfuerzo.

Hay que tener en cuenta que en la primera hipótesis, para vehículos con variaciones de sus características principales relativamente pequeñas, las predicciones han sido buenas. Sin embargo, al aumentar esta variación se obtienen resultados erróneos. Si se quiere llegar a una generalización mayor con este método, el número de datos y de redes neuronales entrenadas ira subiendo considerablemente, llegando un punto donde la segunda hipótesis necesite menos datos para aportar mejores resultados.

En resumen, ambos métodos podrían ser válidos, pero cada uno podría funcionar mejor en diferentes situaciones dependiendo del número de datos disponibles, su variabilidad en cuanto a geometrías de vehículos y dependiendo especialmente si solo se estudia un tipo de vehículo con pequeñas variaciones o si se quiere abarcar todo el mercado con una sola red.

7 CONCLUSIÓN

Este proyecto se inicia como la continuación del proyecto "Sistema Predictivo Anti-Vuelco de Vehículos" (Díaz Díaz, 2018). Anteriormente, el estudio se centró en entender la física que engloba el problema del vuelco de vehículos, junto con el estudio de algunos índices de vuelcos capaces de describir la situación de inestabilidad del vehículo. Sin embargo, este proyecto se ha centrado en la implementación de lo aprendido anteriormente mediante Machine Learning. Esta herramienta que se engloba dentro de la inteligencia artificial ha sido capaz de adaptarse al problema ofreciendo unos resultados satisfactorios, a la vez que brinda la posibilidad de seguir trabajando para la generalización del problema.

El proyecto se centra en la predicción del índice LLT o índice de transferencia de carga lateral, que describe la situación de inestabilidad comparando las fuerzas normales en las ruedas en ambos lados del vehículo. Inicialmente se ha realizado una comparación de la predicción de este índice para diferentes metodologías dentro del Machine Learning. Se han comparado métodos típicos como son el SVM o los árboles de decisión con las redes neuronales, concluyendo que el mejor resultado es ofrecido por las redes neuronales. En general, las redes neuronales han ofrecido mejores resultados, ofreciendo además la posibilidad de personalizar y ampliar el análisis del vuelco para la generalización del problema.

Tras esta comparación, se implementaron las redes para el vuelco de un vehículo determinado, el Crossover. Los resultados han sido muy buenos, ofreciendo errores absolutos medios alrededor de 0.01. El error absoluto es una medida general que se ha utilizado para comparar el valor real con el valor ofrecido por la red neuronal. Este valor es importante debido a la mejora que supone respecto a la representación de la realidad si se compara con otros índices de vuelco.

El Machine Learning, y más concretamente las redes neuronales, tienen un gran potencial a la hora de representar la realidad del vuelco mientras se tengan unos datos de calidad. Los datos que se utilizan para entrenar las redes tienen una gran importancia en los resultados y en este proyecto se ha demostrado. Mediante la modificación de las maniobras que se han simulado, haciendo que estas fueran en general más bruscas, se ha conseguido mejorar los resultados. Al reducir el tamaño de la base de datos, y a su vez aumentar las variaciones de carga a lo largo del tiempo, se ha obtenido una base de datos más compacta y con mayor información válida para entrenar la red.

Por último, se introduce la posibilidad de generalizar el problema mediante dos formas diferentes:

- La primera opción que se ha planteado consiste en utilizar un conjunto de redes neuronales para abarcar distintos tipos de vehículos. La idea es realizar una clasificación previa de los vehículos y catalogarlos en función de sus características geométricas y su masa. En el proyecto se ha estudiado que rango sería posible que admitiera una red entrenada por un vehículo. Utilizando otros vehículos para estudiar la respuesta que es capaz de dar la red previamente entrenada, se concluye que estás redes serían capaces de admitir pequeñas variaciones. Por lo tanto, se podrían crear redes neuronales para diferentes rangos de vehículos y que los resultados de la predicción del vuelco no sufrieran errores inadmisibles. Por otro lado, si se quisiera obtener resultados muy exactos para cualquier tipo de vehículo sería necesario dividir el conjunto de vehículos en muchas subdivisiones que a su vez generarían una gran cantidad de redes neuronales.

- La segunda opción consiste en introducir datos geométricos directamente en la red neuronal y que sea la propiedad red la encargada de predecir el valor del LLT de cualquier vehículo. En este caso se han obtenido muy buenos resultados introduciendo el ancho de vías, la altura del centro de gravedad, la masa y la longitud total del vehículo. La red neuronal, hasta en el caso de un autobús, ha ofrecido unos resultados muy semejantes a la realidad. Sin embargo, en el proyecto se ha realizado un pequeño acercamiento a la realidad: se han realizado las simulaciones con cuatro tipos de vehículos y con un conjunto de datos relativamente pequeño. Para poder llevar esto a cabo en la vida real sería necesario una gran cantidad de datos, con diferentes maniobras y diferentes vehículos para garantizar una generalización real. Además, al introducir una mayor cantidad de datos también aumentarían las opciones de sobreajuste y aumentaría el número de neuronas en la capa oculta.

En el caso de la generalización sería conveniente continuar con el estudio y análisis de estas posibilidades, teniendo en cuenta las particularidades de ambas opciones. De este análisis se puede concluir que, dependiendo del conjunto de datos, la variabilidad de los vehículos y el error que se pueda admitir una opción funcionara mejor que la otra, aunque ambas podrían ser soluciones viables del problema.

En resumen, la implementación del Machine Learning al problema del vuelco de vehículos ha supuesto un paso adelante. Se ha mejorado la predicción de la situación frente al vuelco y se ha demostrado la viabilidad de generalizar el problema mediante redes neuronales. Sin embargo, sería interesante continuar con la generalización del problema y su posible aplicación a mayor escala.

ANEXO

PROGRAMA ANÁLISIS DE VUELCO CON MACHINE LEARNING

Lectura de Datos

```
%Datos vehiculos


    % SEDAN
    alt_sedan=500;
    anch_vias_sedan=1800;
    long_sedan=4700;
    masa_sedan=850;

    % CROSSOVER
    alt_crossover=550;
    anch_vias_crossover=2100;
    long_crossover=4680;
    masa_crossover=850;

    % PICK-UP
    alt_pickup=600;
    anch_vias_pickup=1800;
    long_pickup=5100;
    masa_pickup=1700;

    % BUS
    alt_bus=1100;
    anch_vias_bus=2600;
    long_bus=14300;
    masa_bus=10300;
```

```
%Datos iniciales de simulación
Step_size=0.1;
Prep_datos_segundos=1;


%Añadir datos en función del vehiculo para asignar datos geometricos
    % SEDAN

    data_Sedan=datastore("TRAIN_Sedan*.txt");
    dataTrain_Sedan=readall(data_Sedan);% Tabla

    LLT_Del_sedan=calc_LLT_Trasero(dataTrain_Sedan);

    dataTrain_Sedan=dataTrain_Sedan{:,:}; %Para convertir en matriz
    dataTrain_Sedan=dataTrain_Sedan(:,1:16);
    for aux_7=1:height(dataTrain_Sedan)
    Vect_alt_sedan(aux_7,1)=alt_sedan;
    Vect_anch_vias_sedan(aux_7,1)=anch_vias_sedan;
    Vect_long_sedan(aux_7,1)=long_sedan;
    Vect_masa_sedan(aux_7,1)=masa_sedan;
    end
    dataTrain_Sedan(:,17)=Vect_alt_sedan;
    dataTrain_Sedan(:,18)=Vect_anch_vias_sedan;
    dataTrain_Sedan(:,19)=Vect_long_sedan;
    dataTrain_Sedan(:,20)=Vect_masa_sedan;



    % CROSSOVER

    data_Crossover=datastore("TRAIN_Crossover*.txt");
    dataTrain_Crossover=readall(data_Crossover);% Tabla

    LLT_Del_crossover=calc_LLT_Trasero(dataTrain_Crossover);

    dataTrain_Crossover=dataTrain_Crossover{:,:}; %Para convertir en matriz
    dataTrain_Crossover=dataTrain_Crossover(:,1:16);
    for aux_7=1:height(dataTrain_Crossover)
    Vect_alt_crossover(aux_7,1)=alt_crossover;
    Vect_anch_vias_crossover(aux_7,1)=anch_vias_crossover;
    Vect_long_crossover(aux_7,1)=long_crossover;
    Vect_masa_crossover(aux_7,1)=masa_crossover;
    end
```

```
dataTrain_Crossover(:,17)=Vect_alt_crossover;
dataTrain_Crossover(:,18)=Vect_anch_vias_crossover;
dataTrain_Crossover(:,19)=Vect_long_crossover;
dataTrain_Crossover(:,20)=Vect_masa_crossover;

% PICK-UP

   data_Pickup=datastore("TRAIN_Pickup*.txt");
dataTrain_Pickup=readall(data_Pickup);% Tabla

LLT_Del_pickup=calc_LLT_Trasero(dataTrain_Pickup);

dataTrain_Pickup=dataTrain_Pickup{:,:}; %Para convertir en matriz
dataTrain_Pickup=dataTrain_Pickup(:,1:16);
for aux_7=1:height(dataTrain_Pickup)
Vect_alt_pickup(aux_7,1)=alt_pickup;
Vect_anch_vias_pickup(aux_7,1)=anch_vias_pickup;
Vect_long_pickup(aux_7,1)=long_pickup;
Vect_masa_pickup(aux_7,1)=masa_pickup;
end
dataTrain_Pickup(:,17)=Vect_alt_pickup;
dataTrain_Pickup(:,18)=Vect_anch_vias_pickup;
dataTrain_Pickup(:,19)=Vect_long_pickup;
dataTrain_Pickup(:,20)=Vect_masa_pickup;

%AUTOBUS

data_Bus=datastore("TRAIN_B*.txt");
dataTrain_Bus=readall(data_Bus);% Tabla

LLT_Del_bus=calc_LLT_Trasero_bus(dataTrain_Bus);

dataTrain_Bus=dataTrain_Bus{:,:}; %Para convertir en matriz
dataTrain_Bus=dataTrain_Bus(:,1:16);
for aux_7=1:height(dataTrain_Bus)
Vect_alt_bus(aux_7,1)=alt_bus;
Vect_anch_vias_bus(aux_7,1)=anch_vias_bus;
Vect_long_bus(aux_7,1)=long_bus;
Vect_masa_bus(aux_7,1)=masa_bus;
end
dataTrain_Bus(:,17)=Vect_alt_bus;
dataTrain_Bus(:,18)=Vect_anch_vias_bus;
```

```
    dataTrain_Bus(:,19)=Vect_long_bus;
    dataTrain_Bus(:,20)=Vect_masa_bus;


dataTrain_ALL=[dataTrain_Crossover;dataTrain_Sedan;dataTrain_Pickup;dataTrain_Bus];



data2=datastore("TEST*.txt");
dataTest_ALL=readall(data2);
datatest_plot=dataTest_ALL;

LLT_test=calc_LLT_Trasero(dataTest_ALL);

dataTest_ALL=dataTest_ALL{:,:};
dataTest_ALL=dataTest_ALL(:,1:16);

for aux_8=1:height(dataTest_ALL)
    Vect_alt_test(aux_8,1)=alt_crossover;
    Vect_anch_vias_test(aux_8,1)=anch_vias_crossover;
    Vect_long_test(aux_8,1)=long_crossover;
    Vect_masa_test(aux_8,1)=masa_crossover;
 end
dataTest_ALL(:,17)=Vect_alt_test;
dataTest_ALL(:,18)=Vect_anch_vias_test;
dataTest_ALL(:,19)=Vect_long_test;
dataTest_ALL(:,20)=Vect_masa_test;



LLT_TEST=LLT_test;
LLT_TRAIN=[LLT_Del_crossover;LLT_Del_sedan;LLT_Del_pickup;LLT_Del_bus];
```

Preprocesado de datos

Eliminar ceros

```
%Se elimina cualquier fila que pueda estar duplicada y dar la misma
%informacion
%dataTrain=unique(dataTrain_ALL);


%Se identifica las filas que no dan informacion en los datos de TRAIN
```

```
%Se hace omitiendo el rango entre -0.009 y 0,009
flag_1=0;
for aux_1=2:(height(dataTrain_ALL))
    if dataTrain_ALL(aux_1,2)<0.009
       if dataTrain_ALL(aux_1,2)>-0.009
           flag_1=flag_1+1;
        Vector_aux(flag_1)=aux_1;
       end
    end
end
dataTrain_ALL([Vector_aux],:)=[];
LLT_TRAIN([Vector_aux],:)=[];
```

Eliminar Roll acceleration

```
%Se identifica las filas que no dan informacion
%Se hace omitiendo el rango fuera de -1.1 y 1.1
flag_3=0;
Vector_aux_3=0;
for aux_3=2:(height(dataTrain_ALL))
    if dataTrain_ALL(aux_3,4)>1.2
        flag_3=flag_3+1;
        Vector_aux_3(flag_3)=aux_3;
    elseif dataTrain_ALL(aux_3,4)<-1.2
           flag_3=flag_3+1;
           Vector_aux_3(flag_3)=aux_3;
    end

end
if Vector_aux_3~=0
    dataTrain_ALL([Vector_aux_3],:)=[];
    LLT_TRAIN([Vector_aux_3],:)=[];
end
```

Entrenamiento red neuronal con 12 entradas

```
dataTrain_NN_ALL=dataTrain_ALL(:,[2:6 10:20]);
dataTest_NN_ALL=dataTest_ALL(:,[2:6 10:20]);
Rango_num_Neuronas_ALL=(1:20);
```

```
for aux_6=Rango_num_Neuronas_ALL
NN_6=fitnet(aux_6);
NN_6.divideParam.trainRatio=70/100;
NN_6.divideParam.valRatio=15/100;
NN_6.divideParam.testRatio=15/100;


[NN_6,~]=train(NN_6,dataTrain_NN_ALL',LLT_TRAIN');
LLT_Del_NN_ALL=NN_6(dataTest_NN_ALL');


Error_prediccion_ALL(aux_6)=mean((abs(LLT_TEST-LLT_Del_NN_ALL')));


end


plot(Rango_num_Neuronas_ALL,Error_prediccion_ALL)
title('Curva de aprendizaje 12 ENTRADAS')
xlabel('Numero de neuronas')
ylabel('Error absoluto')



[Error_abs_prediccion_ALL,Numero_Neuronas_ALL]=min(Error_prediccion_ALL)
NN_6=fitnet(Numero_Neuronas_ALL);
NN_6.divideParam.trainRatio=70/100;
NN_6.divideParam.valRatio=15/100;
NN_6.divideParam.testRatio=15/100;


[NN_6,~]=train(NN_6,dataTrain_NN_ALL',LLT_TRAIN');
LLT_Del_NN_ALL=NN_6(dataTest_NN_ALL');
plot(dataTest_ALL(:,1),LLT_TEST)
hold on
plot(dataTest_ALL(:,1),LLT_Del_NN_ALL)
hold off
title('Predicción red neuronal 12 ENTRADAS')
legend("LLT REAL","LLT NN")
xlabel('Tiempo')
ylabel('LLT')



plot(dataTest_ALL(:,1),LLT_TEST-LLT_Del_NN_ALL')
```

```
title('Error LLT 12 ENTRADAS')
xlabel('Tiempo')
ylabel('Error absoluto')


disp('El error medio absoluto para 12 entradas es: ')
disp(Error_abs_prediccion_ALL)
```

Entrenamiento Red neuronal derivada de PCA

```
dataTrain_NN=dataTrain_ALL(:,[11 12 16 6 13 17:20]);
dataTest_NN=dataTest_ALL(:,[11 12 16 6 13 17:20]);
Rango_num_Neuronas=(1:20);
for aux_4=Rango_num_Neuronas
NN_4=fitnet(aux_4);
NN_4.divideParam.trainRatio=70/100;
NN_4.divideParam.valRatio=15/100;
NN_4.divideParam.testRatio=15/100;


[NN_4,~]=train(NN_4,dataTrain_NN',LLT_TRAIN');
LLT_Del_NN=NN_4(dataTest_NN');


Error_prediccion(aux_4)=mean((abs(LLT_TEST-LLT_Del_NN')));


end


plot(Rango_num_Neuronas,Error_prediccion)
title('Curva de aprendizaje PCA')
xlabel('Numero de neuronas')
ylabel('Error absoluto')


[Error_abs_prediccion,Numero_Neuronas]=min(Error_prediccion)
NN_4=fitnet(Numero_Neuronas);
NN_4.divideParam.trainRatio=70/100;
NN_4.divideParam.valRatio=15/100;
NN_4.divideParam.testRatio=15/100;


[NN_4,~]=train(NN_4,dataTrain_NN',LLT_TRAIN');
LLT_Del_NN=NN_4(dataTest_NN');
```

```
plot(dataTest_ALL(:,1),LLT_TEST)
hold on
plot(dataTest_ALL(:,1),LLT_Del_NN)
hold off
title('Predicción red neuronal PCA')
legend("LLT REAL","LLT NN")
xlabel('Tiempo')
ylabel('LLT')



plot(dataTest_ALL(:,1),LLT_TEST-LLT_Del_NN')
title('Error LLT PCA')
xlabel('Tiempo')
ylabel('Error absoluto')



disp('El error medio absoluto es: ')
disp(Error_abs_prediccion)
```

Entrenamiento Red neuronal derivada del articulo

```
dataTrain_NN_articulo=dataTrain_ALL(:,[2 3 15 16 17:20]);
dataTest_NN_articulo=dataTest_ALL(:,[2 3 15 16 17:20]);
Rango_num_Neuronas_articulo=(1:20);
for aux_5=Rango_num_Neuronas_articulo
NN_articulo=fitnet(aux_5);
NN_articulo.divideParam.trainRatio=70/100;
NN_articulo.divideParam.valRatio=15/100;
NN_articulo.divideParam.testRatio=15/100;

[NN_articulo,~]=train(NN_articulo,dataTrain_NN_articulo',LLT_TRAIN');
LLT_Del_NN_articulo=NN_articulo(dataTest_NN_articulo');

Error_prediccion_articulo(aux_5)=mean((abs(LLT_TEST-LLT_Del_NN_articulo')));

end

plot(Rango_num_Neuronas_articulo,Error_prediccion_articulo)
title('Curva de aprendizaje articulo')
```

```
xlabel('Numero de neuronas')
ylabel('Error absoluto')



[Error_abs_prediccion_articulo,Numero_Neuronas_articulo]=min(Error_prediccion_articulo)
NN_articulo=fitnet(Numero_Neuronas_articulo);
NN_articulo.divideParam.trainRatio=70/100;
NN_articulo.divideParam.valRatio=15/100;
NN_articulo.divideParam.testRatio=15/100;

[NN_articulo,~]=train(NN_articulo,dataTrain_NN_articulo',LLT_TRAIN');
LLT_Del_NN_articulo=NN_articulo(dataTest_NN_articulo');
plot(dataTest_ALL(:,1),LLT_TEST)
hold on
plot(dataTest_ALL(:,1),LLT_Del_NN_articulo)
hold off
title('Predicción red neuronal articulo')
legend("LLT REAL","LLT NN")
xlabel('Tiempo')
ylabel('LLT')




plot(dataTest_ALL(:,1),LLT_TEST-LLT_Del_NN_articulo')
title('Error LLT')
xlabel('Tiempo')
ylabel('Error absoluto')



disp('El error medio absoluto utilizando las variables del articulo es: ')
disp(Error_abs_prediccion_articulo)
```

REFERENCIAS

Adautomotor (2018) *¿Qué es el control de estabilidad o ESP?* Available at: https://adautomotor.es/que-es-el-control-de-estabilidad-o-esp (Accessed: 23 November 2020).

Alba, V. B. (2016) 'Tecnicas de aprendizaje automatico para prediccion de dianas de microRNA', *Zaguan.Unizar.Es*, pp. 0–43. Available at: http://zaguan.unizar.es/TAZ/EUCS/2014/14180/TAZ-TFG-2014-408.pdf.

Amat Rodrigo, J. (2017) *Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE*. Available at: https://www.cienciadedatos.net/documentos/35_principal_component_analysis (Accessed: 9 December 2020).

Bouton, N. *et al.* (2007) 'A rollover indicator based on the prediction of the load transfer in presence of sliding: Application to an all terrain vehicle', *Proceedings - IEEE International Conference on Robotics and Automation*, (April), pp. 1158–1163. doi: 10.1109/ROBOT.2007.363141.

Caparrini, F. S. (2017a) *Introducción al Aprendizaje Automático*. Available at: http://www.cs.us.es/~fsancho/?e=75 (Accessed: 12 November 2020).

Caparrini, F. S. (2017b) *Redes Neuronales: una visión superficial*. Available at: http://www.cs.us.es/~fsancho/?e=72 (Accessed: 16 November 2020).

DGT (2019) 'Las principales cifras de la siniestralidad vial 2019', pp. 1–226. Available at: http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/2015-2228_principales_cifras_de_la_Siniestralidad_Vial_2014_ACCESIBLE.pdf.

Díaz Díaz, J. (2018) 'Proyecto Fin de Grado Grado en Ingeniería de las Tecnologías Industriales Sistema Predictivo Anti-Vuelco de Vehículos'.

Freudling, T. (2018) *Detección de sobreviraje en automóviles BMW con aprendizaje automático*. Available at: https://es.mathworks.com/company/newsletters/articles/detecting-oversteering-in-bmw-automobiles-with-machine-learning.html (Accessed: 24 November 2020).

Gajdar, T., Rudas, I. and Suda, Y. (1997) 'Neural Network based estimation of friction coefficient of wheel and rail', in *IEEE International Conference on Intelligent Engineering Systems, Proceedings, INES*. IEEE, pp. 315–318. doi: 10.1109/ines.1997.632437.

García Guzmán, J. *et al.* (2018) 'Real-Time Vehicle Roll Angle Estimation Based on Neural Networks in IoT Low-Cost Devices', *Sensors (Basel, Switzerland)*, 18(7). doi: 10.3390/s18072188.

Lock, G. (2000) 'STABILITY CHARACTERISTICS OF Simulations Of Three And Four Wheeled Vehicles S Challener , K Chan & G Lock Project : Stability Characteristics Of Three And Four Wheeled Vehicles', (1).

Mangeas, M., Glaser, S. and Dolcemascolo, V. (2002) 'Neural networks estimation of truck static weights by fusing weight-in-motion data', in *Proceedings of the 5th International Conference on Information Fusion, FUSION 2002*. IEEE Computer Society, pp. 456–462. doi: 10.1109/ICIF.2002.1021190.

Martinez Heras, J. (2019) *Máquinas de Vectores de Soporte (SVM)*. Available at: https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/ (Accessed: 10 December 2020).

Merayo, P. (2020) *Qué son los árboles de decisión y para qué sirven*. Available at: https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/ (Accessed: 10 December 2020).

Merkle (2020) *Machine Learning y Support Vector Machines: porque el tiempo es dinero*. Available at: https://www.merkleinc.com/es/es/blog/machine-learning-support-vector-machines (Accessed: 10 December 2020).

Orellana Alvear, J. (2018) *Arboles de decision y Random Forest*. Available at: https://bookdown.org/content/2031/ (Accessed: 10 December 2020).

Rodrigo, J. A. (2016) 'Análisis discriminante lineal (LDA) y Análisis discriminante cuadrático (QDA)', p. 50. Available at: https://www.cienciadedatos.net/documentos/28_linear_discriminant_analysis_lda_y_quadratic_discriminant_analysis_qda.

RP, M., BC, J. and SJ, H. (2006) 'A study on SUV parameters sensitivity on rollover propensity', *SAE International*, 2006-01–07(724), pp. 1–12. doi: 10.4271/2006-01-0795.

Sanjuán, P. P. (1994) 'Un curso de automoción'.

Sheela, K. G. and Deepa, S. N. (2013) 'Review on methods to fix number of hidden neurons in neural networks', *Mathematical Problems in Engineering*, 2013. doi: 10.1155/2013/425740.

Tafner, R., Reichhartinger, M. and Horn, M. (2014) 'Robust online roll dynamics identification of a vehicle using sliding mode concepts', *Control Engineering Practice*, 29, pp. 235–246. doi: 10.1016/j.conengprac.2014.03.002.

Thangavel, P. (2018) *Supervised Learning - K Nearest Neighbors*. Available at: https://prasanth-ntu.github.io/html/ML-Course-12.html (Accessed: 10 December 2020).

Vargas-Meléndez, L. *et al.* (2016) 'A sensor fusion method based on an integrated neural network and Kalman Filter for vehicle roll angle estimation', *Sensors (Switzerland)*, 16(9). doi: 10.3390/s16091400.

WIDEBERG, N. P. J. I. (2016) 'Dinámica Lateral : Vuelco'.