Universidad Internacional de la Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Predicción de trayectorias en aplicaciones de control de movimiento industriales

Trabajo Fin de Máster

presentado por: Javier Menchén Agut

Dirigido por: Daniel Martínez Ortiz

Ciudad:Barcelona

Fecha: 13 de julio de 2021

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

Resumen

En ciertas aplicaciones de control de movimiento industriales donde un eje esclavo replica la trayectoria de un eje maestro, el hecho de anticipar la trayectoria del eje maestro puede suponer una mejora en el error de seguimiento del esclavo. La trayectoria es una sucesión temporal de posiciones de la cual se pretende anticipar cual va a ser el siguiente valor. En este trabajo se ha ensayado un modelo tipo Long Short-Term Memory (LSTM) para tal fin por su reconocida eficacia en predicciones de series temporales. El modelo LSTM se ha comparado con un modelo base tipo Multilayer Perceptron (MLP) y con un modelo de regresión polinómica. Se ha determinado el error medio y máximo y la relación del error con diferentes factores. Estos modelos han sido probados trabajando sin ruido y con ruido añadido a los datos de entrada. Se ha constatado que al trabajar sin ruido el modelo LSTM tiene un rendimiento parecido a la regresión polinómica pero se adapta más rápido a cambios en la trayectoria. Con ruido añadido a los datos el modelo MLP presenta una mayor robustez en la predicciones que los otros dos.

Palabras Clave: Control de Movimiento, Long Short-Term Memory, Predicción de Trayectorias

Abstract

In some industrial motion control applications where a slave axis follows the trajectory of a master axis, predicting the path of a master axis can reduce the following error of the slave. The trajectory is a time series of positions for which the next value will be predicted. For that purpose a Long Short-Term Memory (LSTM) model has been chosen due to its recognized efficiency in time series predictions. The LSTM model has been compared with a Multi Layer Perceptron (MLP) base model and with a polynomial regression model. The mean and maximum error of the model has been measured and also the relation of the error with diferent factors. These models have been tested working with and without noise added to the data. It has been found that when working without noise, the LSTM model presents a similar performance than the polynomial regression model but it is faster in adapting to changes in the trajectory. With noise added to the data, the MLP model presents a greater robustness in the predictions than the other two.

Palabras Clave: Motion Control, Trajectory Prediction, Long Short-Term Memory.

Capítulo 1

Introducción

1.1. Motivación

Desde el inicio de la revolución industrial la automatización de maquinaria y procesos ha seguido una evolución continua buscando siempre el incremento de capacidad de producción, la mejora de la calidad y la reducción de costes.

En el mundo de la automatización industrial el control de movimiento juega un papel de gran relevancia. El control de movimiento de elementos mecánicos permite mover los componentes, transformarlos, emsamblarlos para formar el producto final y finalmente empaquetarlo.

La tecnología de control de movimiento se ha ido refinando con el tiempo para conseguir mejores precisiones de posicionamiento y movimientos más coordinados puesto que esto repercute en la calidad final del producto. Esto ha sido posible gracias al incremento de potencia computacional de los controladores, a la mejora de los sensores y de la electrónica de control.

El movimiento en la maquinaria y los procesos industriales se lleva a cabo mediante actuadores que pueden ser lineales cuando llevan a cabo la traslación de un elemento o bien rotacionales cuando lo rotan. En algunas aplicaciones es habitual que un eje esclavo deba acoplarse a un eje maestro del cual no se tiene el control. El caso más simple de acoplamiento es cuando el eje esclavo replica el movimiento del eje maestro para seguirlo. En la figura 1.1 se puede observar el acoplamiento entre un robot y una cinta transportadora. Para poder manipular los objetos de la cinta mientras ésta está en movimiento la pinza del robot debe moverse en sincronismo con la cinta.

En estas aplicaciones el eje maestro debe comunicar su posición al eje esclavo para que

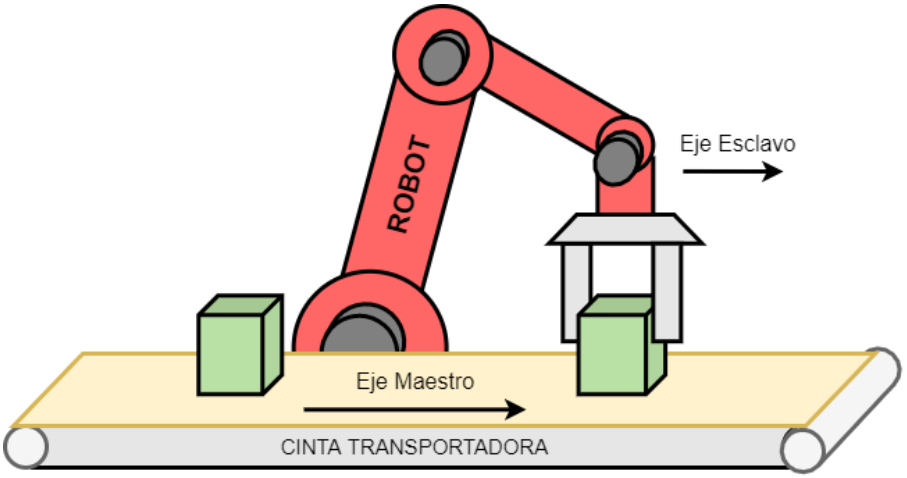


Figura 1.1. *Acoplamiento entre robot y cinta transportadora. Fuente: Propia*

éste pueda replicarla. Esta comunicación implica un retardo en el control que se traduce en un error de seguimiento. En este contexto puede ser interesante poder predecir los siguientes puntos de la trayectoria del eje maestro sobre el que no se tiene control directo. Esta anticipación permitiría reducir notablemente el error de seguimiento lo cual puede ser crucial en ciertas aplicaciones.

Otra posible aplicación de la predicción de trayectorias podría ser en la lectura de posición de encoder lineales. Los encoders son los elementos que permiten medir la posición real de un eje y transmitirla al controlador como feedback para la regulación. Los encoders lineales pueden verse afectados por la presencia de suciedad lo que interrumpe momentáneamente la lectura de posición, generando un error en el control. El hecho de poder suplir la lectura real por una predicción durante estos breves momentos mejoraría la fiabilidad del sistema.

1.2. Solución propuesta

La predicción de trayectorias en general se engloba dentro del campo de la predicción de series temporales. En este tipo de problemas las Redes Neuronales Recurrentes y más particularmente las redes Long Short-Term Memory (LSTM) han demostrado dar muy buenos resultados. Por tanto en este trabajo vamos probar este tipo de modelo al caso concreto de las predicción de trayectorias de ejes industriales.

La entrada del modelo va a ser una serie de posiciones, desde la posición actual y contando hacia atrás en el tiempo: t_0 y t_{-1}, t_{-2}, t_{-3}.... La predicción del modelo será la

posición en el siguiente instante t_2

1.3. Estructura de la memoria

En el capítulo 2 dedicado al estado del arte se hace una introducción a las aplicaciones industriales de control de movimiento para dar a conocer el contexto del trabajo. Seguidamente se revisan diferentes técnicas que se han empleado para la predicción de trayectorias en diversos ámbitos y justificaremos la elección del modelo LSTM para esta aplicación.

En el capítulo 3 se revisan las bases de funcionamiento de las redes Long Short-Term Memory por ser el modelo central de este trabajo.

En el capítulo 4 se exponen los objetivos que se pretenden alcanzar y la metodología empleada para su consecución.

En el capítulo 5 se detalla cómo se han generado los datos de entrenamiento, la arquitectura del modelo y el proceso de entrenamiento.

En el capítulo 6 se analiza el error que comete el modelo LSTM en las predicciones y la distribución del error respecto a diferentes variables.

En el capítulo 7 se comparan entre sí los modelos LSTM, MLP y la regresión polinómica en base al error cometido a la hora de predecir las mismas trayectorias.

En el capítulo 8 se reflexiona sobre qué modelo es más conveniente utilizar según diferentes posibles escenarios y se proponen trabajos futuros de mejora.

Capítulo 2

Estado del arte

En este capítulo, dedicado a repasar el estado del arte vamos a empezar introduciendo las aplicaciones industriales de control de movimiento para dar a conocer el contexto de la aplicación. Veremos en qué casos puede ser útil llevar a cabo una predicción de trayectorias en este campo. Seguidamente se va a hacer un repaso de diferentes técnicas de Inteligencia Artificial empleadas en la predicción de trayectorias, la mayoría de ellas aplicadas a diferentes ámbitos del que nos concierne pero con cierto grado de similitud.

2.1. Las aplicaciones de control de movimiento

Cuando es necesario controlar el movimiento de ejes mecánicos de forma precisa se suelen utilizar servomotores rotacionales o lineales que cuentan con encoders de alta resolución para determinar su posición. Estos encoders actúan como realimentación de un regulador automático que se encarga de que el motor mantenga en todo momento su posición de consigna a pesar de perturbaciones externas. Un control superior genera la trayectoria de los ejes para llevar a cabo diferentes tipos de movimiento. La trayectoria definida consiste en una serie temporal de consignas de posición. El controlador de movimiento entregará en cada intervalo de tiempo una nueva consigna al regulador del eje. Este intervalo de tiempo es el tiempo de ciclo del controlador.

Las trayectorias de los ejes para moverse de un punto a otra se definen en base a unos parámetros que son la velocidad máxima, la aceleración máxima y el jerk que es la derivada o rampa de la aceleración. Si la posición en un movimiento uniformemente acelerado se determina mediante una ecuación de segundo grado, en este tipo de movimiento se hace con un polinomio de tercer grado (Payeur et al., 1995).

$$x(t) = \frac{1}{6}at^3 + \frac{1}{2}bt^2 + ct + x_0 \tag{2.1}$$

En la generación de trayectorias es típico la utilización de perfiles de movimiento de 7 fases (Lewin, 2007). En cada una de estas fases los parámetros de la curva se mantienen constantes. En cambio al cambiar de fase cambian también los parámetros. Esto significa que en cada fase la aceleración y la velocidad se van a comportar de manera diferente tal como se puede apreciar en la figura 2.1

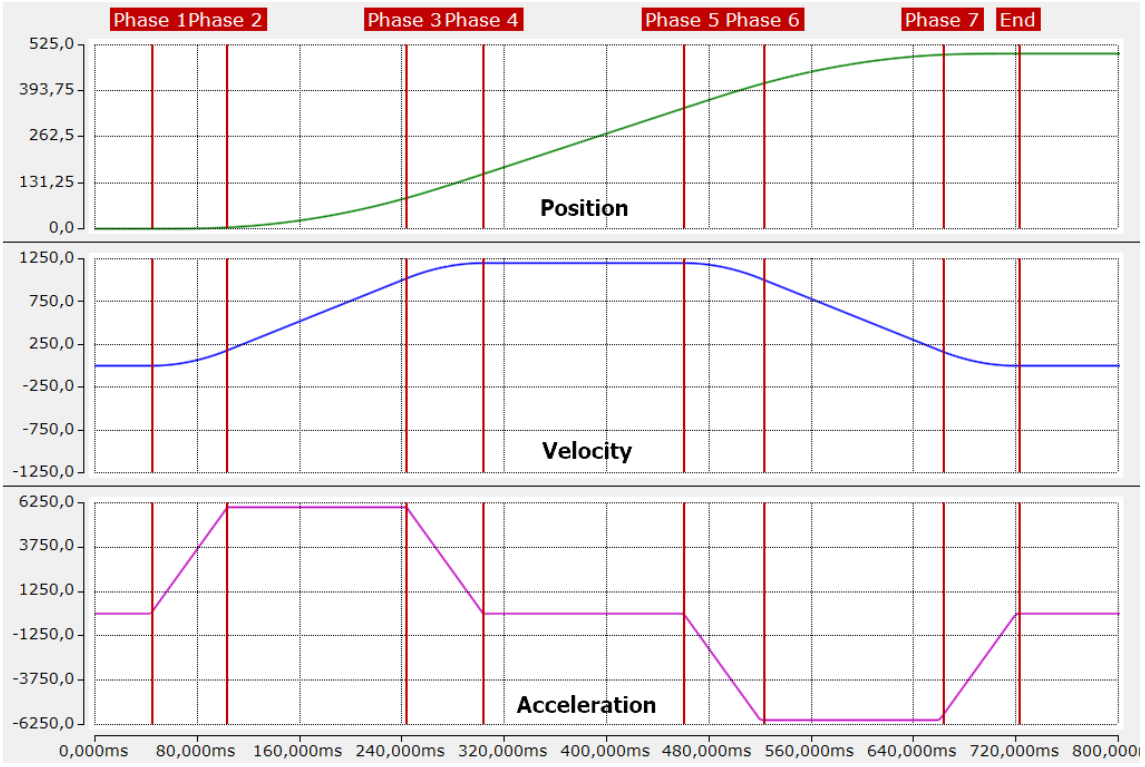


Figura 2.1. *Perfil de movimiento de 7 fases. Fuente: Propia*

Con esta estrategia no se producen escalones abruptos de aceleración. En su lugar la aceleración aumenta linealmente hasta alcanzar su valor máximo. Esto permite que la velocidad cambie suavemente siguiendo una función cuadrática.

A continuación se expone la descripción de cada fase de la trayectoria.

- Fase 1. Empieza la fase de aceleración en que ésta es lineal. Permite una transición suave entre la velocidad cero y la fase de velocidad lineal

- Fase 2. Aceleración constante. La velocidad aumenta linealmente

- Fase 3. Fin de la fase de aceleración. Se decrementa linealmente de la aceleración. Permite una transición suave entre la velocidad lineal y la velocidad constante.
- Fase 4. Aceleración cero y velocidad constante. La velocidad constante se mantiene hasta aproximarse a la posición de destino.
- Fase 5. Empieza la base de desaceleración. Incremento lineal negativo de la aceleración. Análoga a la fase 3.
- Fase 6. Aceleración constante negativa análoga a la fase 2
- Fase 7. Decremento lineal negativo de la aceleración. Análoga a la fase 1. Finaliza en estado de reposo con velocidad cero.

En las aplicaciones control de movimiento en máquinas industriales en ocasiones es necesario que un controlador esclavo deba sincronizar uno o más ejes con un eje maestro gobernado por otro controlador. Hay varios tipos de sincronización que se pueden dar. El más sencillo de todos seria un acoplamiento lineal en velocidad en que el eje esclavo debe seguir una trayectoria proporcional a la del eje maestro aplicando un factor de acoplamiento. En caso de que este factor valga 1 el eje esclavo replicará exactamente la trayectoria del eje maestro. En todo caso la velocidad del esclavo será igual a la del maestro multiplicada por el factor de acoplamiento.

En otras ocasiones la sincronización del eje esclavo con el maestro se lleva a cabo mediante levas electrónicas para modelar una relación no lineal entre ambos. En la figura 2.2 se puede apreciar un ejemplo de descripción de leva electrónica.

Estas levas electrónicas se representan mediante tablas que relacionan la posición del eje esclavo con la del eje maestro. El eje esclavo se representa en el eje Y de la tabla y el eje maestro en el eje X. Se definen ciertos puntos de paso en la tabla y también se definen las funciones para interpolar la posición entre lo puntos definidos. Estas funciones suelen ser polinómicas de diferente grado. Las polinómicas de grado 7 son bastante adecuadas ya que nos permiten definir posición, velocidad, aceleración y jerk en el punto inicial y en el punto final del segmento lo cual corresponde a 8 restricciones.

2.2. Problemática en la sincronización y soluciones actuales

Cuando el eje maestro y el eje esclavo de un acoplamiento están controlados por el mismo controlador las consignas de posición se calculan y se entregan simultáneamente a

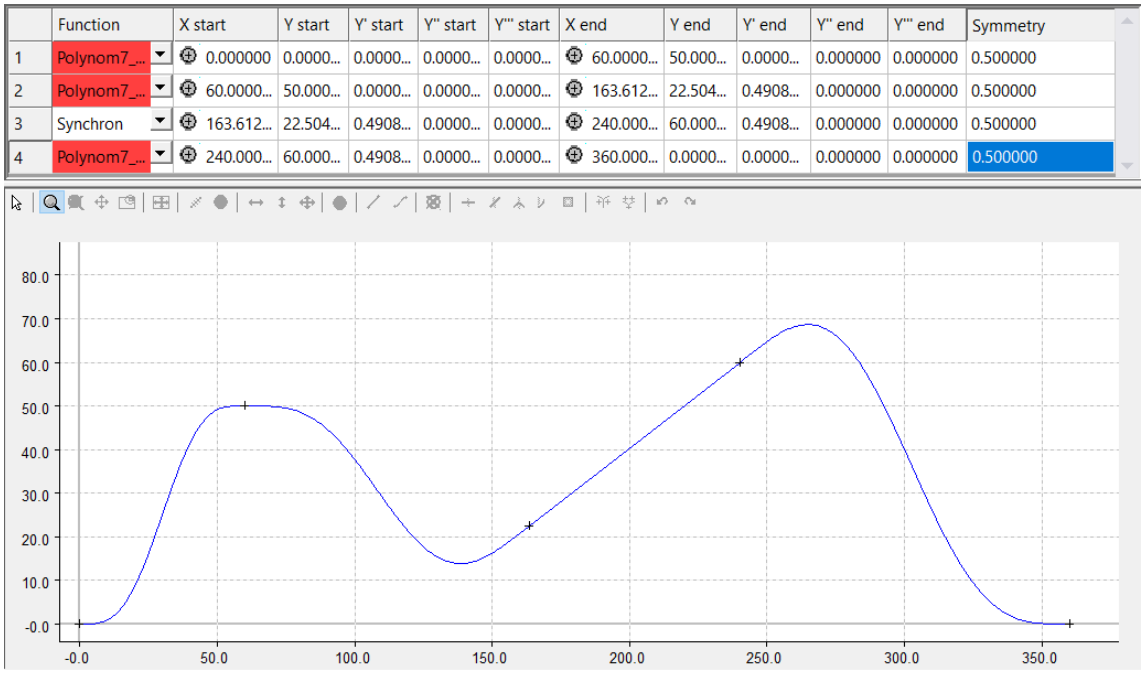


Figura 2.2. *Ejemplo de leva electrónica. Fuente: Propia*

ambos ejes por lo que no se produce un desfase entre ambos.

Sin embargo en algunas ocasiones el eje maestro y el eje esclavo se encuentran en diferentes controladores por lo que el controlador del eje esclavo no conoce la posición del eje maestro a priori. En esta circunstancia es necesario comunicar la consigna de posición desde el controlador maestro al controlador esclavo. Esto se puede hacer mediante un bus de campo (EtherCAT, ProfiNET, Sercos...) o bien mediante en envío de pulsos de encoder. Esta comunicación introduce un retardo en la llegada de las posiciones al controlar esclavo que se traduce en un error de seguimiento.

Cuando se requiere mucha precisión en un acoplamiento maestro-esclavo se suele procurar que ambos ejes estén controlados por el mismo controlador, con lo cual el problema desaparece. Sin embargo en ocasiones esto no es posible porque los ejes pertenecen a máquinas diferentes.

Una forma de minimizar el problema es reducir al máximo el tiempo de ciclo de la comunicación de consignas entre controlador maestro y esclavo. Esto reduce el error de seguimiento pero nunca la reduce del todo. Hay aplicaciones en que el error resultante es admisible. En otras ocasiones se requiere más precisión en la sincronización para el correcto funcionamiento o incluso para evitar colisiones mecánicas. Por lo tanto seria deseable reducir todavía más este error de seguimiento.

El hecho de predecir la siguiente posición de la trayectoria del eje maestro permitirá al controlador esclavo anticiparse y tener una reducción significativa del error de seguimiento. Las trayectorias que suelen seguir los ejes industriales son curvas polinómicas con parámetros fijos definidos por tramos. Por tanto dentro de un tramo dado la trayectoria es predecible. Al cambiar de un tramo a otro cambian los parámetros y esto no es tan predecible por lo que en esas transiciones tendremos que esperar un error de seguimiento puntual.

2.3. Problemática de los encoder lineales y soluciones actuales

En ejes lineales en ocasiones se lleva a cabo una lectura de la posición real mediante un encoder lineal. El encoder lineal se compone de una regla óptica o magnética instalada a lo largo del recorrido y una cabeza lectora que se mueve con el eje y lee la posición de la regla.

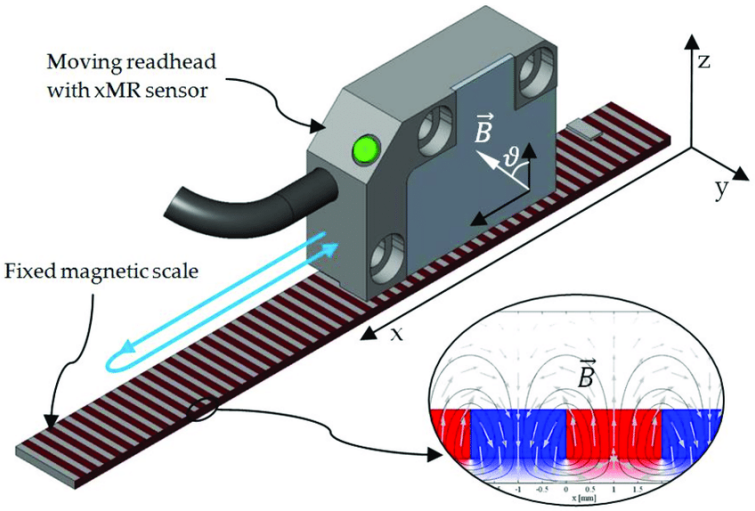


Figura 2.3. *Imagen de cabeza lectora de regla magnética. Fuente: (ResarchGate, 2021)*

Estos sistemas son sensibles a suciedad que se pueda depositar sobre la regla impidiendo la lectura de posición en ciertas zonas. Esta pérdida momentánea de lectura de posición es crítica puesto que se utiliza como retroalimentación de la regulación del eje.

En la actualidad existen algunos sistemas que permiten estimar la posición durante la interrupción en base a una estimación de la velocidad y la aceleración previa. Esto soluciona el problema en cierta medida pero ante una pérdida un poco más prolongada no

es una estimación suficientemente precisa.

En este caso una predicción de la posición mediante una red LTSM entrenada con trayectorias más complejas mejoraría el funcionamiento ante la pérdida de lectura. Cuanto mejor sea la predicción de la trayectoria durante la pérdida mejor será el reenganche con la posición real una vez se restablece la lectura de posición.

2.4. Predicción de trayectorias basada en modelos probabilísticos

Una de las maneras de enfocar la predicción trayectorias es mediante modelos probabilísticos como modelos de Markov o modelos mixtos gausianos. Estos métodos sin embargo no hacen una verdadera regresión sino que, mediante una estrategia u otra, hacen una clasificación sobre un conjunto finito de posible elecciones.

En el trabajo de (Choi and Hebert, 2006) se aborda la predicción de trayectorias para objetos mediante cadenas de Markov. Se dividen las trayectorias en segmentos de una longitud determinada. Los segmentos se normalizan para que partan del origen y con la misma orientación. Estos segmentos se agrupan mediante K-means según similitud donde cada grupo es representado por un segmento llamado segmento latente. Los segmentos reales se consideran que son derivados de los segmentos latentes con ruido añadido. Se asume que la ocurrencia de un segmento latente determinado es dependiente del segmento latente anterior. Mediante un modelo de Markov se computan las probabilidades de co-ocurrencia de dos estados latentes. Los autores escogen para este trabajo un modelo de Markov de segundo orden por lo que se tienen en cuenta dos segmentos latentes pasados para computar la probabilidad del siguiente.

Como podemos observa este modelo no devuelve una predicción en el espacio continuo de coordenadas xy. Por el contrario nos devuelve una selección del segmento latente más probable de un número finito de segmentos latentes. Esto va a representar una aproximación de las trayectoria real.

En este otro trabajo de (Ye et al., 2016) se utilizan modelos ocultos de Markov junto con el algoritmo de Viterbi para predecir la trayectoria más probable de vehículos. En este caso las trayectorias tratadas no son continuas en el plano xy sino que representan la posición del vehículo en un grafo que modeliza el mapa de carreteras, donde los nodos son intersecciones y las aristas son tramos de carretera entre dos nodos. Un vehículo sólo

puede cambiar de trayectoria al llegar a una intersección.

Vemos por tanto que el tipo de problema es bastante diferente al que se pretende abordar en este trabajo donde lo que nos interesa es predecir posiciones en el espacio continuo con la mayor precisión posible.

Finalmente también tenemos ejemplos en que se han aplicado métodos probabilísticos como modelos mixtos gausianos (Wiest et al., 2012) a la predicción de trayectorias de vehículos.

2.5. Modelo MLP de Payeur para la prediccion de trayectorias

Revisando el estado del arte para la predicción de trayectorias nos encontramos con que hay muy pocas aplicaciones para el control de ejes en máquinas industriales. Sin embargo hay una aplicación muy similar aunque antigua (Payeur et al., 1995) en que se hace la predicción de trayectorias de objetos móviles mediante redes neuronales de tipo Multilayer Perceptron (MLP). Hay que tener presente que esta aplicación es anterior a la aparición de las redes LSTM.

Este trabajo aborda el problema de predecir la trayectoria de un objeto móvil que se desliza por una rampa. El objeto puede hacer movimientos de traslación en los ejes x,y y también puede rotar sobre el eje z. Un sistema de visión artificial se encarga de transmitir al robot estas posiciones a intervalos regulares.

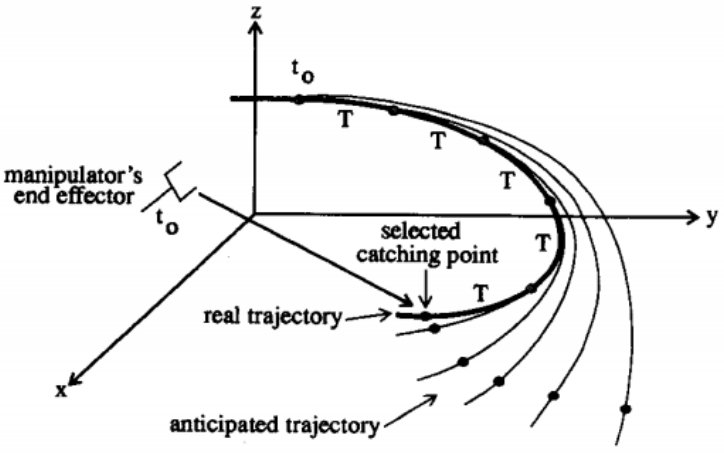


Figura 2.4. *Predicción de trayectorias en el plano para la captura del robot. Fuente: (Payeur et al., 1995)*

El objetivo es que un robot haga uso de dicha predicción para planificar una trayectoria que le permita interceptar el objeto.

Para cada uno de los ejes x,y y θ se pretende hacer una predicción del siguiente valor de posición, velocidad y aceleración. Para cada una de estas magnitudes se emplea un modelo diferente. No obstante la entrada de cada uno de estos modelos es común y consiste en los últimos seis valores de posición del eje correspondiente.

El modelo empleado es de tipo Multilayer Perceptrón con dos capas ocultas de 20 unidades cada una como se puede observar en la figura 2.5. Como función de activación se usa la función sigmoide.

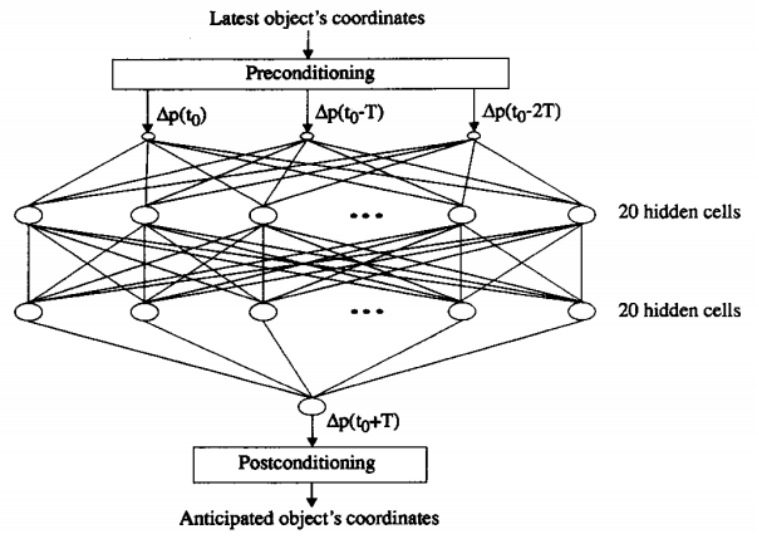


Figura 2.5. *Modelo empleado tipo Multi Layer Perceptron. Fuente: (Payeur et al., 1995)*

En lugar de utilizar directamente las posiciones absolutas como entradas de la red neuronal se utilizan los incrementos de cada posición con la siguiente. Esta información es necesaria para calcular las sucesivas derivadas de la trayectoria. Esto se hace en una etapa previa de pre-procesamiento para liberar a la red neuronal de encontrar estas características por si misma durante el entrenamiento. Además la salida de la red debe ser la predicción del siguiente incremento de posición. De esta manera se consigue que las predicciones sean totalmente independientes de la posición absoluta.

2.5.1. Resultados y conclusiones del modelo de Payeour

Los autores comparan el resultado de hacer la predicción de las trayectorias mediante un modelo de regresión polinómica y mediante una red neuronal entrenada para predecir

trayectorias de este tipo. De hecho se utiliza la función cubica para generar los datos de entrenamiento de la red neuronal. Se comprueba que el error medio cometido por ambos modelos es prácticamente igual con la ecuación cubica que con la red neuronal. Esto indica que el error cometido por la red neuronal proviene del modelo cúbico con el que se ha entrenado.

En la figura 2.6 puede observarse una gráfica con una trayectoria senoidal para el eje X y la predicción del modelo para la misma trayectoria. La calidad de la gráfica es bastante mala pero pueden apreciarse pequeñas divergencias entre trayectoria real y la predicción. En el trabajo no se especifica cual es el error medio para este caso particular. Esto tendremos oportunidad de estudiarlo en más detalle mediante la réplica del modelo MLP sometida a los mismos test que el LSTM

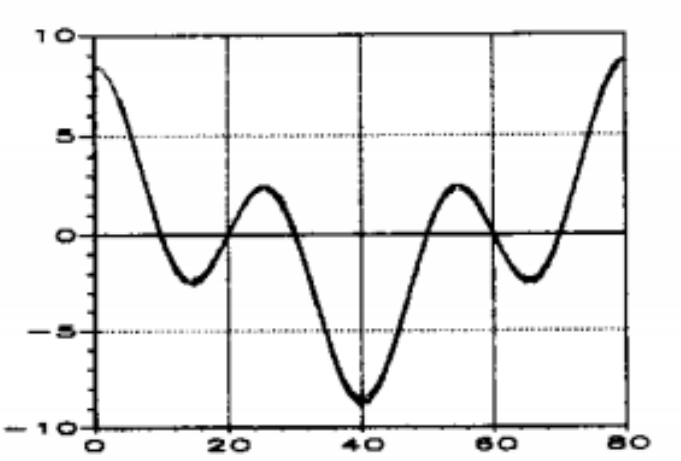


Figura 2.6. *Trayectoria real y predicción de trayectoria del eje X. Fuente: (Payeur et al., 1995)*

Por otro lado se compara el desempeño de ambos modelo añadiendo ruido aleatorio a las variables de entrada. Se observa que el ruido medio en la predicción es prácticamente idéntico en un caso y en el otro. En cambio la varianza del ruido es menor con la red neuronal que con la ecuación cúbica de lo que se deduce que la red neuronal es menos sensible al ruido.

En este artículo se destacan ciertas ventajas de las redes neuronales frente a la ecuación analítica. La primera de ellas es que la red neuronal puede ser entrenada de forma flexible para modelizar diferentes tipos de trayectorias. Si un modelo cúbico no fuera suficiente se podría entrenar para modelizar trayectorias más complejas. La otra ventajas más significativa es que la red neuronal es menos sensible al ruido en los datos de entrada.

2.5.2. Limitaciones del modelo de Payeour

Para el entrenamiento del modelo de (Payeur et al., 1995) se generan trayectorias de entrenamiento mediante una ecuación polinómica de tercer grado. Se generan secuencias de 6 posiciones con diferentes combinaciones de parámetros pero los parámetro para una trayectoria son siempre constantes. Por tanto en este trabajo no se mide la capacidad del modelo a adaptarse a las trayectorias cambiantes que tiene un perfil de movimiento de 7 fases.

2.6. Predicción de trayectorias mediante modelos LSTM

Los modelos LSTM también se han empleado para la predicción de trayectorias de diferentes tipos de objetos o entidades.

Un campo muy activo para la predicción de trayectorias es el de los vehículos autónomos. Es muy conveniente poder adaptar su comportamiento dependiendo de los movimientos de los vehículos próximos. En ese contexto se enmarca este trabajo de (Altché and de La Fortelle, 2017) en que se utilizan modelos LSTM para la predicción de trayectorias de vehículos en autopistas. Como características para el modelo se utilizan la posición y velocidad longitudinal y transversal (respecto el sentido del tráfico) del vehículo objetivo. También se toman la posición y velocidad de los vehículos que rodean a éste. Además se utiliza como entrada el tipo de cada uno de estos vehículos. Estos datos se escalan en rango cercano al de las funciones de activación.

El modelo utilizado consta de una primera capa de 256 unidades LSTM seguidas de dos capas Fully Connected, una de 256 y otra de 128 unidades. Finalmente la capa de salida contendrá tantas unidades cómo variables queramos predecir. La posiciones y velocidades del vehículo objetivo se repiten y puentean directamente a la capa final. Los autores mencionan que esto mejora ligeramente el resultado. El modelo se entrena empleando ventanas de 100 datos que corresponden con 10s.

Se implementan diferentes variantes del modelo y se lleva a cabo un bagging de los cuatro mejores modelos lo cual mejora el resultado final. Este trabajo se compara con otro similar basado en Multi Player Perceptrón y se observa que el comportamiento es mejor para horizontes temporales mayores.

En un trabajo similar de (Park et al., 2018) se empleó un modelo Sequence-to-Sequence basado en LSTM para resolver este mismo problema. También se utilizan como entradas

las posiciones y velocidades de los vehículos vecinos y del vehículo objetivo. El encoder LSTM analiza el patrón subyacente en la trayectoria pasada de los vehículos y lo codifica en su estado interno c. El decodificador en esta ocasión no genera directamente la secuencia de coordenadas de salida correspondientes a la predicción de la trayectoria. En su lugar se encarga de generar un mapa de probabilidades de ocupación sobre un grid de posiciones que se conoce como Occupancy Grid Map (OGM) . Esto se combina con un algoritmo beam search que se encarga de seleccionar la salida más probable del decodificador.

El codificador consta de tres capas Fully-Connected y de dos capas LSTM. Las capas FC convierten los datos de entrada de dimensión 6 en la dimensión 256 de las capas LSTM. De esta manera se expande la complejidad de la información a capturar de la trayectoria. El decodificador presenta una estructura similar pero inversa a la que se añade una capa softmax y una capa de embedding.

Para los vehículos autónomos también es crucial poder predecir trayectorias de peatones y en este contexto se enmarca este otro trabajo de (Alahi et al., 2016) también llevado a cabo con modelos LSTM. Cuando los peatones se encuentran en espacios concurridos se deben evitar mutuamente por lo que es necesario resolver las interacciones entre sus trayectorias. Un simple modelo LSTM no es capaz de resolver estas dependencias. Los autores resuelven esta circunstancia añadiendo una capa llamada Social pooling en la cual las diferentes redes LSTM comparten los estados ocultos h de los peatones vecinos.

En este trabajo de (Violos et al., 2020) vemos una aplicación para la predicción de rumbo de barcos. Los autores utilizan un algoritmo genético para escoger el mejor modelo LSTM de un repositorio de modelos pre-entrenados con diferentes arquitecturas. Lamentablemente no se detallan las arquitecturas empleadas en estos modelos. El objetivo es hacer transfer-learning y optimizar de esta manera el proceso de entrenamiento. En lugar de utilizar directamente la latitud y la longitud como datos de entrada y salida del modelo se ha utilizado la distancia entre posiciones y el rumbo. Así se independiza el problema de predicción de la posición absoluta del barco. Por otro lado los datos son escalados entre 0 y 1 para facilitar el entrenamiento.

En este otro trabajo (Wang et al., 2019) se presenta una aplicación de predicción de trayectorias de personas utilizando la información de geolocalización de sus teléfonos móviles. El objetivo en este caso es anticipar la asignación de recursos de la red 5G y la gestión de la movilidad. Se llevan a cabo dos experimentos. En el primero de ellos se utiliza un modelo LSTM para predecir la localización de un individuo. El modelo se

entrena con trayectorias registradas de ese individuo particular. La entrada del modelo son posiciones en un plano bidimensional. La primera capa es una fully connected de 128 unidades destinada a expandir la dimensionalidad interna del modelo. A continuación se insertan tres capas LSTM en stack, donde la salida de una capa es la entrada de la siguiente . La capa final consta de dos neuronas que vuelven a pasar de dimensión 128 a dimensión 2 y de esta manera dar como salida la predicción de la siguiente posición en el plano.

El resultado se compara con una regresión lineal y no se constatan grandes diferencias en el error cometido para la predicción de la siguiente posición. Sin embargo al llamar al algoritmo recursivamente para hacer predicciones de sucesivas posiciones futuras es cuando el modelo LSTM presenta un rendimiento mucho mejor. En tal caso el error del modelo lineal se incrementa con el tiempo (predicciones más alejadas en el futuro) mientras el modelo LSTM mantiene un error mucho más constante y acotado.

En el segundo experimento se lleva a cabo un modelo para predicciones en una zona geográfica acotada y para multitud de usuarios. En este caso se implementa un modelo tipo Seq2Seq basado también en redes LSTM. Los modelos Seq2Seq permiten traducir una secuencia de entrada en otra secuencia de salida de longitud diferente. La arquitectura es similar a la anterior pero utilizando dos modelos LSTM, una como codificadora y otra como decodificadora. En lugar de utilizar tres redes LSTM en stack se utilizan dos. Esta arquitectura permite obtener la predicción de una secuencia de posiciones sin necesidad de hacer llamadas recurrentes como en el caso del modelo LSTM simple.

Esta arquitectura supera al resto de modelos con los que se compara para el mismo problema: regresión lineal, Supor Vector Regressor, LSTM simple, GRU y modelos Seq2Seq basados en GRU y en Attention.

2.7. Predicción de trayectorias mediante regresión polinómica

En el caso de que la trayectoria a predecir se pueda modelizar mediante una función polinómica de parámetros estáticos se puede aplicar una regresión polinómica sobre los últimos puntos de la trayectoria. El resultado de esta regresión son los coeficientes de la función polinómica que mejor se ajustan al error cuadrático medio de los puntos analizados según podemos ver en (Ostertagová, 2012). Una vez se han obtenido los parámetros se puede aplicar la función obtenida para extrapolar las siguientes posiciones de la trayectoria.

El inconveniente de este método es que está limitado a una función polinómica de un grado determinado y de parámetros estáticos. Si la trayectoria a predecir cambia los parámetros del movimiento este método no va a ser efectivo hasta que se reciban unos cuantos puntos nuevos con los nuevos parámetros.

2.8. Conclusiones del estado del arte

No se ha encontrado constancia de aplicaciones anteriores basadas en inteligencia artificial para predicción de trayectorias de ejes de control de movimiento en máquinas. Actualmente los errores derivados de la sincronización con ejes externos se intentan minimizar reduciendo el tiempo de ciclo de la comunicación. Sin embargo siempre queda un error residual que hasta la fecha se ha tenido que asumir. En otras ocasiones este error se evita utilizando el mismo controlador tanto para eje maestro como para eje esclavo, sin embargo esta opción no siempre es posible.

En este trabajo se propone el uso de técnicas de Inteligencia Artificial. Se pretende así mejorar el error de seguimiento y llevar un paso más allá las prestaciones de este tipo de aplicaciones.

El método de regresión polinómica puede ser aplicable pero está limitado a trayectorias que sigan una funciones polinómicas estáticas. En este trabajo se va a intentar conseguir una predicción en que los parámetros del movimiento cambien tal como sucede con un perfil de movimiento de 7 segmentos.

Como hemos visto hay diferentes aproximaciones para la predicción de trayectorias dependiendo del ámbito de aplicación y el formato de la predicción. Los modelos probabilísticos discretizan de una manera u otra el espacio de resultados por lo que llevan a cabo más bien una clasificación. Cuando se trata de hacer predicciones de valores en coma flotante las redes LSTM son la opción más utilizada y que mejores resultados está dando.

El éxito de esta redes en otros ámbitos hace pensar que también puedan dar buen resultado en la predicción de trayectorias de ejes que se rigen por ecuaciones polinómicas por tramos y la incertidumbre se limita a los momentos de cambio de tramo. Además si estos cambios de tramo siguen unos patrones también pueden llegar a ser aprendidos por la red LSTM de manera que reaccione rápido a ellos.

Capítulo 3

Redes Long Short-Term Memory

En el presente capítulo se va hacer un repaso del marco teórico de las redes Long Short-Term Memory y se va a explicar el motivo de su selección para el problema de predicción de trayectorias que nos ocupa.

3.1. Las Redes Neuronales Recurrentes

Las redes neuronales recurrentes son un tipo de redes neuronales aptas para procesar secuencias de datos de longitud variable. Estas secuencias de datos es habitual que sean series temporales. Un estado interno h actúa como memoria que se actualiza de forma recurrente con cada nueva entrada de la secuencia. Cada paso en que se recibe una nueva entrada se conoce como time step. El estado interno h se actualiza multiplicando el vector de entrada por un matriz de pesos W_{xh} y el valor anterior del estado interno h_{t-1} por otra matriz de pesos W_{hh}. A la suma de ambos productos se le aplica una función no lineal que suele ser la tangente hiperbólica.

$$h(t) = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{3.1}$$

La salida de la red se obtiene de multiplicar el estado oculto por otra matriz de pesos W_{hh}.

$$y(t) = W_{hy}h_t \tag{3.2}$$

Las redes neuronales recurrentes suelen verse afectadas por ciertos efectos negativos en el entrenamiento que pueden dificultar que éste se complete con éxito. Estos efectos

son el desvanecimiento de gradientes (vanishing gradients) o la explosión de gradientes (exploding gradients). El efecto es que los gradientes tienden a cero o a un valor enorme respectivamente. Se producen a la hora de computar el backpropagation de forma iterativa con los mismos valores de pesos para todos los time steps. Otro de los problemas de la redes recurrentes es que tienden a .ºlvidarïnformación conforme va quedando lejana en la secuencia de entrada.

3.2. Las redes Long Short-Term Memory

Las redes Long Short Term Memory (Hochreiter and Schmidhuber, 1997) son una mejora de las redes recurrentes que se crearon con el fin de solucionar las debilidades antes mencionadas. Además del hidden state h se añade otro estado llamado cell state c. Se incorpora una unidad llamada input gate i que controla como se actualiza el estado c. También se añade una unidad llamada output gate que define qué parte de c se actualiza en h. En un trabajo posterior se agregó un tercer vector llamado forget gate f (Gers et al., 1999) que controla que parte de c se debe olvidar.

A continuación se exponen los gates que participan en las redes LSTM y su cometido a grandes rasgos:

- i: input gate. Define cómo se actualiza el estado c
- f: forget gate. Define qué parte se olvida del estado c
- o: output gate. Define que parte de c pasa al estado h

Las tres gates se calculan aplicando la función sigmoide a una combinación lineal de h y de x con unos pesos que se tendrán que optimizar mediante el entrenamiento:

$$i(t) = \sigma(W_{ih}h_{t-1} + W_{ix}x_t) \tag{3.3}$$

$$f(t) = \sigma(W_{fh}h_{t-1} + W_{fx}x_t) \tag{3.4}$$

$$o(t) = \sigma(W_{oh}h_{t-1} + W_{ox}x_t) \tag{3.5}$$

Posteriormente se calculan el estado c y el estado h mediante las siguientes expresiones:

$$c(t) = f_t \circ c_{t-1} + i_t \circ \tanh(W_{ch}h_{t-1} + W_{cx}x_t) \tag{3.6}$$

$$h(t) = o_t \circ \tanh(c_t) \tag{3.7}$$

En la literatura se suele definir g(t) como una cuarta puerta que no recibe ningún nombre particular y cuya expresión es:

$$g(t) = \tanh(W_{ch} h_{t-1} + W_{cx} x_t) \tag{3.8}$$

Con lo cual las expresiones para c(t) y h(t) quedan de la siguiente manera:

$$c(t) = f_t \circ c_{t-1} + i_t \circ g_t \tag{3.9}$$

$$h(t) = o_t \circ \tanh(c_t) \tag{3.10}$$

La introducción del estado interno cell state c(t) calculado en base a las cuatro gates permite reducir significativamente los problemas de vanishing y exploding gradients que presentan las redes neuronales recurrentes simples.

3.3. Tipos de modelos según las entradas y salidas

Podemos distinguir diferentes tipos de modelos recurrentes según si sus entradas y salidas consisten en secuencias de datos o bien en datos únicos.

Podemos tener modelos en que la entrada sea una secuencia de valores de longitud variable y la salida un valor único. Este caso se puede observar en la figura 3.1.

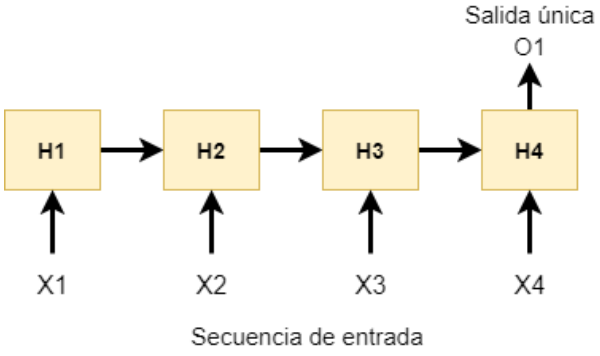


Figura 3.1. *Modelo recurrente con una secuencia de entrada y un valor de salida.*

Fuente: Propia

También podemos tener un único valor de entrada y una salida en forma de secuencia como se puede ver en la figura 3.2. :

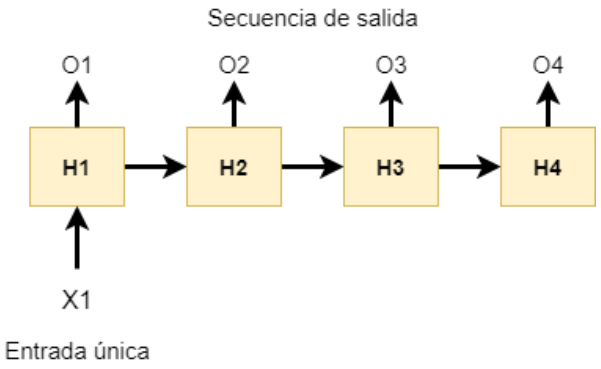


Figura 3.2. *Modelo con un valor único de entrada y una secuencia de salida. Fuente: Propia*

Otro caso posible es que tanto la entrada del modelo como la salida sean secuencias de valores. Este seria el caso de un traductor de texto entre dos idiomas:

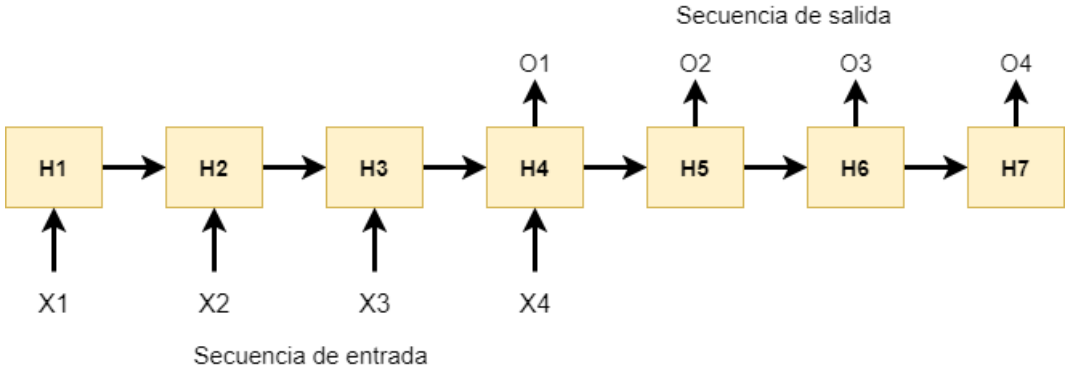


Figura 3.3. *Modelo con una secuencia de entrada y una secuencia de salida. Fuente: Propia*

3.4. Elección del modelo LSTM para el problema

Las redes LSTM han demostrado su eficacia en la predicción de series temporales en diversos campos. En el trabajo de (Fischer and Krauss, 2018) tenemos un ejemplo de aplicación para los mercados financieros. Otra aplicación destacable ha sido el reconocimiento de voz como nos muestra (Sak et al., 2014). En la sección 2.6 se explica cómo se ha utilizado también para la predicción de trayectorias aunque no de ejes industriales sino de barcos.

En el problema de predicción de trayectorias que nos ocupa se va a emplear un modelo

basado en capas LSTM en que la entrada del modelo va a ser una serie corta de posiciones consecutivas de la trayectoria. La salida del modelo será un único valor correspondiente a la predicción de la siguiente posición. Esto corresponde a un modelo del tipo de la figura 3.1.

En el capítulo 5 se detalla la arquitectura del modelo, con sus capas, funciones de activación, número de neuronas y los parámetro de entrenamiento.

Capítulo 4

Objetivos y metodología

El objetivo general de este trabajo consiste en evaluar el rendimiento de un modelo LSTM en la predicción de trayectorias de ejes industriales y comparar dicho rendimiento con un modelo base tipo MLP y con el método de regresión polinómica. El propósito final de esta predicción de trayectorias es la reducción del error de seguimiento en acoplamiento de ejes industriales.

Los modelos de entrenarán específicamente para predecir trayectorias generadas con un perfil de 7 fases como el descrito en la sección 2.1. Dichas trayectorias de entrenamiento se generarán por programa creando un repertorio variado con diferentes valores de parámetros.

Para medir el éxito de este modelo se van a realizar una serie de mediciones. El objetivo para cada una de estas mediciones es alcanzar con el modelo LSTM un resultado mejor que los otros dos modelos de la comparativa:

- Se medirá el error medio y el error máximo del modelo a la hora de hacer predicciones con un conjunto de datos de test no usados durante el entrenamiento. Se verificará que estos errores estén dentro de un rango aceptable.

- Se estudiará la evolución del error según la distancia a un cambio de fase. En cada cambio de fase se produce un cambio en los parámetros de la trayectoria que es previsible que aumente el error de predicción. No obstante el modelo debe ser capaz de reducir este error de seguimiento en los sucesivos pasos temporales. La capacidad de adaptación a los cambios de parámetros de la trayectoria es importante porque es el argumento principal frente a usar un modelo de regresión polinómica.

- Se estudiará el error en cada una de las fases para comprobar como afecta al error el hecho de aplicar una velocidad constante, una velocidad uniformemente acelerada o una aceleración cuadrática.
- Se comprobará como se comporta el error en todo el rango de velocidades y de aceleraciones aplicadas.
- Se compararán todos los puntos anteriores con un modelo MLP que replica al de Payeur et al. (1995). También se comparará con una regresión polinómica.
- Se repetirá el experimento añadiendo ruido a los datos de las trayectorias para comprobar como afecta eso a cada uno de los modelos. Según el trabajo de Payeur et al. (1995) la robustez al ruido es una de las ventajas mas destacadas de usar redes neuronales en la predicción de trayectorias.

Capítulo 5

Descripción del experimento

En este capítulo se va a detallar cómo se han generado los datos de las trayectorias que después se van utilizan tanto para el entrenamiento como para la verificación de rendimiento. Se va a describir el diseño de los modelos aplicados y cómo se ha llevado a cabo el proceso de entrenamiento.

5.1. Generación de los datos de entrenamiento

Para el entrenamiento del modelo se requieren muchos ejemplos de trayectorias del tipo que se pretenden predecir. Estas trayectorias no se van a adquirir de aplicaciones reales sino que se van a generar programáticamente utilizando las funciones polinómicas correspondientes.

5.1.1. La función de generación de trayectorias

Se ha implementado una función en Python que genera un perfil de movimiento trapezoidal de 7 fases en base a los parámetros de jerk, aceleración máxima y velocidad máxima. En la figura 5.1 se muestran dos ejemplos de perfiles de movimiento generados mediante dicha función y con diferentes valores de parámetros.

Para la generación de la secuencia de posiciones en cada fase se emplea la siguiente fórmula:

$$x(t) = \frac{1}{6}jt^3 + \frac{1}{2}a_0t^2 + v_0t + x_0 \tag{5.1}$$

Donde j es el jerk o tasa de incremento de la aceleración, a_o, v_0 y x_0 son la aceleración,

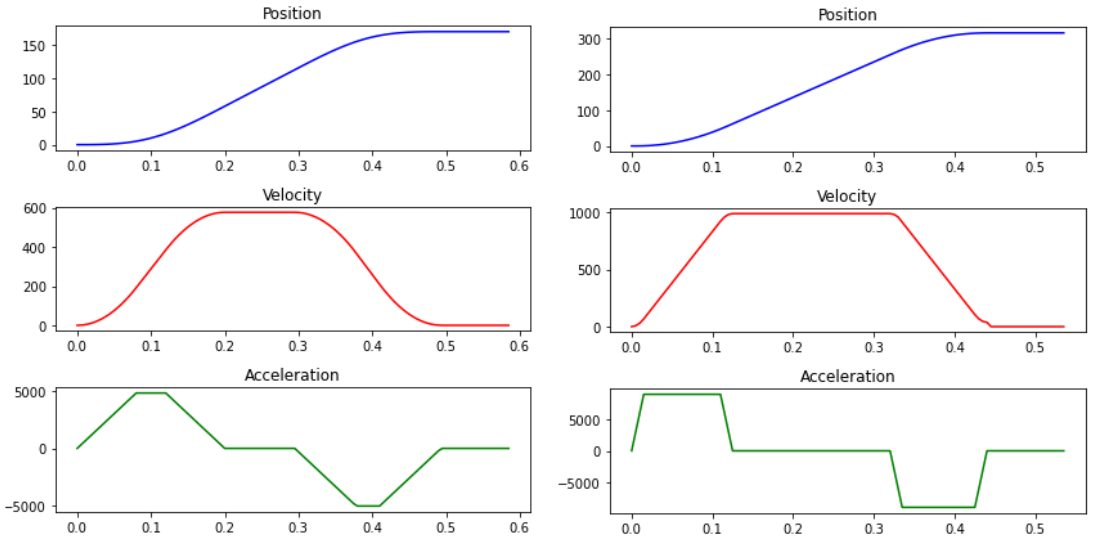


Figura 5.1. *Ejemplo de dos perfiles de movimiento con diferentes parámetros. Fuente: Propia*

la velocidad y la posición en el instante inicial de la fase.

También se han calculado la velocidad y la aceleración mediante las fórmulas:

$$v(t) = \frac{1}{2}jt^2 + v_0 t \tag{5.2}$$

$$a(t) = jt + a_0 \tag{5.3}$$

Las trayectorias se pueden generar en sentido positivo o en sentido negativo. Esto se controla mediante un parámetro booleano. Para generar movimientos negativos se multiplican por -1 los valores de velocidad, aceleración y jerk.

La función de generación de trayectorias devuelve tres arrays con los valores de posición, velocidad y aceleración calculados en intervalos de tiempo de 0.005 segundos. Esta resolución temporal está dentro del rango en que suelen trabajar los controladores de movimiento industriales.

También se dispone un array que guarda en el mismo intervalo de tiempo a qué fase se encuentra la trayectoria del 1 al 8. La octava fase se corresponde al estado de reposo después del movimiento. Este array nos permitirá comparar el error en la predicción por cada una de las fases.

En otro array se memoriza la distancia a un cambio de fase. En el instante en que se cambia de fase este valor valdrá 0. En el siguiente intervalo de tiempo el valor valdrá 1.

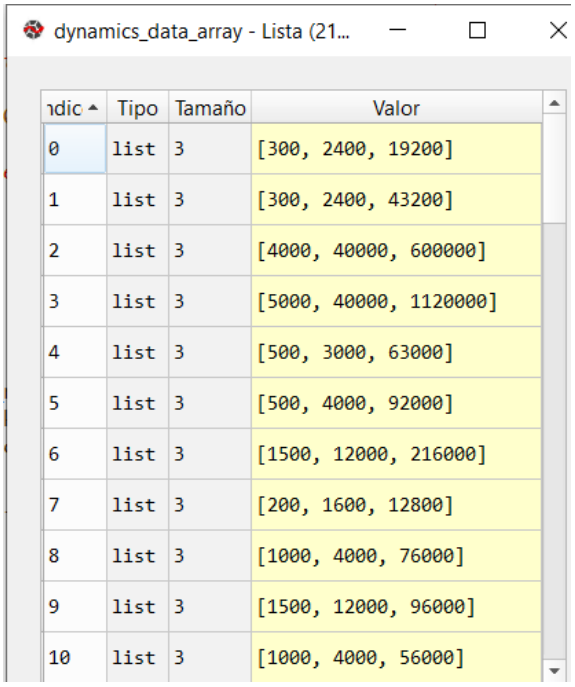
El valor se irá incrementando hasta llegar a 7 momento en el cual se mantendrá constante porque la distancia deja de ser relevante. Este array nos permitirá ver como se comporta el error ante un cambio de fase. Es esperable que el error aumente considerablemente en el momento del cambio de fase ya que es un evento impredecible.

5.1.2. Concatenación de trayectorias variadas

Para un correcto entrenamiento se considera necesario obtener una gran cantidad de trayectorias con valores de parámetros diversos dentro del rango normal para cada parámetro.

Se utilizarán velocidades entre 200 mm/s y 5000 ms. El valor de aceleración estará comprendido entre 4 y 12 veces la velocidad con un paso de 2. El valor de jerk será de entre 5 y 50 veces la aceleración con un paso de 5.

En primer lugar se creará un array con todas las posibles combinación de parámetros en orden creciente. Después se aplicará una reordenación aleatoria. en la figura 5.2 se puede apreciar un ejemplo de este array de parámetros:



ıdic	Tipo	Tamaño	Valor
0	list	3	[300, 2400, 19200]
1	list	3	[300, 2400, 43200]
2	list	3	[4000, 40000, 600000]
3	list	3	[5000, 40000, 1120000]
4	list	3	[500, 3000, 63000]
5	list	3	[500, 4000, 92000]
6	list	3	[1500, 12000, 216000]
7	list	3	[200, 1600, 12800]
8	list	3	[1000, 4000, 76000]
9	list	3	[1500, 12000, 96000]
10	list	3	[1000, 4000, 56000]

Figura 5.2. *Array de parámetros de movimiento. Fuente: Propia*

Una vez se tienen definidos los parámetros se genera una concatenación de trayectorias con los mismos. En cada parámetro se introduce una variabilidad aleatoria adicional. La elección de si el sentido del movimiento es positivo o negativo también es aleatoria. Con

la adición de esta componente aleatoria se pretende evitar que el modelo se adapte a trayectorias concretas en el entrenamiento y que consecuentemente mejore su capacidad de generalización.

En la figura 5.3 se puede observar un ejemplo de concatenación de varios movimientos. Podemos ver como la aceleración alcanza diferentes valores máximos y adopta diferentes pendientes que vienen dadas por el jerk. La velocidad también alcanza diferentes máximos. La curvatura de la velocidad es también consecuencia del jerk. Cuanto menor es el jerk más curvatura adopta la velocidad en las transiciones. Por el contrario un jerk más elevado implica cambios mas agudos en la velocidad producidos por saltos más abruptos en la aceleración. El objetivo final es tener movimientos variados dentro de un margen.

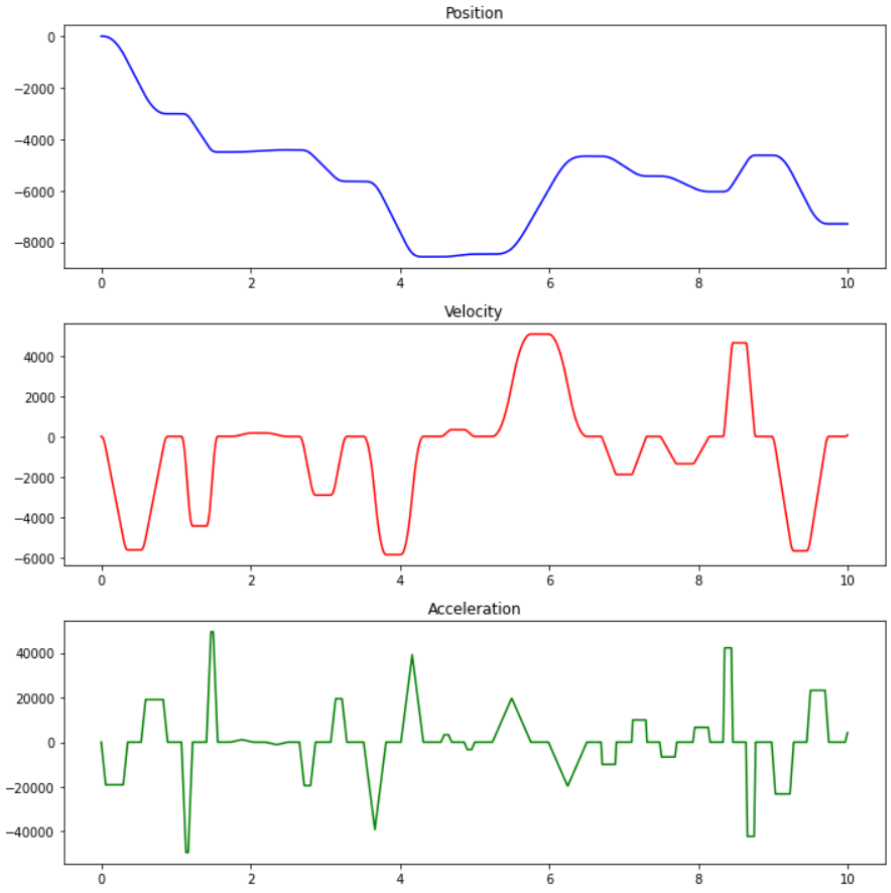


Figura 5.3. *Ejemplo de varios movimientos concatenados. Fuente: Propia*

5.1.3. Preparación de los datos para el entrenamiento

Es importante que las predicciones del modelo sean independientes de las posiciones absolutas que además pueden estar en un rango muy amplio y son diferentes para cada

aplicación. Sería muy costoso si tuviéramos que preocuparnos de la variabilidad de posición en el entrenamiento además de la variabilidad de velocidad, aceleración y jerk.

Inspirado por trabajos anteriores como el de (Payeur et al., 1995) y el de (Violos et al., 2020) donde esta estrategia ha demostrado mejores resultados, se ha optado por alimentar al modelo con incrementos de posiciones en lugar de posiciones absolutas. Por tanto el array con secuencias de posiciones se ha sustituido por un array en que se calcula la diferencia entre cada posición y la posición anterior. Estos arrays se pueden observar en la figura 5.4. El objetivo del modelo será predecir el siguiente incremento de posición y no la posición absoluta. A la hora de hacer inferencia también habrá que hacer esta conversión en el preprocesado y la transformación inversa en el postprocesado para volver a pasar a posiciones absolutas.

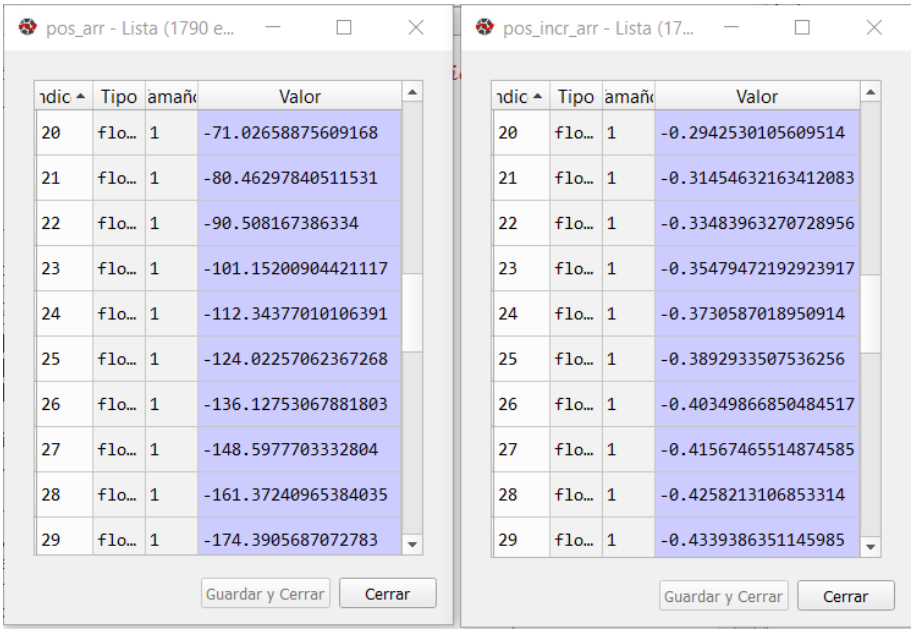
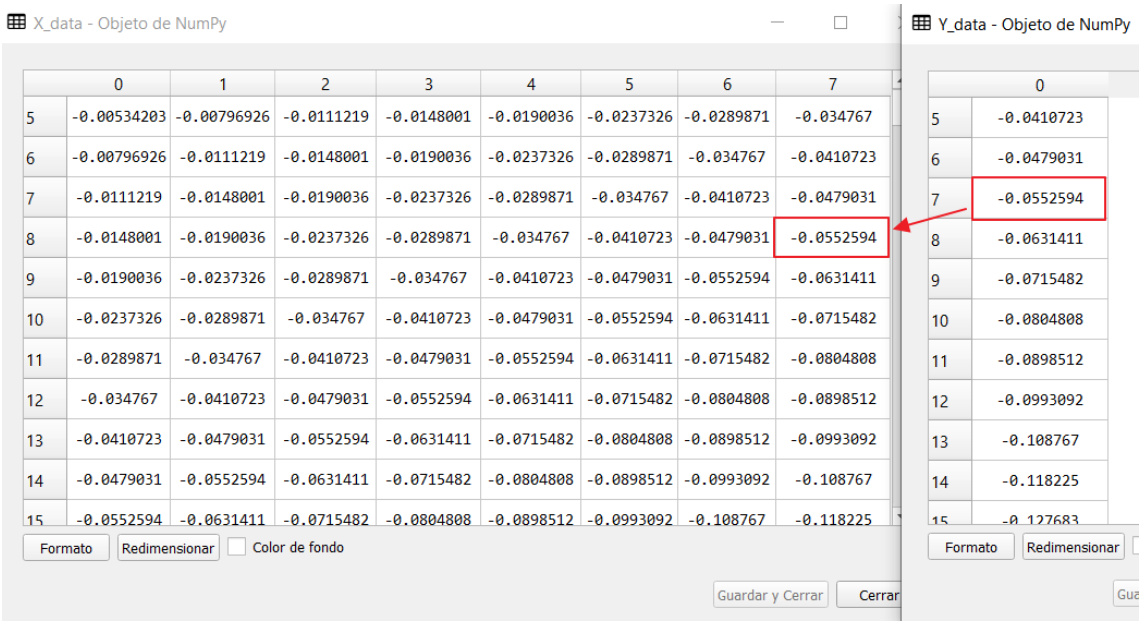


Figura 5.4. *Posiciones absolutas e incrementos de posición. Fuente: Propia*

Una vez obtenidos los incrementos es conveniente normalizarlos para que se ajusten bien a las funciones de activación de los modelos. El mayor valor de incremento de posición, dado la máxima velocidad de 5000 mm/s con su variabilidad adicional y el tiempo de ciclo de 0.005 segundos está acotado a 30 mm. Por tanto se dividen todas las posiciones por 30 para obtener valores entre 0 y 1.

Finalmente para llevar a cabo el entrenamiento se divide toda la secuencia de posiciones en ventanas de longitud 8 que se utilizarán como entrada del modelo y se toma la siguiente posición de cada ventana como valor objetivo a predecir. El valor de 8 es suficiente para

que el modelo pueda estimar la velocidad, aceleración y jerk de la trayectoria. El valor mínimo de puntos para determinar una función cúbica es de 4. En la figura 5.5 se pueden observar las secuencias de datos de entrenamiento de longitud 8. Como se puede apreciar el valor de etiqueta asociado a cada secuencia de entrada pasará a formar parte de la siguiente secuencia de entrenamiento en último lugar.



	0	1	2	3	4	5	6	7
5	-0.00534203	-0.00796926	-0.0111219	-0.0148001	-0.0190036	-0.0237326	-0.0289871	-0.034767
6	-0.00796926	-0.0111219	-0.0148001	-0.0190036	-0.0237326	-0.0289871	-0.034767	-0.0410723
7	-0.0111219	-0.0148001	-0.0190036	-0.0237326	-0.0289871	-0.034767	-0.0410723	-0.0479031
8	-0.0148001	-0.0190036	-0.0237326	-0.0289871	-0.034767	-0.0410723	-0.0479031	-0.0552594
9	-0.0190036	-0.0237326	-0.0289871	-0.034767	-0.0410723	-0.0479031	-0.0552594	-0.0631411
10	-0.0237326	-0.0289871	-0.034767	-0.0410723	-0.0479031	-0.0552594	-0.0631411	-0.0715482
11	-0.0289871	-0.034767	-0.0410723	-0.0479031	-0.0552594	-0.0631411	-0.0715482	-0.0804808
12	-0.034767	-0.0410723	-0.0479031	-0.0552594	-0.0631411	-0.0715482	-0.0804808	-0.0898512
13	-0.0410723	-0.0479031	-0.0552594	-0.0631411	-0.0715482	-0.0804808	-0.0898512	-0.0993092
14	-0.0479031	-0.0552594	-0.0631411	-0.0715482	-0.0804808	-0.0898512	-0.0993092	-0.108767
15	-0.0552594	-0.0631411	-0.0715482	-0.0804808	-0.0898512	-0.0993092	-0.108767	-0.118225

	0
5	-0.0410723
6	-0.0479031
7	-0.0552594
8	-0.0631411
9	-0.0715482
10	-0.0804808
11	-0.0898512
12	-0.0993092
13	-0.108767
14	-0.118225
15	-0.127683

Figura 5.5. *Secuencias de valores de entrada y valor objetivo. Fuente: Propia*

5.2. Diseño del modelo LSTM

Haciendo uso de los frameworks Keras y Tensorflow se ha diseñado un modelo cuyo elemento principal es una primera capa de tipo LSTM. En la figura 5.6 se puede ver un resumen de la arquitectura del modelo.

5.2.1. Capas del modelo

Para la capa LSTM se han testeado diferentes cantidad de neuronas y con 64 se han obtenido los mejores resultados.

La segunda capa es la de salida. Es de tipo Dense con tan sólo una neurona y una función de activación lineal. Esta neurona por tanto devolverá una combinación lineal de las salidas de las unidades LSTM. Durante el entrenamiento esta salida se intentará ajustar al valor de deseado mediante la retropropagación de los gradientes.

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 64)                16896
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 65
=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

Figura 5.6. *Arquitectura del modelo basado en LSTM. Fuente: Propia*

5.2.2. Número de parámetros

El número de parámetros de la capa LSTM (Karakaya Murat, 2020) viene dado por la siguiente fórmula:

$$Parametros = 4((m+n)n+n) \tag{5.4}$$

Donde m es la dimensión de los datos de entrada (en este caso 1) y n es el tamaño de la capa LSTM. La multiplicación por cuatro es debida a que la unidad LSTM incorpora cuatro unidades densas. Aplicando dicha fórmula obtenemos los 16896 parámetros que se observan en el resumen del modelo. La capa Fully-Conneced consta de una única neurona con 64 entradas. Sumando el bias obtenemos los 65 parámetros de este capa.

5.2.3. Funciones de activación

Para la capa LSTM se han testado las funciones de activación tanh, linear y relu. Las diferencias en el resultado no han sido muy grandes pero la linear y la relu han funcionado algo mejor que la tanh. Finalmente se ha optado por utilizar la función lineal por ser la que menor valor de coste ha dado tras el entrenamiento.

Normalmente en las funciones de activación de las redes neuronales se busca introducir una no-linearidad. En el caso particular de las capas LSTM esta no-linearidad ya está incorporada en su función de activación para la recursividad que por defecto es la sigmoid en Keras (Keras API Reference, 2021). La función de activación por defecto en Keras es la tanh pero se ha cambiado por la función lineal.

5.3. Diseño del modelo MLP

Para la comparativa entre el modelo de (Payeur et al., 1995) y el resto de modelos contamos con los datos que reporta dicho trabajo pero en algunos casos son un tanto confusos y poco detallados. Por tanto se toma la decisión de implementar un modelo MLP con la misma arquitectura y testearlo con los mismos datos de entrenamiento y validación que el resto. En la figura 5.7 se puede observar la arquitectura de dicho modelo que consta de dos capas ocultas de 20 neuronas cada una y una capa de salida de una única neurona. Se utiliza la función de activación sigmoid.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 20)                180
_________________________________________________________________
dense_2 (Dense)              (None, 20)                420
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 21
=================================================================
Total params: 621
Trainable params: 621
Non-trainable params: 0
_________________________________________________________________
```

Figura 5.7. *Resumen del modelo MLP que replica el de Payeur et al. Fuente: Propia*

La única diferencia con el modelo original es el tamaño de los datos de entrada. En lugar de utilizar secuencia de 6 valores como en el modelo original se han utilizado secuencia de 8 valores como en el modelo LSTM. Esto no se espera que tenga un impacto significativo en el resultado final y facilita que se comparen los modelos en igualdad de condiciones.

5.4. Diseño del modelo de regresión polinómica

La regresión polinómica es un método numérico que permite encontrar un polinomio de un grado determinado que mejor ajusta el error cuadrático medio a una serie de puntos. Una vez obtenidos los coeficientes del polinomio con una serie de puntos de la trayectoria se aplicará extrapolación para obtener el valor correspondiente al siguiente paso temporal.

Se ha empleado la función polyfit() de la libreria numpy (Community, 2021). Se ha optado por utilizar un polinomio de grado 3 para modelizar la trayectoria. En lugar de utilizar 8 puntos como en los otros modelos se han utilizado tan sólo los cuatro últimos puntos de cada secuencia. Cuatro son los puntos mínimos para poder encontrar un poli-

nomio de grado 3. Si se utilizan más puntos entonces la regresión seria más robusta frente al ruido pero el modelo perdería capacidad para ajustarse a cambios en la trayectoria por dar más peso a valores pasados.

Como valores para la variable independiente no se ha tomado el tiempo real sino simplemente la secuencia 1,2,3,4. Por tanto la siguiente posición a predecir corresponderá al valor 5 en la secuencia.

Para cada predicción es necesario llamar de nuevo a la función polyfit() para encontrar un nuevo polinomio que se ajuste a la última secuencia de puntos. Una vez obtenido el polinomio se calcula el valor que devuelve dicho polinomio para el valor 5 de la variable independiente.

5.5. Entrenamiento del modelo

En la sección 5.1 se explicó como se generaron y se prepararon los datos para el entrenamiento.

Se han generado una secuencia de en torno a 150.000 puntos que se han dividido en unos 120.000 para entrenamiento y 30.000 para la validación.

5.5.1. Parámetros del entrenamiento

Al tratarse de un problema de regresión se ha optado por una función de coste Mean Square Error (MSE) dado que dicha función va a penalizar errores grandes entre la predicción y la etiqueta. Como métrica a observar se ha seleccionado el error absoluto medio (MAE) y el error absoluto porcentual (MAPE).

La métrica MAPE es interesante porque relativiza el error respecto al valor absoluto de incremento de posición pero presenta el problema de que no se puede calcular cuando el incremento de posición es cero, y este es un caso que se da frecuentemente en la aplicación.

Se han testeado diferentes algoritmos de optimización para el entrenamiento como el SGD, el adam y el RMSprop. Finalmente el algoritmo adam ha sido el seleccionado por aportar mejores resultados.

También se han probado diferentes valores de batchsize. Finalmente se ha escogido de batch size de 16 que, aunque produce un entrenamiento más lento que un batch size menor, da mejores resultados en el valor de loss y validation loss.

Como es habitual que el mejor resultado no se de en la última epoch del entrenamiento

se ha empleado una función de callback para guardar los parámetros del modelo en un archivo cada vez que mejora el valor de validation loss. De esta manera siempre se conserva el mejor modelo de entre todas las epochs del proceso de entrenamiento.

En la figura 5.8 se puede apreciar la evolución del coste (loss) y del coste de validación.

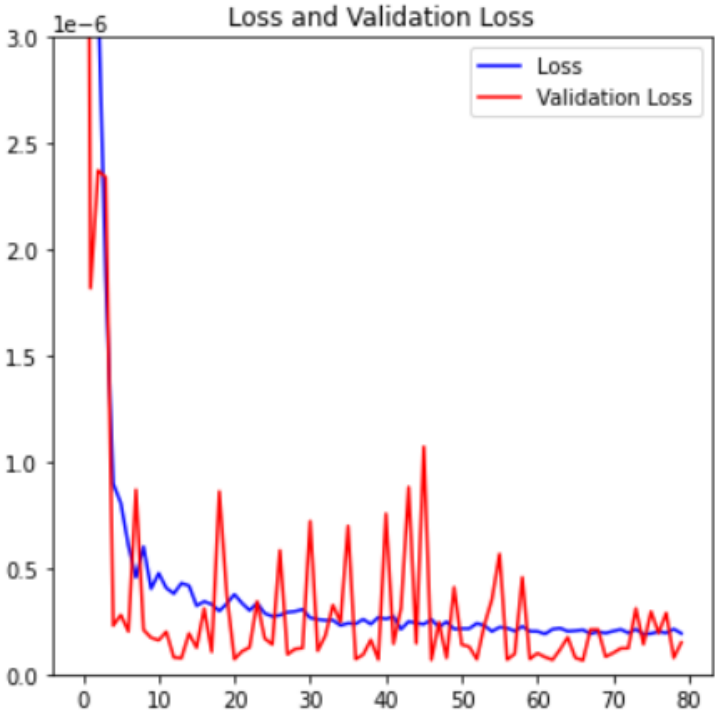


Figura 5.8. *Curva de evaluación del loss y el validation loss. Fuente: Propia*

Capítulo 6

Descripción de los resultados para el modelo LSTM

6.1. Estudio del error del modelo

6.1.1. Distribución del error absoluto

Una vez entrenado el modelo, se aplicación predicciones sobre todo el conjunto de validación y se mide el error absoluto de cada predicción. Como se puede observar de la tabla 6.1 Se obtiene una error absoluto medio de 0.0022 mm y un error absoluto máximo de 0.18 mm lo cual a priori parece bastante satisfactorio. Recordemos que se trata de predicciones de incrementos de posición y el incremento máximo considerado es de 30 mm.

	Normalizado	Absoluto (mm)	Relativo a 30 mm (%)
Error medio	7,3e-5	0.0022	0,007
Error máximo	0,006	0.18	0,6

Tabla 6.1. *Tabla de errores medios y máximos. Fuente: Propia.*

En la figura 6.1 obtiene el histograma del error absoluto y se aprecia una altísima concentración de frecuencias en los bins correspondientes a los errores más bajos. El resto de frecuencias es inapreciable en escala lineal por lo que se gráfica también el histograma en escala logarítmica.

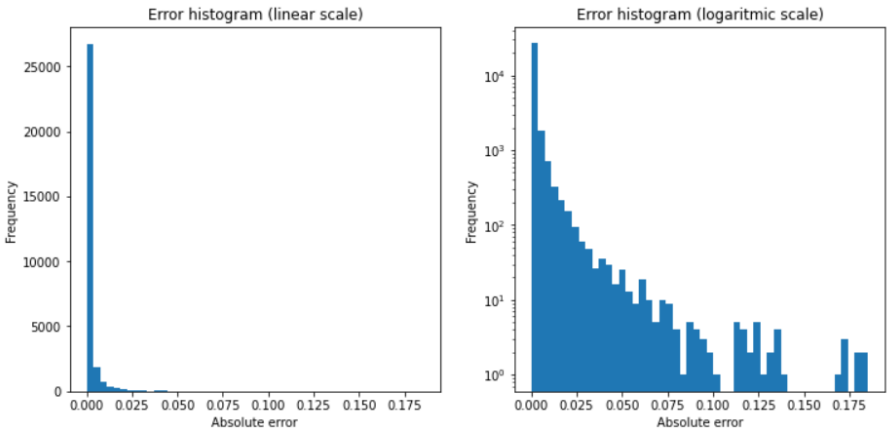


Figura 6.1. *Histograma del error en escala lineal y logarítmica. Fuente: Propia*

6.1.2. El error según la distancia al cambio de fase

Es interesante observar la distribución del error según la distancia al instante del cambio de fase. Recordamos que en momento de cambio de fase es cuando cambian los parámetros del movimiento y es por tanto cuando el modelo tiene menos información para la predicción de la siguiente posición. En ese instante se espera un incremento en el error. A medida que se aleja el instante del cambio de fase el modelo puede adaptarse a los nuevos parámetros y reducir el error en la predicción.

En la figura 6.2 se puede observar la gráfica en forma de diagrama de bigotes para la distancia al cambio de fase con dos escalas diferentes para el eje y.

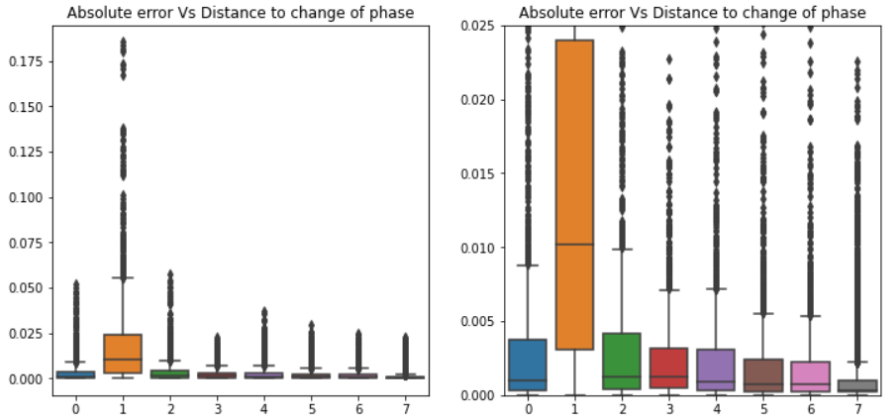


Figura 6.2. *Error según la distancia al cambio de fase. Fuente: Propia*

Se puede apreciar que los mayores errores se concentran a una distancia de 1 del cambio de fase. En el instante 0 se produce el cambio de fase y el error es algo más elevado que

la media pero no demasiado. En el instante 1 el modelo ya ha detectado el cambio de fase pero le falta información para adaptarse al cambio y parece que sobrereacciona en la predicción. En el instante 2 el error vuelve a bajar a un valor más próximo a la media. De ahí en adelante el error va bajando progresivamente. El valor de distancia 7 en x realmente corresponde a las distancias mayores o iguales a 7.

6.1.3. El error según la fase del movimiento

También es interesante ver cómo se comporta el error en cada fase del movimiento. Esto lo podemos ver en la figura 6.3.

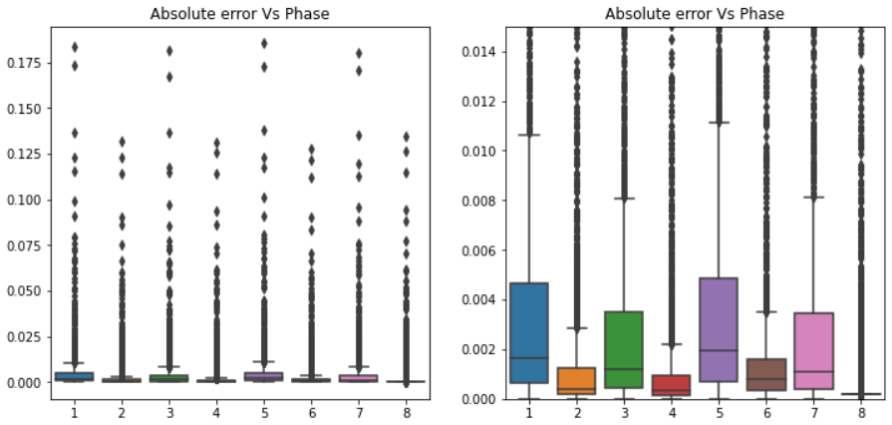


Figura 6.3. *Distribución del error según la fase. Fuente: Propia*

Se puede apreciar como el error es mayor en las fases 1,3,5 y 7. En dichas fases el movimiento sigue una evolución polinómica de grado 3. Son las fases en que la trayectoria es más compleja ya que la aceleración no es constante. En cambio el error es menor en las fases 2,4 y 6 donde hay la aceleración es constante y por tanto la trayectoria es grado 2 y por tanto menos compleja. Finalmente es puede observar que el error es mínimo para la fase 8 que corresponde con los momentos de reposo.

6.1.4. El error según la velocidad y la aceleración

En la gráfica 6.4 se puede observar la relación entre el error y la velocidad y entre el error y la aceleración. No se observa una relación entre ambas. Sí que se aprecia una serie de valores anómalos sobre la velocidad cero. En la gráfica del error contra la aceleración también se observan una serie de anomalías distribuidas en varias rectas de diferente pendiente.

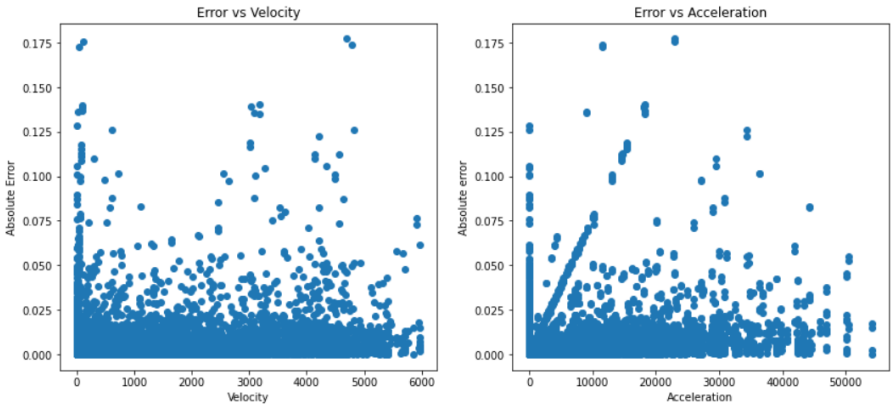


Figura 6.4. *El error vs la velocidad y la aceleración. Fuente: Propia*

El valor de estas anomalías se sitúa por encima de 0.05. Viendo la gráfica de la figura 6.2 vemos que los errores por encima de 0.05 son los producidos a una distancia de 1 del cambio de fase y son debidos a la sobrereacción del modelo ante éste.

Los valores anómalos situados sobre la velocidad cero son los más perjudiciales de todos. En velocidades próximas a cero los incrementos de posición también son próximos a cero, por tanto el impacto de un error determinado en esta zona es mucho mayor. Además el valor absoluto de los errores anómalos en esta zona alcanza en alguna ocasión su valor máximo que corresponde a 0.18. Esto provocaría un salto abrupto del eje de 0.18 mm cuando realmente debería permanecer quieto. Esto es claramente un punto a mejorar del modelo.

6.1.5. Observación de las anomalías en la curva de las predicciones

En las secciones anteriores se han mencionados unas anomalías en las predicciones. En la gráfica de la figura 6.5 se ha dibujado la curva de las predicciones y la curvas de las etiquetas. Una curva no se distingue de la otra porque están sobrepuestas. Las anomalías no son observables en la escala de esta gráfica.

Sin embargo si ampliamos la escala en Y de la gráfica sobre la zona próxima a cero las anomalías se hacen distinguibles como se puede apreciar en 6.6

En la gráfica 6.7 se observa un error anómalo ampliado. Se puede apreciar como la curva de predicciones lleva a cabo una oscilación amortiguada para ajustarse al cambio abrupto de la dinámica del movimiento. Al finalizar dicho transitorio queda un error constante remanente.

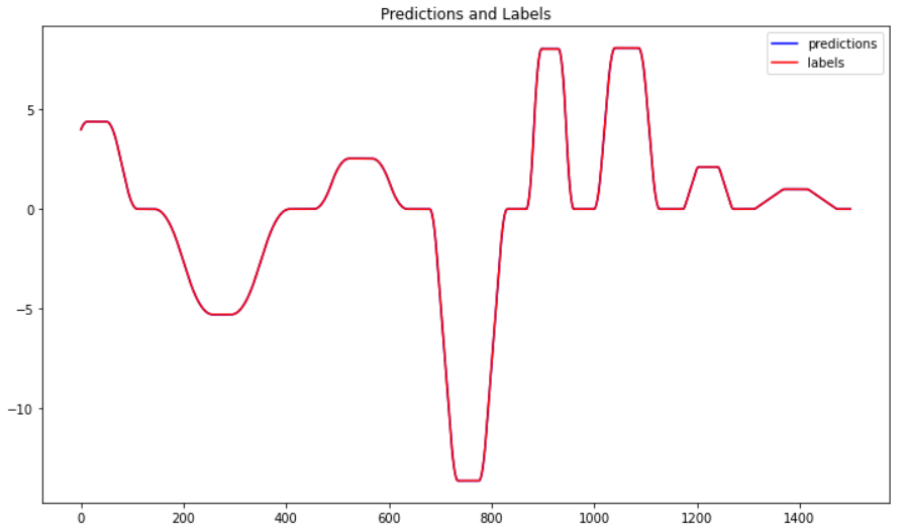


Figura 6.5. *Gráfica de predicciones y etiquetas. Fuente: Propia*

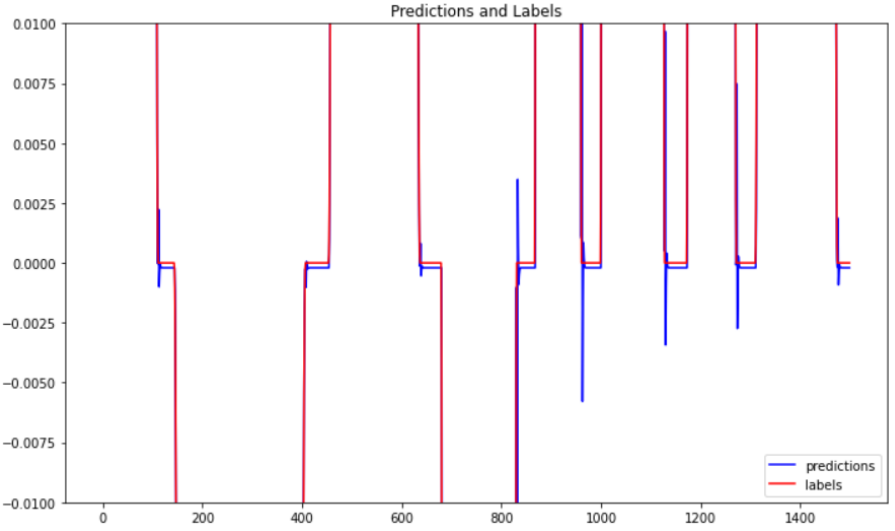


Figura 6.6. *Gráfica de predicciones y etiquetas ampliada. Fuente: Propia*

6.2. Reducción del error de seguimiento aportada por el modelo

En una aplicación de acoplamiento de ejes en que no se tiene control sobre el eje maestro el error de seguimiento mínimo que podemos esperar siendo optimistas es igual al último incremento de posición producido entre dos ciclos de comunicación. Esto es así porque el controlador esclavo recibe la posición actualizada del maestro hasta transcurrido el siguiente tiempo de ciclo.

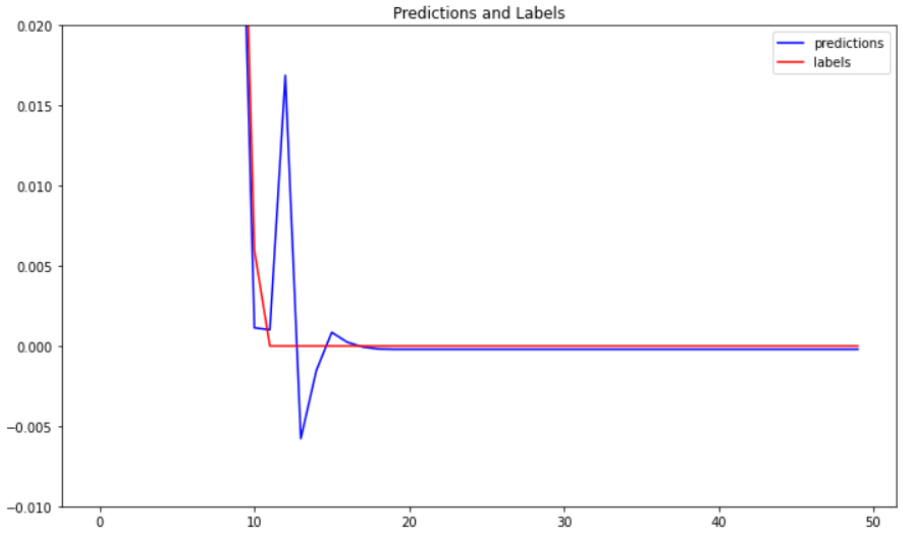


Figura 6.7. *Oscilación amortiguada del error ante cambio brusco de trayectoria.*

Fuente: Propia

Haciendo la suposición que el error de seguimiento en tal caso es exactamente igual al último incremento de posición se va a determinar la mejora que supone la incorporación del modelo.

Capítulo 7

Comparativa entre los modelos LSTM, el MLP de Payour y Regresión

En este capitulo se va a comparar el modelo LSTM desarrollado en este trabajo con dos modelos más.

Por un lado se va a comparar con un modelo de tipo Multilayer Perceptron. que replica al utilizado por de (Payeur et al., 1995). Se ha escogido este modelo para la comparativa por tratarse del trabajo que más se asemeja al presente trabajo. Además la modelización de la trayectoria a predecir y los rangos de velocidad, aceleración y jerk están en el mismo orden.

Por otro lado también vamos a incluir en la comparación un método de regresión polinómica. Este es un método numérico a diferencia de los anteriores que forman parte del aprendizaje automático. De esta manera será posible comparar ambas familias de métodos.

Se van a llevar a cabo para los tres modelos las mismas observaciones que se hicieron para el modelo LSTM en el capítulo 6. Esto se va a hacer en primer lugar utilizando los datos originales de validación y posteriormente añadiendo un porcentaje de ruido aleatorio a los datos de posición. Se observará que el comportamiento de los modelos es muy diferente en una circunstancia y en la otra.

7.1. Resultado de la comparativa con datos sin ruido

En primer lugar se ensayan los tres modelos señalados haciendo predicciones sobre el mismo conjunto de datos de validación. En este experimentos los valores de posición se mantienen inalterados.

En las condiciones expuestas el modelo MLP presenta un error medio de 0,013 mm frente a los 0,0022 mm de los modelos LSTM y la regresión. Esto corresponde a un error seis veces mayor. El error máximo del MLP es de 0,39 mm frente a 0,18 mm del LSTM y 0,14 mm de la regresión. En el trabajo original de (Payeur et al., 1995) se mencionaba un error de 0,45 mm sin especificar si se trataba de error medio o máximo. En vista de los resultados del modelo replicado se deduce que era el error máximo.

En la tabla 7.1 se muestra una comparativa de resultados y de algunas características relevantes de los modelos.

	Modelo LSTM	Regresión	Modelo MLP
Error medio absoluto (mm)	0.0022	0.0022	0.013
Error máximo absoluto (mm)	0,18	0,14	0.39
Error medio respecto incremento máximo (%)	0.007 %	0.007 %	0.043 %
Incremento máximo contemplado (mm)	30	30	30
Longitud de la secuencia de entrada	8	4	8
Número de parámetros	16896	-	621

Tabla 7.1. *Diferencias entre los modelos. Fuente: Propia.*

Por tanto atendiendo a estos datos parece que el método que da un mejor rendimiento de momento es el de regresión polinómica.

En la figura 7.1 podemos comparar el histograma del error de los modelos LSTM, de la regresión polinómica y del modelo MLP. La amplitud de la distribución para el modelo MLP es sensiblemente mayor que para los otros dos.

Se aprecia que el error máximo de la regresión es algo menor que el del modelo LSTM pero por otro lado hay más frecuencias de estos errores máximos.

Al observar el error respecto a la distancia al cambio de fase en la figura 7.2 apreciamos notables diferencias entre los modelos. El modelo LSTM presenta un incremento notable

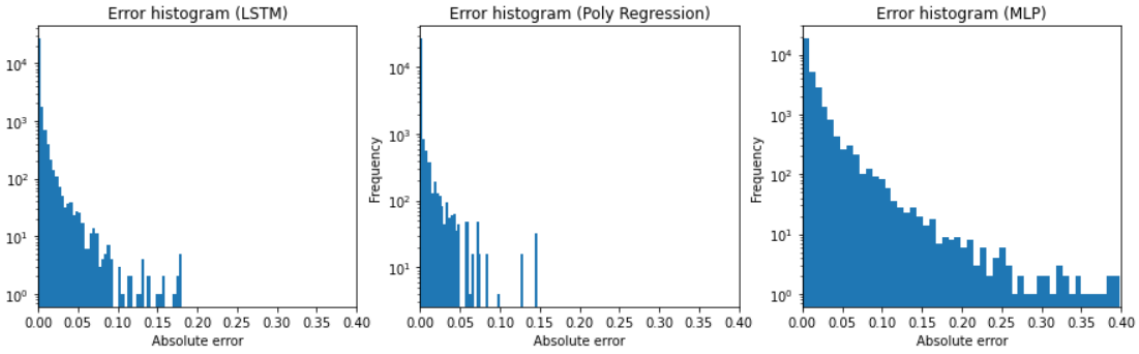


Figura 7.1. *Histogramas del error para los modelos. Fuente: Propia*

sólo a una distancia de 1 y a partir de ahí corrige rápidamente el error. El modelo de regresión mantiene un error elevado entre las distancias 1 y 2. A partir de la distancia 3 presenta un error mucho menor que el resto. El modelo MLP tiene un error mucho mayor a todas las distancias estando el pico en la distancia 2.

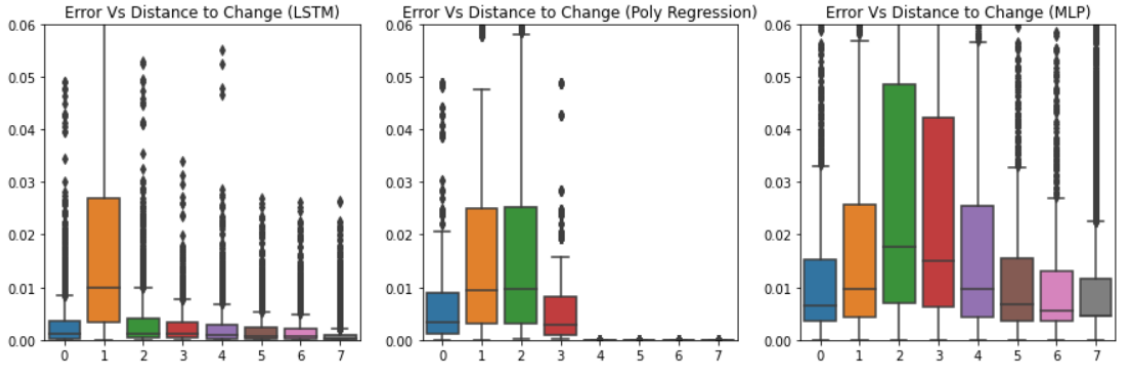


Figura 7.2. *Error frente a la distancia del cambio de fase. Fuente: Propia*

En cuanto a adaptación al cambio de fase el modelo LSTM es más eficaz que el resto y el MLP es bastante peor.

En la figura 7.3 podemos observar el error por fase para los modelos en estudio

Tanto en el modelos LSTM como en el caso de la regresión se presentan errores mayores en las fases en que la aceleración cambia linealmente que en las fase en que la aceleración es constante. Esto es debido a que en esos casos la trayectoria es más compleja. En el caso del modelo MLP podemos apreciar que no tiene una distribución clara a la que se puede dar una explicación. De hecho ante diferentes entrenamientos se pudo observar que el error por fase variaba. El método de regresión presenta un error menor para cada una de la fases.

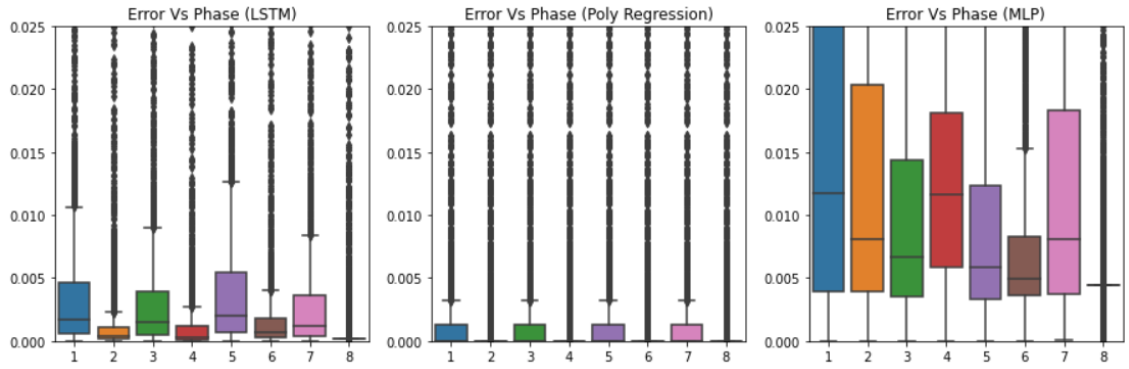


Figura 7.3. *Errores por fase de los modelos LSTM, regresión y MLP. Fuente: Propia*

En la gráfica 7.4 podemos comparar el comportamiento del error respecto a la velocidad en los tres modelos. Podemos apreciar cómo la distribución es similar pero con mayor rango de error para el modelos MLP que para los otros dos.

Figura 7.4. *Error respecto la velocidad de los modelos LSTM, regresión y MLP. Fuente: Propia*

Se puede apreciar como los errores anómalos en velocidades próximas a cero están presentes en los tres modelos

Finalmente en la figura 7.5 observamos el error respecto a la aceleración. También en este caso las gráficas son similares para los tres modelos difiriendo tan sólo en el rango del error del modelo MLP. En todos los casos se aprecian diversas correlaciones lineales entre el error y la aceleración.

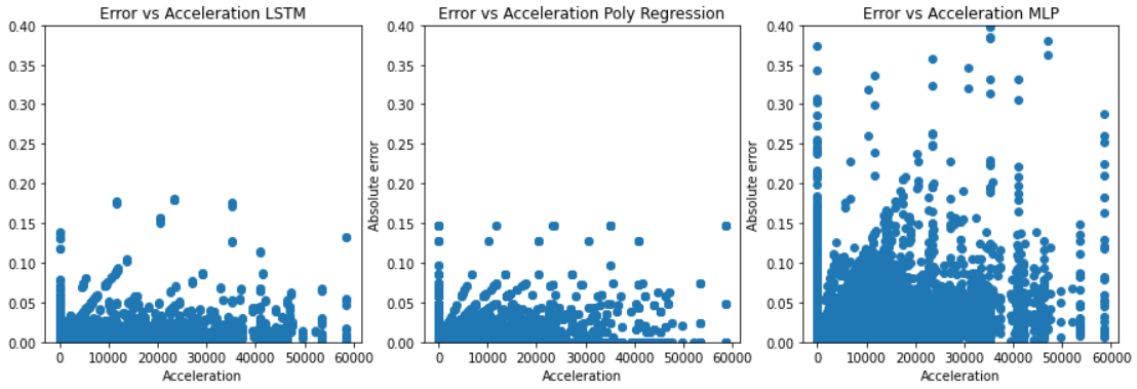


Figura 7.5. *Error respecto la aceleración de los modelos LSTM, regresión MLP. Fuente: Propia*

7.2. Resultado de la comparativa con ruido añadido a los datos

En el trabajo de (Payeur et al., 1995) se menciona que el uso de redes neuronales mejora el comportamiento ante el ruido de los métodos tradicionales de regresión o interpolación. Con objeto de verificar este hecho se van a repetir las observaciones anteriores añadiendo un ruido aleatorio de 0,5 % a los valores de posición del conjunto de datos de validación.

En la tabla 7.2 vemos los valores de error medio y máximo para cada uno de los modelos haciendo predicciones con ruido añadido.

	Modelo LSTM	Regresión	Modelo MLP
Error medio absoluto (mm)	0.065	0.093	0.024
Error máximo absoluto (mm)	1,23	1,85	0.43
Error medio respecto incremento máximo (%)	0.21 %	0.31 %	0.08 %

Tabla 7.2. *Diferencias entre los modelos con 0,5 % de ruido. Fuente: Propia.*

Como podemos apreciar los resultados han cambiado notablemente ahora que se están haciendo predicciones con ruido. El modelo que presenta un menor error medio y máximo con diferencia es el MLP. El incremento de error que ha experimentado es muy leve. En cambio tanto el modelo MLP como la regresión polinómica han aumentado su error entre 10 y 30 veces.

En la figura 7.6 observamos el histograma del error de los tres modelos donde se aprecia

que el modelo MLP presenta un rango significativamente menor del mismo.

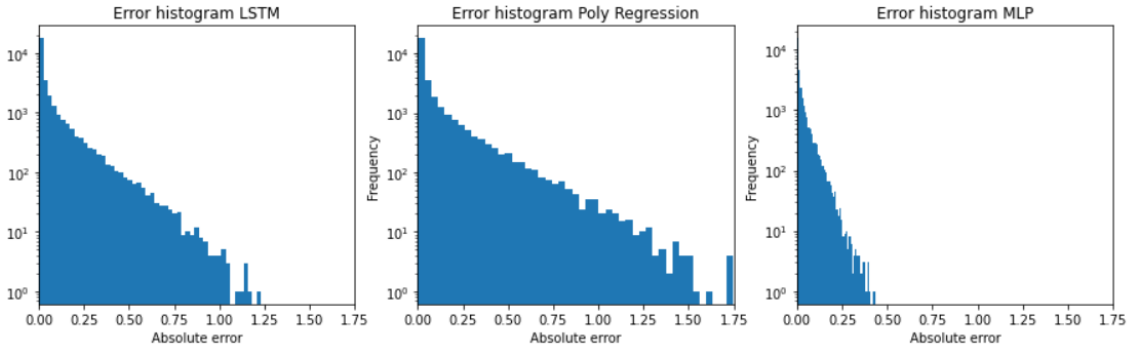


Figura 7.6. *Histogramas del error para los modelos. Fuente: Propia*

En la figura **??** se muestran las gráficas respecto a la distancia del cambio de fase.

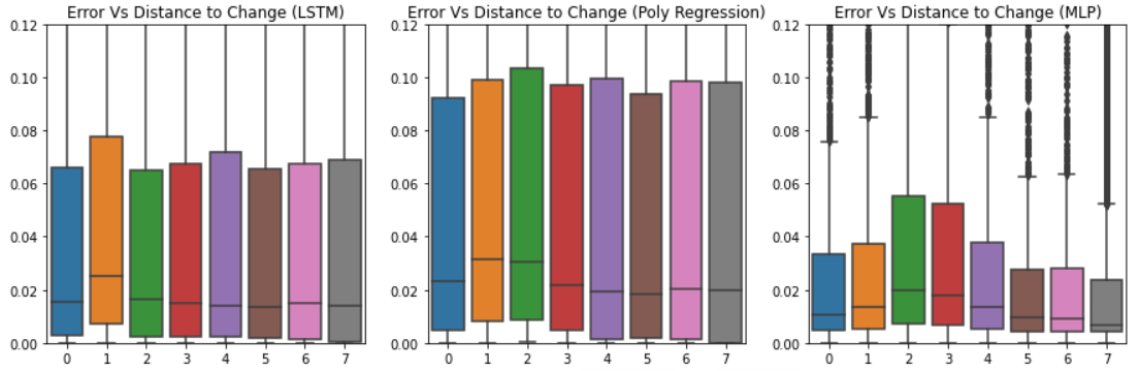


Figura 7.7. *Error frente a la distancia con ruido añadido. Fuente: Propia*

Vemos que los modelos LSTM y la regresión lineal tienen un error constante e independiente de la distancia mientras que el MLP sigue conservando su capacidad de reducir el error con las distancia.

En la figura 7.8 podemos observar el error por fase con ruido añadido. Para los tres modelos el error es mayor para las fases en que se alcanza mayor velocidad.

En la gráfica 7.9 se muestra la gráfica del error respecto a la velocidad al trabajar con ruido añadido. Se puede apreciar una correlación entre error y velocidad que no era tan evidente cuando se trabajaba sin ruido.

Para acabar tenemos en al figura 7.10 el error respecto a la aceleración con ruido añadido. En estas gráficas no se aprecia una relación clara entre ambas magnitudes.

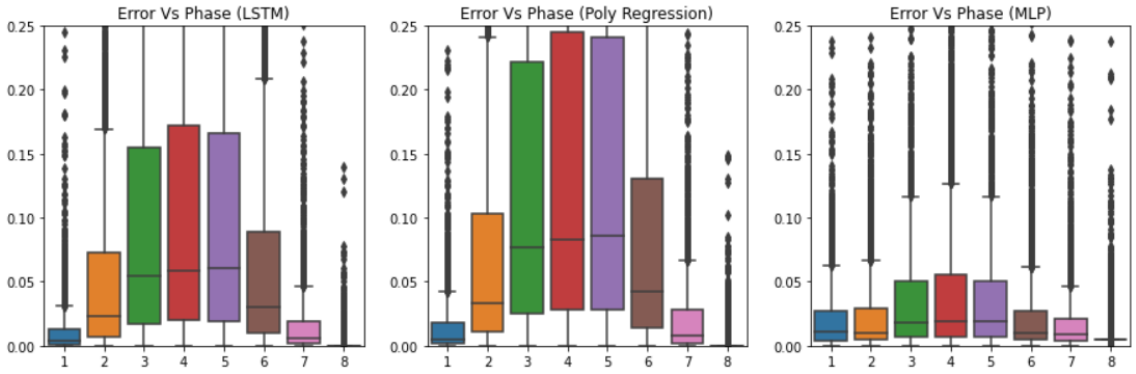


Figura 7.8. *Errores por fase con ruido añadido. Fuente: Propia*

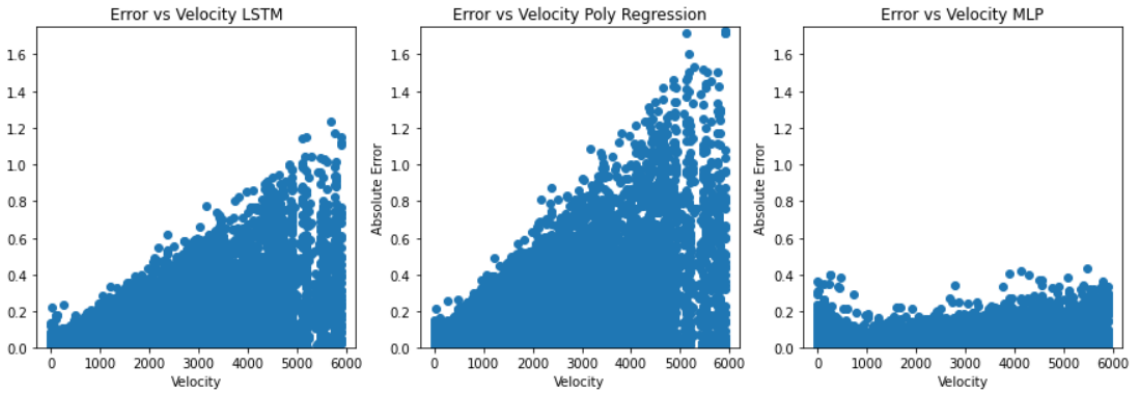


Figura 7.9. *Error respecto la velocidad con ruido añadido. Fuente: Propia*

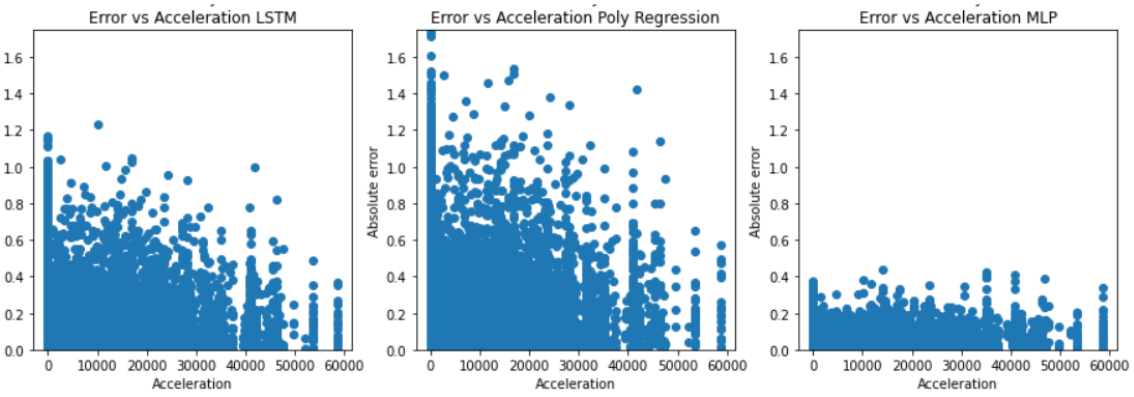


Figura 7.10. *Error respecto la aceleración con ruido añadido. Fuente: Propia*

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se va a contrastar los resultados obtenidos con los objetivos marcados inicialmente y se van a proponer nuevas vías de mejora del trabajo actual.

8.1. Repaso de cumplimiento de objetivos

El objetivo general era evaluar el rendimiento de un modelo LSTM para la predicción de trayectorias de ejes industriales que siguen un perfil de movimiento de siete fases.

A continuación se va a repasar cual ha sido el desempeño del modelo en cada uno de los objetivos específicos y se va a comparar con los modelos alternativo MLP y regresión polinómica. Esta comparación se va a llevar a cabo tanto sin ruido añadido como con ruido añadido a los datos.

El primer objetivo específico consistía en medir el error absoluto medio y el error absoluto máximo con un conjunto de trayectorias de test. En la sección 6.1.1 vimos que los errores del modelo LSTM es menor al del modelo MLP. Sin embargo la regresión lineal iguala al modelo LSTM en el error medio y obtiene un valor máximo incluso menor. Dada la sencillez de método de regresión polinómica ésta parece ser la mejor alternativa.

El segundo objetivo era estudiar el error según la distancia al cambio de fase. Como es de esperar todos los modelos incrementan el error a una distancia de 1 time-step. En cambio el modelo LSTM reduce rápidamente de nuevo el error desde la distancia 2 en adelante. El modelo de regresión lineal mantiene un error elevado en las distancias 1 y 2. El modelo MLP arrastra un error grande en las distancias 2,3 y 4. En este aspecto el modelo LSTM demuestra una mayor rapidez en la adaptación al cambio de parámetros del movimiento.

El siguiente objetivo era observar la distribución del error para cada una de la fases del movimiento. En este sentido el modelo LSTM y la regresión presentan un error algo mayor en las fases 1,3,5 y 7 donde la aceleración se incrementa linealmente y la posición sigue una función cúbica. Es lógico que con una trayectoria más compleja se cometa más error de predicción. En cambio el modelo MLP presenta una distribución de error por fase más aleatoria.

El cuarto objetivo consistía en estudiar el error en función de la velocidad y la aceleración. Al observar la nube de puntos de errores respecto a la velocidad no se observan patrones o correlaciones significativas con ninguno de los modelos. En cambio todos los modelos presentan algunos errores anómalos a velocidad cero o próxima a cero. En el estudio del error con respecto a la aceleración si que se observan correlaciones lineales en los tres modelos.

Al añadir ruido aleatorio a los datos el comportamiento de los modelos cambia drásticamente. El modelo MLP pasa a ser claramente superior a los otros. Mientras el modelo MLP multiplica su error por un factor entre 1.5 y 2 los otros dos modelos multiplican su error por un factor de entre 10 y 30.

8.2. Conclusiones

Analizando los datos objetivamente se concluye que el modelo LSTM no ha demostrado una clara superioridad frente a los otros dos modelos de la comparativa para el propósito de predicción de trayectorias tal como se ha planteado en el presente trabajo.

En algunas aplicaciones podemos considerar que no existe ruido en los datos. Éste seria el caso en que las consignas teóricas de trayectorias se calculan y se comunican al controlador esclavo. En este caso hemos podido comprobar que el modelo LSTM es el más rápido en reducir el error ante cambios en la trayectoria. Sin embargo la diferencia con el modelo de regresión polinómica no es demasiada y éste último es mucho más sencillo de implementar y más ligero computacionalmente.

En otras aplicaciones podemos considerar que si que hay ruido en los datos. Este seria el caso en que el controlador esclavo lee la posición del eje maestro mediante un encoder. En este caso el modelo MLP ha demostrado ser mucho más robusto al ruido que el modelo LSTM y también que la regresión polinómica.

8.3. El principal problema: los errores anómalos a velocidad cero

Los tres modelos testados presentan algunas apariciones de errores grandes que se producen a velocidad próximas a cero lo cual es un inconveniente.

Esto significa que en algunas ocasiones cuando se espera un incremento de posición próximo o igual a cero el modelo está prediciendo un incremento de posición muy elevado. Además esto dificulta el evaluar el error porcentualmente porque el denominador en estos casos es cero.

Estos errores anómalos se producen tan sólo en un instante de tiempo y se corrigen rápidamente en el siguiente pero van a provocar pequeños saltos en el eje a controlar. Por tanto se trata de un defecto del modelo para el que se van a proponer algunos vías de mejora en la siguiente sección

8.4. Propuestas de mejora en trabajos futuros

El principal defecto de los modelos testados son los errores anómalos que se producen en el cambio de fase, y son especialmente preocupantes cuando se producen a velocidad cero o próxima a cero porque es cuando tienen mayor impacto. A continuación se proponen una serie de estrategias a explorar para intentar reducir dichos errores.

8.4.1. Filtrado de la predicción

Una posible solución a ensayar seria filtrar el valor de predicción. Podría plantearse hacer una media del último valor leído con la predicción real y entregar dicha media como predicción final. Esto previsiblemente incrementaría el error medio de las predicciones pero seguramente reduciría el error máximo que es más problemático. También se podría contemplar un filtrado más acusado a velocidades próximas a cero que a velocidad mayores. Recordemos que los errores grandes a velocidades próximas a cero son los que tienen un mayor impacto.

8.4.2. Utilización de ensembles

Una posible manera de paliar los errores anómalos podría ser trabajar con un emsemble de varios modelos y aplicar como predicción la media de las predicciones de cada uno

de los modelos. Esto es lo que se conoce como bagging (Breiman, 1996). Este ensemble podría estar compuestos de modelos LSTM similares al desarrollado en este trabajo pero con pequeñas variantes en sus parámetros. También podrían ser modelos entrenados con diferentes sets de datos. De esta manera se espera conseguir una reducción del error medio y también del error máximo ya que esta técnica reduce la varianza global.

8.4.3. Entrenamiento con más cambios de fase

Otra posible estrategia para reducir el error en el cambio de fase seria construir un dataset de trayectorias de entrenamiento con más cantidad de cambios de fase. Los eventos de cambio de fase se dan con una frecuencia bastante menor que el mantenimiento de una fase por tanto estos casos están menos representados en el dataset de entrenamiento. Por otro lado es posible que al aumentar el número de cambios de fase el modelo se comporte mejor en estos pero se comporte peor de forma global.

Bibliografía

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971.

Altché, F. and de La Fortelle, A. (2017). An lstm network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359. IEEE.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Choi, P. P. and Hebert, M. (2006). Learning and predicting moving object trajectory: a piecewise trajectory segment approach. *Robotics Institute*, 337:1–17.

Community, T. N. (2021). Api reference, numpy.polyfit.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Karakaya Murat (2020). Lstm: Understanding the number of parameters. `https://medium.com/deep-learning-with-keras/lstm-understanding-the-number-of-parameters-c4e087575756#:~:text=Summary%20indicates%20the%20total%20number,48%20as%20we%20computed%20above!`, Last accessed on 2021-05-28.

Keras API Reference (2021). Lstm layer. `https://keras.io/api/layers/recurrent_layers/lstm/`, Last accessed on 2021-05-28.

Lewin, C. (2007). Mathematics of motion control profiles. *URL: https://www. pmdcorp. com/resources/type/articles/get/mathematics-of-motion-controlprofiles-article (7.11. 2019.)*.

Ostertagová, E. (2012). Modelling using polynomial regression. *Procedia Engineering*, 48:500–506.

Park, S. H., Kim, B., Kang, C. M., Chung, C. C., and Choi, J. W. (2018). Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE.

Payeur, P., Le-Huy, H., and Gosselin, C. M. (1995). Trajectory prediction for moving objects using artificial neural networks. *IEEE Transactions on Industrial Electronics*, 42(2):147–158.

ResarchGate (2021). Lagging-domain model for compensation of hysteresis of xmr sensors in positioning applications. `https://www.researchgate.net/figure/The-setup-of-a-linear-encoder-A-read-head-with-xMR-sensor-moves-along-the-x-axis-above_fig3_326418583`, Last accessed on 2021-05-28.

Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*.

Violos, J., Tsanakas, S., Androutsopoulou, M., Palaiokrassas, G., and Varvarigou, T. (2020). Next position prediction using lstm neural networks. In *11th Hellenic Conference on Artificial Intelligence*, pages 232–240.

Wang, C., Ma, L., Li, R., Durrani, T. S., and Zhang, H. (2019). Exploring trajectory prediction through machine learning methods. *IEEE Access*, 7:101441–101452.

Wiest, J., Höffken, M., Kreßel, U., and Dietmayer, K. (2012). Probabilistic trajectory

prediction with gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*, pages 141–146. IEEE.

Ye, N., Zhang, Y., Wang, R., and Malekian, R. (2016). Vehicle trajectory prediction based on hidden markov model.

Apéndice A

Apendices

Predicción de trayectorias en aplicaciones de control de movimiento industriales

Javier Menchén Agut

Universidad Internacional de la Rioja, Logroño (España)

15 de Julio de 2021

RESUMEN

En ciertas aplicaciones de control de movimiento industriales donde un eje esclavo replica la trayectoria de un eje maestro, el hecho de anticipar la trayectoria del eje maestro puede suponer una mejora en el error de seguimiento del esclavo. En este trabajo se ha ensayado un modelo tipo Long Short-Term Memory (LSTM) para tal fin por su reconocida eficacia en predicciones de series temporales. El modelo LSTM se ha comparado con un modelo base tipo Multilayer Perceptron (MLP) y con un modelo de regresión polinómica. Se ha determinado el error medio y máximo y la relación del error con diferentes factores. Estos modelos han sido probados trabajando sin ruido y con ruido añadido a los datos de entrada. Se ha constatado que al trabajar sin ruido el modelo LSTM tiene un rendimiento parecido a la regresión polinómica pero se adapta más rápido a cambios en la trayectoria. Con ruido añadido a los datos el modelo MLP presenta una mayor robustez en la predicciones que los otros dos.

PALABRAS CLAVE

Control de Movimiento, Long Short-Term Memory, Predicción de Trayectorias

I. INTRODUCCIÓN

El movimiento en la maquinaria y los procesos industriales se lleva a cabo mediante actuadores que pueden ser lineales o rotacionales. En algunas aplicaciones es habitual que un eje esclavo deba acoplarse a un eje maestro del cual no se tiene el control. En la figura 1 se puede observar el acoplamiento entre un robot y una cinta transportadora. Para poder manipular los objetos de la cinta mientras está en movimiento la pinza del robot debe moverse en sincronismo con ella.

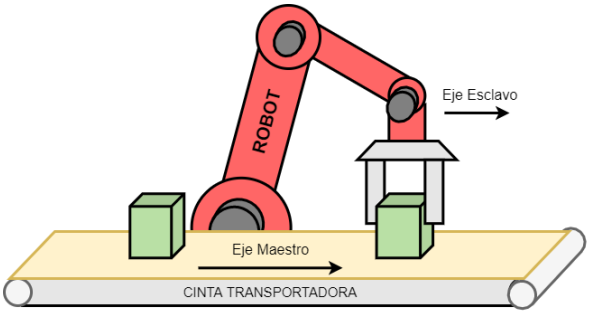


Figura 1: Acoplamiento entre robot y cinta transportadora. Fuente: Propia

La comunicación de la posición del maestro al esclavo implica un retardo en el control que se traduce en un error de seguimiento. En este contexto puede ser interesante poder predecir los siguientes puntos de la trayectoria del eje maestro. Esta anticipación permitiría reducir notablemente el error de seguimiento.

En aplicaciones de predicción series temporales las redes Long Short Term Memory (LSTM) han demostrado dar muy buenos resultados. Por tanto ese ha sido el modelo seleccionado en este trabajo. Para medir el rendi-

miento del modelo LSTM se compara con una red neuronal MLP y con una regresión polinómica.

En estado del arte se hace una introducción a las aplicaciones industriales de control de movimiento, se revisan diferentes técnicas para la predicción de trayectorias y se justifica la elección del modelo LSTM para esta aplicación. En la siguiente sección se detallan los objetivos de este trabajo y la metodología empleada para su consecución.

En la sección de contribución se detalla cómo se han generado los datos de entrenamiento, la arquitectura de los modelos y su entrenamiento. A continuación se presentan los resultados comparativos del modelo LSTM con respecto a los otros dos. En la siguiente sección se evaluan dichos resultados.

Por último en la sección conclusiones se reflexiona sobre la conveniencia del uso del modelo LSTM y se proponen nuevos trabajos de mejora del actual.

II. ESTADO DEL ARTE

A. Las aplicaciones de control de movimiento

Las trayectorias que genera un controlar de ejes industriales para describir un movimiento de un punto a otro consiste en una serie temporal de consignas de posición. El controlador de movimiento entregará en cada intervalo de tiempo una nueva consigna al regulador del eje.

Las trayectorias de los ejes se definen en base a unos parámetros que son la velocidad máxima, la aceleración máxima y el jerk que es la derivada o rampa de la aceleración. Estos serán los parámetros de una función polinómica de tercer grado [1].

$$x(t) = \frac{1}{6}at^3 + \frac{1}{2}bt^2 + ct + x_0 \qquad (1)$$

En la generación de trayectorias es típico la utilización de perfiles de movimiento de 7 fases [2]. En cada una de estas fases los parámetros de la curva se mantienen constantes. En cambio al cambiar de fase cambian también los parámetros como se puede apreciar en la figura 2

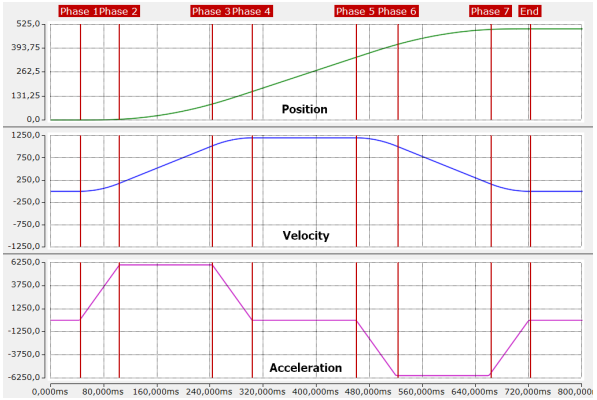


Figura 2: Perfil de movimiento de 7 fases. Fuente: Propia

Con esta estrategia no se producen escalones abruptos de aceleración. En cada fase el movimiento cambia entre velocidad constante, movimiento uniformemente acelerado y movimiento acelerado linealmente.

En las aplicaciones control de movimiento en máquinas industriales en ocasiones es necesario que un controlador esclavo deba sincronizar uno o más ejes con un eje maestro gobernado por otro controlador. En el caso más sencillo de acoplamiento el eje eslavo replica la posición del eje maestro. En algunas ocasiones el eje maestro y el eje esclavo se encuentran en diferentes controladores por lo que el controlador del eje esclavo no conoce la posición del eje maestro por lo que es necesario comunicar la consigna de posición desde el maestro al esclavo. Esta comunicación introduce un retardo en la llegada de las posiciones que se traduce en un error de seguimiento.

El hecho de predecir la siguiente posición de la trayectoria del eje maestro permitirá al controlador esclavo anticiparse y tener una reducción significativa del error de seguimiento.

B. Predicción de trayectorias basada en modelos probabilísticos

Una de las maneras de enfocar la predicción trayectorias es mediante modelos probabilísticos como modelos de Markov o modelos mixtos gausianos. Estos métodos sin embargo no hacen una verdadera regresión sino que, mediante una estrategia u otra, hacen una clasificación

sobre un conjunto finito de posible elecciones.

En el trabajo de [3] se aborda la predicción de trayectorias para objetos mediante cadenas de Markov. En otro trabajo [4] se utilizan modelos ocultos de Markov junto con el algoritmo de Viterbi para predecir la trayectoria más probable de vehículos representada como la posición en un grafo que modeliza el mapa de carreteras.

C. Modelo MLP de Payeur para la prediccion de trayectorias

La aplicación más similar a la que se trata en este trabajo es la de [1] en que se hace la predicción de trayectorias de objetos móviles mediante redes neuronales de tipo Multi Layer Perceptron (MLP). El objeto móvil se desliza por una rampa y tiene movimientos de traslación en los ejes x,y y también puede rotar sobre el eje z. Para cada uno de los ejes x,y y θ se hace una predicción del siguiente valor de posición para que un robot pueda planificar una trayectoria que le permita interceptar el objeto. El modelo consta de dos capas ocultas de 20 unidades cada una y una función de activación sigmoide.

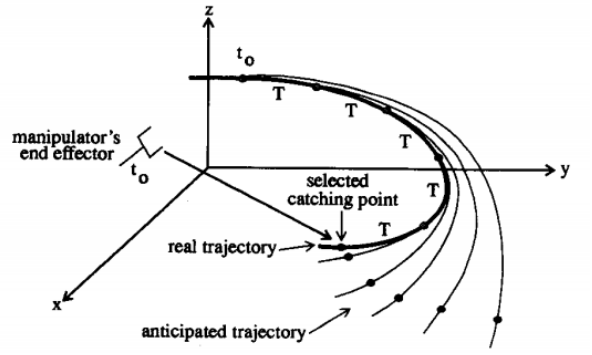


Figura 3: Predicción de trayectorias en el plano para la captura del robot. Fuente: [1]

Los autores comparan este modelo con el de una regresión polinómica. Se comprueba que el error medio cometido por ambos modelos es prácticamente igual pero se destaca que las redes neuronales son más robusta al ruido en los datos de entrada que el modelo de regresión polinómica.

D. Predicción de trayectorias mediante modelos LSTM

Los modelos LSTM se han empleado para la predicción de trayectorias en diferentes ámbitos. Un campo muy activo de investigación es el de los vehículos autónomos donde se enmarca este trabajo de [5] en que se utilizan modelos LSTM para la predicción de trayectorias de vehículos en autopistas. En un trabajo similar [6] se empleó un modelo Sequence-to-Sequence basado en LSTM para resolver este mismo problema. También se han empleado para predecir trayectorias de peatones [7]

Cambiando de campo, en este trabajo [8] se describe una aplicación para la predicción de rumbo de barcos. Los autores utilizan un algoritmo genético para escoger el mejor modelo LSTM de un repositorio de modelos preentrenados con diferentes arquitecturas.

En este otro trabajo [9] se presenta una aplicación de predicción de trayectorias de personas utilizando la información de geolocalización de sus teléfonos móviles.

E. Predicción de trayectorias mediante regresión polinómica

En el caso de que la trayectoria a predecir se pueda modelizar mediante una función polinómica de parámetros estáticos se puede aplicar una regresión polinómica sobre los últimos puntos de la trayectoria. El resultado de la regresión polinómica son los coeficientes de la función polinómica que mejor se ajustan al error cuadrático medio de los puntos analizados [10]. El inconveniente de este método es que está limitado a una función polinómica de un grado determinado y de parámetros estáticos.

F. Conclusiones del estado del arte

Como hemos visto cuando se trata de hacer predicciones de valores en el espacio continuo las redes LSTM son la opción más utilizada y que mejores resultados está dando. El éxito de esta redes en otros ámbitos hace pensar que también puedan dar buen resultado en la predicción de trayectorias de ejes que se rigen por funciones polinómicas y la incertidumbre se li-

mita a los momentos de cambio de fase donde cambian los parámetros la función.

III. LONG SHORT-TERM MEMORY

Las redes Long Short Term Memory [11] son una mejora de las redes recurrentes que se crearon con el fin de solucionar ciertos problemas habituales propios de las mismas conocidos como desvanecimiento de gradientes o la explosión de gradientes. Además del hidden state h se añade otro estado llamado cell state c. Se calculan una serie de vectores conocidos como gates. La input gate i controla como se actualiza el estado c junto con el gate g sin nombre particular. La output gate define qué parte de c se actualiza en h. Finalmente la forget gate f [12] controla que parte de c se debe olvidar.

Las gates se calculan mediante las siguientes expresiones:

$$i(t) = \sigma(W_{ih}h_{t-1} + W_{ix}x_t) \tag{2}$$

$$f(t) = \sigma(W_{fh}h_{t-1} + W_{fx}x_t) \tag{3}$$

$$o(t) = \sigma(W_{oh}h_{t-1} + W_{ox}x_t) \tag{4}$$

$$g(t) = \tanh(W_{ch}h_{t-1} + W_{cx}x_t) \tag{5}$$

Las expresiones para el cálculo de c(t) y h(t) son:

$$c(t) = f_t \circ c_{t-1} + i_t \circ g_t \tag{6}$$

$$h(t) = o_t \circ \tanh(c_t) \tag{7}$$

IV. OBJETIVOS Y METODOLOGÍA

El objetivo general de este trabajo consiste en evaluar el rendimiento de un modelo LSTM en la predicción de trayectorias de ejes industriales y comparar dicho rendimiento con un modelo base tipo MLP y con el método de regresión polinómica.

Los modelos de entrenarán específicamente para predecir trayectorias generadas con un perfil de 7 fases. Dichas trayectorias de entrenamiento se generarán por programa creando un repertorio variado con diferentes valores de parámetros.

Para medir el éxito de este modelo se van a realizar una serie de mediciones. El objetivo para cada una de estas mediciones es alcanzar con el modelo LSTM un resultado mejor que los otros dos modelos de la comparativa:

- Se medirá el error medio y el error máximo del modelo a la hora de hacer predicciones con un conjunto de datos de test.
- Se estudiará la evolución del error según la distancia a un cambio de fase. En cada cambio de fase se produce un cambio en los parámetros de la trayectoria que es previsible que aumente el error de predicción.
- Se comprobará como se comporta el error en todo el rango de velocidades y de aceleraciones aplicadas.
- Se compararán todos los puntos anteriores con el modelo MLP de [1] y con una regresión polinómica. Esta comparación se llevara a cabo tanto sin ruido como con ruido en los datos de entrada.

V. CONTRIBUCIÓN

En este capítulo se va a detallar cómo se han generado los datos de las trayectorias que después se van utilizan tanto para el entrenamiento como para la verificación de rendimiento. Se va a describir el diseño de los modelos aplicados y cómo se ha llevado a cabo el proceso de entrenamiento.

A. Generación de los datos de entrenamiento

Las trayectorias necesarias para el entrenamiento se han generado utilizando las funciones polinómicas. Se ha implementado una función en Python que genera un perfil de movimiento trapezoidal de 7 fases en base a los parámetros

de jerk, aceleración máxima y velocidad máxima. En la figura 4 se muestran dos ejemplos de perfiles de movimiento con diferentes valores de parámetros.

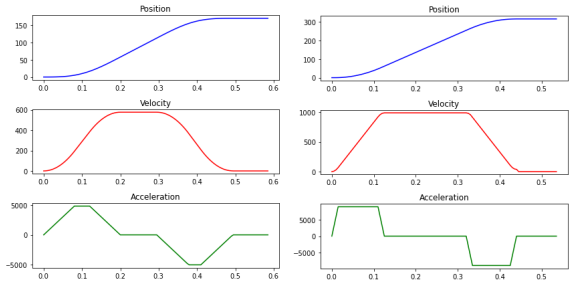


Figura 4: Ejemplo de dos perfiles de movimiento con diferentes parámetros. Fuente: Propia

La fórmula empleada es 8, donde j es el jerk y a_o, v_0 y x_0 son la aceleración, la velocidad y la posición iniciales.

$$x(t) = \frac{1}{6}jt^3 + \frac{1}{2}a_0t^2 + v_0t + x_0 \qquad (8)$$

Con estos parámetros se genera una concatenación de trayectorias introduciendo una variabilidad aleatoria como técnica de regularización. En la figura 5 se puede observar un ejemplo de concatenación de varios movimientos.

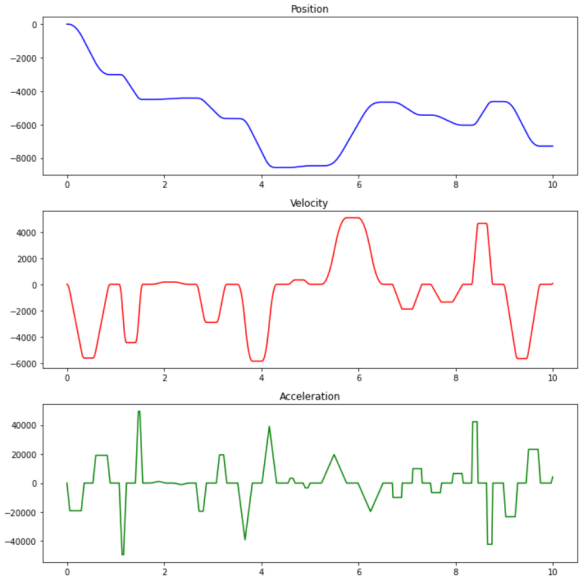


Figura 5: Ejemplo de varios movimientos concatenados. Fuente: Propia

Inspirado por trabajos anteriores como el de [1] y el de [8] se ha alimentado al modelo con incrementos de posición. Una vez obtenidos los incrementos se han normalizado a valores entre 0 y 1. Finalmente para llevar a cabo el entrenamiento se ha dividido toda la secuencia de posiciones en ventanas de longitud 8 y se ha tomado la siguiente posición como valor objetivo a predecir.

B. Diseño de los modelos de la comparativa

Haciendo uso de los frameworks Keras y Tensorflow se ha diseñado un modelo cuyo elemento principal es una primera capa de tipo LSTM de 64 unidades. Se ha empleado la función de activación lineal. La capa de salida es de tipo Dense con tan sólo una neurona y una función de activación lineal que será la encarga de entregar el valor de predicción del modelo

Por otro lado se ha implementado también con Keras un modelo MLP que replica la misma arquitectura de [1]. Consta de dos capas de 20 neuronas cada una y una capa final de salida. Se ha utilizado la función de activación sigmoid por ser utilizada también en el trabajo original.

Para la implementación de la regresión polinómica se ha empleado la función polyfit() de la libreria numpy [13]. Se ha utilizado un polinomio de grado 3. Para cada predicción es necesario llamar a la función polyfit() para encontrar un nuevo polinomio que se ajuste a la última secuencia de puntos, después se calcula el valor que devuelve dicho polinomio para el siguiente punto.

C. Entrenamiento del modelo

Se han generado una secuencia de en torno a 150.000 puntos que se han dividido en unos 120.000 para entrenamiento y 30.000 para la validación. Se ha optado por una función de coste Mean Square Error (MSE) ya que penaliza errores grandes. Como métrica a observar se ha seleccionado el error absoluto medio (MAE)

Se ha seleccionado el algoritmo de optimización adam y un batch size de 16. Se ha empleado una función de callback para guardar

los parámetros del modelo cada vez que mejora el validation loss. En la figura 6 se puede apreciar la evolución del coste (loss) y del coste de validación durante el entrenamiento.

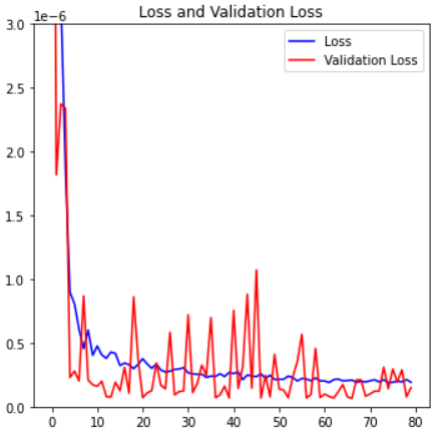


Figura 6: Curva de evaluación del loss y el validation loss. Fuente: Propia

VI. RESULTADOS

A. Resultados sin ruido

En primer lugar se ensayan los tres modelos señalados haciendo predicciones sobre el mismo conjunto de datos de validación. En este experimento no se añade ruido a los datos de entrada

El modelo MLP presenta un error medio de 0,013 mm frente a los 0,0022 mm de los modelos LSTM y la regresión. El error máximo del MLP es de 0,39 mm frente a 0,18 mm del LSTM y 0,14 mm de la regresión. En la tabla 1 se muestra una comparativa de resultados.

Por tanto atendiendo a estos datos parece que el método que da un mejor rendimiento de momento es el de regresión polinómica.

En la figura 7 podemos comparar el histograma del error de los modelos LSTM, de la regresión polinómica y del modelo MLP. La amplitud de la distribución para el modelo MLP es sensiblemente mayor que para los otros dos. Se aprecia que el error máximo de la regresión es algo menor que el del modelo LSTM pero por otro lado hay más frecuencias de estos errores máximos.

	LSTM	Regres.	MLP
Error medio absoluto (mm)	0.0022	0.0022	0.013
Error máximo absoluto (mm)	0,18	0,14	0.39
Error medio porcentual (%)	0.007 %	0.007 %	0.043 %
Longitud de la secuencia de entrada	8	4	8
Número de parámetros	16896	-	621

Cuadro 1: Diferencias entre los modelos. Fuente: Propia.

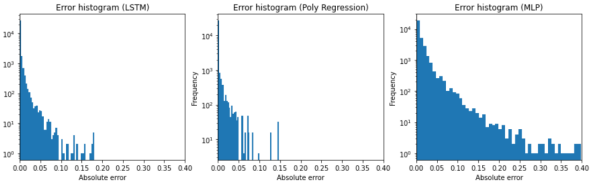


Figura 7: Histogramas del error para los modelos. Fuente: Propia

Al observar el error respecto a la distancia al cambio de fase en la figura 8 apreciamos notables diferencias entre los modelos. El modelo LSTM presenta un error grande sólo a una distancia de 1 y a partir de ahí corrige rápidamente el error. El modelo de regresión mantiene un error elevado entre las distancias 1 y 2. A partir de la distancia 3 presenta un error mucho menor que el resto. El modelo MLP tiene un error mucho mayor a todas las distancias estando el pico en la distancia 2.

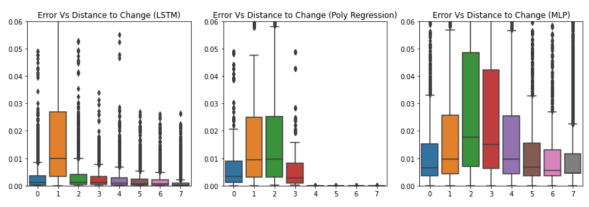


Figura 8: Error frente a la distancia del cambio de fase. Fuente: Propia

En cuanto a adaptación al cambio de fase el modelo LSTM es más eficaz que el resto y el MLP es bastante peor.

En la gráfica 9 podemos comparar el comportamiento del error respecto a la velocidad en los tres modelos. Podemos apreciar cómo la distribución es similar pero con mayor rango de error para el modelos MLP que para los otros dos. Los errores anómalos en velocidades próximas a cero están presentes en los tres modelos

Figura 9: Error respecto la velocidad de los modelos LSTM, regresión y MLP. Fuente: Propia

B. Resultados con ruido

En este experimento se añade un ruido aleatorio de 0,5 % a los valores de posición del conjunto de datos de validación y se repiten las predicciones con los mismos. En la tabla 2 vemos los valores de error medio y máximo.

	LSTM	Regres.	MLP
Error medio absoluto (mm)	0.065	0.093	0.024
Error máximo absoluto (mm)	1,23	1,85	0.43
Error medio porcentual (%)	0.21 %	0.31 %	0.08 %

Cuadro 2: Errores de los modelos con 0,5 % de ruido. Fuente: Propia.

En esta ocasión el modelo que presenta un error menor con diferencia es el MLP. Esto también es apreciable en el histograma de la figura 10.

En la figura 11 se muestran las gráficas respecto a la distancia del cambio de fase. Los modelos LSTM y la regresión tienen un error

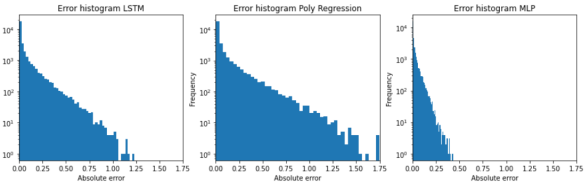


Figura 10: Histogramas del error para los modelos. Fuente: Propia

constante mientras que el MLP conserva su capacidad de reducir el error con la distancia.

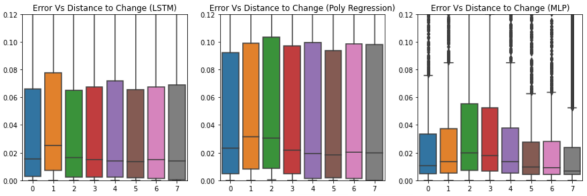


Figura 11: Error frente a la distancia con ruido añadido. Fuente: Propia

En la gráfica 12 se muestra el error respecto a la velocidad. Se puede apreciar una correlación entre error y velocidad que existía cuando se trabajaba sin ruido. Esto es debido a que el ruido introducido es proporcional al incremento de posición.

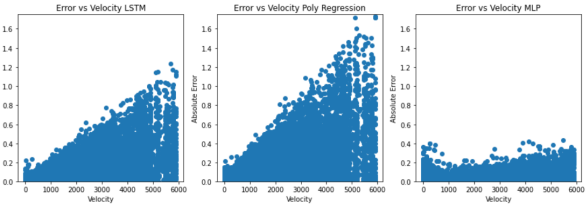


Figura 12: Error respecto la velocidad con ruido añadido. Fuente: Propia

VII. ANÁLISIS DE RESULTADOS

El objetivo general era evaluar el rendimiento de un modelo LSTM para la predicción de trayectorias de ejes industriales que siguen un perfil de movimiento de siete fases.

El primer objetivo específico consistía en medir el error absoluto medio y el error absoluto

máximo. Hemos visto que la regresión lineal iguala al modelo LSTM en el error medio y obtiene un valor máximo incluso menor. Dada la sencillez de este método la regresión parece ser la mejor alternativa.

El segundo objetivo era estudiar el error según la distancia al cambio de fase. El modelo LSTM reduce rápidamente el error desde la distancia 2 en adelante. El modelo de regresión lineal mantiene un error elevado en las distancias 1 y 2. El modelo MLP arrastra un error grande en las distancias 2,3 y 4. Por tanto modelo LSTM demuestra una mayor rapidez en la adaptación al cambio de parámetros del movimiento

El tercer objetivo consistía en estudiar el error en función de la velocidad y la aceleración. No se aprecian correlaciones significativas con ningun modelo. Sin embargo todos ellos presentan algunos errores anómalos a velocidad próxima a cero.

Al añadir ruido aleatorio a los datos el comportamiento de los modelos cambia drásticamente. El modelo MLP pasa a ser claramente superior a los otros.

VIII. CONCLUSIONES

Analizando los datos objetivamente se concluye que el modelo LSTM no ha demostrado una clara superioridad frente a los otros dos modelos para la predicción de trayectorias tal como se ha planteado en el presente trabajo.

En aplicaciones sin ruido significativo el modelo LSTM es el más rápido en adaptarse a cambios en la trayectoria. Sin embargo la diferencia con el modelo de regresión polinómica no es demasiada y éste último es mucho más sencillo de implementar por lo que parece preferible

En otras aplicaciones si puede haber ruido en los datos como cuando el controlador esclavo lee la posición del maestro mediante un encoder. En este caso el modelo MLP ha demostrado ser mucho más robusto al resto.

El principal defecto de los modelos ensayados son los errores que se producen a velocidades próximas a cero. A continuación se proponen una serie de estrategias a explorar para intentar reducir dichos errores en trabajos futuros.

Una posible solución seria filtrar el valor de predicción haciendo una media del último valor leído con la predicción real. Esto incrementaría el error medio de las predicciones pero seguramente reduciría el error máximo que es más problemático. Otra estrategia podría ser trabajar con un emsemble de varios modelos. De esta manera se espera conseguir una reducción del error medio y máximo. Otra posible solución para reducir el error en el cambio de fase seria construir un dataset de trayectorias de entrenamiento con más cantidad de cambios de fase.

Referencias

[1] Pierre Payeur, Hoang Le-Huy, and Clement M Gosselin. Trajectory prediction for moving objects using artificial neural networks. *IEEE Transactions on Industrial Electronics*, 42(2):147–158, 1995.

[2] Chuck Lewin. Mathematics of motion control profiles. *URL: https://www. pmdcorp. com/resources/type/articles/get/mathematics-of-motion-controlprofiles-article (7.11. 2019.)*, 2007.

[3] Patrick Pakyan Choi and Martial Hebert. Learning and predicting moving object trajectory: a piecewise trajectory segment approach. *Robotics Institute*, 337:1–17, 2006.

[4] Ning Ye, Yingya Zhang, Ruchuan Wang, and Reza Malekian. Vehicle trajectory prediction based on hidden markov model. 2016.

[5] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359. IEEE, 2017.

[6] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[7] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[8] John Violos, Stylianos Tsanakas, Maro Androutsopoulou, Georgios Palaiokrassas, and Theodora Varvarigou. Next position prediction using lstm neural networks. In *11th Hellenic Conference on Artificial Intelligence*, pages 232–240, 2020.

[9] Chujie Wang, Lin Ma, Rongpeng Li, Tariq S Durrani, and Honggang Zhang. Exploring trajectory prediction through machine learning methods. *IEEE Access*, 7:101441–101452, 2019.

[10] Eva Ostertagová. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 2012.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[13] The Numpy Community. Api reference, numpy.polyfit, 2021.