CIMAT

Centro de Investigación en Matemáticas, A.C.

CONTRIBUCIONES A LA PREDICCIÓN DE TRAYECTORIAS HACIENDO USO DE APARIENCIA Y FLUJO ÓPTICO

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

Presenta

Andrea Berenice Ek Hobak

Directores de Tesis:

Dr. Jean Bernard Hayet

Dr. Ángel Sánchez Calle

Autorización de la versión final

Guanajuato, Gto., a 8 de enero de 2021

Agradecimientos

Este trabajo fue posible gracias a la ayuda de muchas personas que ya formaban parte de mi vida y que también fui conociendo en el camino de esta aventura.

En el proceso de la maestría sucedieron situaciones inesperadas, que me hicieron sentir que nunca la iba poder culminar. Por ello agradezco a mi tutor académico del CIMAT, Dr. Johan J. Van Horebeek, su apoyo e interés en mi formación académica y por los consejos personales que me dio en su momento.

A mi asesor de tesis, el Dr. Jean Bernard Hayet, le agradezco infinitamente por su incansable paciencia, ideas y guía en todo el proceso de esta tesis, por las correcciones que me hacía en la escritura y en las cosas que no comprendía correctamente. Al igual le agradezco toda la ayuda que él y su familia (Dra. Cladia Esteves y Paul) me brindaron, nunca lo olvidaré. A mi coasesor el Dr. Ángel, por las ideas, lectura y correcciones de esta tesis.
Al igual agradezco al Dr. Mariano Rivera Meraz (CIMAT) por haber sido mi sinodal, por tomarse el tiempo de leer la tesis, así como de los comentarios y correcciones que me hizo. Al Dr. David Gómez Gutiérrez (INTEL), por ser mi sinodal, así como también por los comentarios y sugerencias que me hizo en el proceso de la tesis.

A mis compañeros de oficina, Dani, Martel, Edith, Favia, Luis, por los momentos que pasamos juntos. A Jamie, Bernardo, y Oscar, por siempre apoyarme en el transcurso de la maestría, por aguantarme cuando me sentía triste, al igual por los momentos alegres que pasamos, sin ustedes nada hubiera sido igual. A Alán y José por su amistad, sus enseñanzas y su tiempo. A los profesores de inglés, Sigfrido, Janet Mary Izzo por siempre enseñar con amor y paciencia. A mis compañeros de la clase de inglés por compartir sus conocimientos conmigo.

A mi madre Julia Hobak, por siempre apoyar y defender mis sueños, jamás podre terminar de agradecer cada una de las cosas que has hecho por mi, por trabajar para que yo estudiará, por dejar todo y venir a un lugar desconocido para tí, para cuidar a bebezón. A mi papá Jorge Ek y Balam por ser un padre que enseña y aconseja con amor, por siempre trabajar para que nunca nos falte nada. A mi esposo Xavier por apoyarme y animarme cuando sentía que ya no podía, a bebezón por sus sonrisas y travesuras que me dan las fuerzas para seguir.

Al centro de Investigación en matemáticas, A.C. (CIMAT), por darme las herramientas para desarrollarme, por enseñarme tantas cosas que para mí eran desconocidas. A cada uno de mis profesores por su dedicación en las clases.

Al Consejo Nacional de Ciencias y Tecnología (CONACyT) por otorgarme la beca de estudios y manutención por los dos años que duro la maestría.

Resumen

Este trabajo aborda el desafiante problema de la predicción de la trayectoria de los peatones, cuando las observaciones de estos peatones se pueden recopilar a través de un sistema de monitoreo de video urbano. Dado que la mayoría de los sistemas de vanguardia en este campo ahora se basan en redes neuronales recurrentes profundas, en este trabajo estudiamos una característica específica de estos sistemas, a saber, la forma en que codifican su entrada/salida. Para ello comparamos tres representaciones diferentes de la salida y una representación de entrada/salida, y mostramos que aquellas representaciones que trabajan con residuos (en particular, los desplazamientos con respecto a la última posición del peatón o modelos de regresión lineal de errores residuales) producen predicciones mucho más precisas que las que manejan coordenadas absolutas. Por otro lado se analiza la contribución de la información de pose para la inferencia de trayectorias y se propone una nueva forma de modelar las interacciones de los peatones con su contexto, por medio del *flujo óptico*.

Palabras clave: predicción de la trayectoria de los peatones; LSTM; Modelos de regresión basados en datos; Representaciones de salida/entrada; keypoints; Interacción social, flujo óptico.

Índice general

Siglas

ADE Average Displacement Error. 54, 115, 122, 124, 134

ANN Artificial Neural Network. 32

DNN Deep Neuronal Network. 32

FDE Final Displacement Error. 54, 122, 124, 134

LSTM Long Short Term Memory. 23, 40, 42, 44, 45, 56, 57, 70, 80, 81, 84

RNN Recurrent Neural Network. 22, 38, 40, 42

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación de la tesis

En los últimos años, pronosticar la ruta futura de peatones se volvió de mucha importancia para aplicaciones a vehículos no tripulados, robots humanoides y vehículos autónomos, entre otros, ya que, para cumplir con su objetivo, esos sistemas tienen que ir evadiendo obstáculos, y en particular los peatones que se encuentran en su mismo entorno.

En algunas situaciones, basta con analizar el ambiente en un instante t y tomar una decisión en base al análisis de éste. Sin embargo, en otros contextos, hay que *anticipar* cuáles serán los próximos movimientos que realizará un peatón, ya que esto permite a los vehículos inteligentes tomar medidas de seguridad, por ejemplo recalcular o desviar su trayectoria, para evitar colisiones.

Este problema de inferencia es desafiante, en particular cuando la gente camina en un espacio público abarrotado, como aceras, parques, centros comerciales, cruces o terminales de camiones. En este tipo de ambiente, los peatones siguen reglas de sentido común, tales como respetar el espacio personal, ceder el paso a otro peatón, pero las trayectorias resultantes pueden ser altamente complejas. Al mismo tiempo, los peatones son capaces de adaptarse al espacio físico y a los obstáculos en su camino.

Como lo veremos, muchos trabajos han abordado este problema recientemente, en particular en las comunidades de visión por computadora y robótica. En su forma más estándar, el problema se plantea simplemente como: “dada una serie de obser-

vaciones realizadas sobre un agente móvil, ¿cuál será su trayectoria en los próximos segundos?". La información observada, en la versión más básica del problema, es la sucesión de posiciones que tomó el agente en el tiempo (información espacial).

En este trabajo de tesis, se pretende analizar la incorporación de otro tipo de información, además de la información de coordenadas espaciales de las trayectorias, para usar en la predicción de posiciones futuras de peatones. En particular, investigamos el uso de la información de pose (postura) de los peatones así como el uso de información de flujo óptico. Estudiamos también diferentes formas de generar la salida del sistema, haciendo uso de sólo información de coordenadas espaciales. Todo esto lo evaluamos para realizar la predicción de trayectorias en diferentes escenarios (lo que nos lleva a proceder en evaluaciones sobre diferentes conjuntos de datos).

1.2. Estado del arte

En esta sección, presentamos brevemente una selección de diferentes trabajos sobre predicción de trayectorias y modelación de las interacciones.

Interacción Humano-Humano

Para predecir el comportamiento futuro de los peatones en las multitudes, se necesita modelar las interacciones entre los peatones. Uno de los primeros trabajos en esa dirección fue el de Helbing y Molnár [11], el cual propuso el modelo de Fuerzas Sociales, que modela el movimiento de los peatones utilizando fuerzas atractivas que los guían hacia el destino, y fuerzas repulsivas que aseguran la prevención de colisiones con obstáculos. No obstante, no modela la cooperación humano-humano y humano-robot. En el artículo de Trautman et al. [25], se hace uso de procesos Gaussianos para la descripción de las distribuciones predictivas de las trayectorias, y se plantea el uso de mezclas de ellos para lidiar con la multimodalidad generada por la presencia de múltiples objetivos potenciales. La interacción entre peatones se maneja similarmente a Helbing y Molnár [11]. Sin embargo, estos métodos utilizan funciones de potencial de energía hechas "a mano", basados en distancias relativas, con reglas específicas. En contraste, los métodos basados en datos, y en particular con RNNs, están flexibles y han superado a los métodos anteriores. Como lo veremos,

permiten mucho más flexibilidad y adaptación al contexto.

Predicción de Trayectorias Humanas

En el artículo de Alahi et al. [1], se aborda el problema de predicción con un nuevo modelo llamado *Social-LSTM*. Está basado en redes recurrentes de tipo Long Short Term Memory (LSTM) y puede predecir conjuntamente los caminos de todas las personas que se encuentran en la escena, teniendo en cuenta (al menos eso se espera) las reglas de sentido común y las convenciones sociales que los humanos suelen utilizar a medida que navegan. En particular, en este trabajo suponen que sólo las personas que se encuentran en una vecindad cercana al peatón, son capaces de afectar la trayectoria de éste. A estas interacciones, les modelan con una capa que le llaman agrupación social. Para ello, crean una rejilla de ocupación y, en base a ella, agrupan los estados ocultos de los LSTM de las personas que se encuentran en la vecindad. De esa manera, se crea un vector que tiene la información de la interacción social.

Otro de los trabajos que también han modelado la interacción social en lugares concurridos es el artículo de Vemula et al. [26], que, a diferencia del de Alahi et al. [1], toma en cuenta a personas que no están necesariamente cercanas al peatón. La idea es que, en ocasiones, las personas que se encuentran en nuestro alrededor podrían no ser tan importantes como otras personas que se encuentran más lejos, que sí podrían colisionar con nosotros en el futuro. A dicho modelo lo nombraron Atención Social, y en él toman en cuenta las distancias relativas que hay entre los peatones, la velocidad, el tiempo de colisión, la aceleración y el rumbo. El modelo resultante captura la influencia de cada persona sobre la otra y la naturaleza de su interacción, y predice sus trayectorias futuras, haciendo uso de una mezcla de redes neuronales recurrentes.

El trabajo de Pfeiffer et al. [21], es el primer enfoque que utiliza LSTM incorporando tanto obstáculos estáticos como peatones circundantes para la predicción de trayectorias. Para codificar a los peatones circundantes, proponen una forma de modelado en una rejilla peatonal angular, que combina una baja dimensionalidad con un gran contenido de información. Para los obstáculos fijos, usan una cuadrícula ocupacional 2D local, centrada en el peatón observador.

Una de las investigaciones que también hace uso de los estados ocultos de las LSTM's es el de Zhang et al. [28], en el cual se muestra que es importante actualizar los estados de los vecinos para la inferencia de interacción, ya que muchos métodos como [1, 10], se basan en estados ocultos vecinos anteriores, pero ignoran la importancia de la intención actual de los vecinos. Otro de los objetivos de esta investigación es darle un trato individual a las características (patrones de movimiento) de los peatones vecinos, ya que no son igualmente importantes para predecir la trayectoria de un peatón, y esto depende de las diferentes condiciones de interacción que haya entre el peatón al cual se le va a predecir su trayectoria y cada uno de sus vecinos.

Uno de los trabajos más recientes en esta tarea (*predicción de trayectorias*), es el de Giuliari et al. [9]. Este deja de lado los mecanismos sociales y de mapeo que se usan en los modelos con LSTM's, y propone modelar las trayectorias de los peatones de forma individual por medio de redes Transformer, las cuales utilizan únicamente la atención en vez del procesamiento secuencial. Dicho de otra manera, mira todas las observaciones disponibles y las pondera según el mecanismo de atención.

Modelos Generativos

A diferencia de otros trabajos que predicen una trayectoria futura única, en los modelos basados en redes adversarias generativas (GANs), se predicen múltiples trayectorias de una aproximación a la distribución predictiva. Eso permite contemplar casos donde el futuro de una trayectoria es, por ejemplo, multimodal. En particular, en el artículo de Gupta et al. [10], se predicen múltiples trayectorias futuras socialmente aceptables, es decir tales que los peatones tienen que cumplir con reglas de convivencia social. Otro de los trabajos realizados con GANs es el de Sadeghian et al. [22] que, para predecir las trayectorias futuras, modela no sólo la influencia de la propia historia del estado, sino también la del estado de otros agentes y la naturaleza del terreno físico alrededor de su camino.

En el artículo *Social-BIGAT* de Kosaraju et al. [13], proponen una red adversarial generativa basada en gráficos. Genera predicciones de trayectorias multimodales al modelar de una mejor forma las interacciones sociales de los peatones en una escena. Esto lo consiguen usando redes de atención de gráficos, sobre las escenas, y así obtienen características que codifican las interacciones sociales.

En el trabajo de Ivanovic y Pavone [12], combinan elementos de los modelos generativos profundos variacionales (CVAE), modelos de secuencia recurrente (LSTM) y estructuras gráficas espacio-temporales dinámicas para producir trayectorias multimodales de alta calidad que modelan y predicen los comportamientos futuros. Contempla desafíos importantes que tienen los seres humanos: altamente multimodales, dinámicos y variables. Otro trabajo relacionado a este es el *Trajectron++* de Salzmann et al. [23], el cual también desarrolla un modelo de predicción de múltiples agentes que contempla la dinámica de los agentes, y en particular de los vehículos terrestres; produce predicciones condicionadas por las posibles trayectorias futuras de los robots, útiles para la planificación inteligente teniendo en cuenta las respuestas humanas. Proporciona un enfoque de aplicación general, abierto y extensible que puede utilizar datos heterogéneos sobre el entorno circundante. Puede agregar información ambiental por ejemplo, mapas, imágenes de cámara, lidar, lo que permite producir predicciones que difieren según la estructura de la escena.

En este trabajo, para limitar el alcance de la tesis, se decidió no abordar el tema de múltiples predicciones, para enfocar el estudio a los puntos llave que hemos mencionado.

Modelos Multitareas

Recientemente, se publicó el trabajo de Liang et al. [16], que es una arquitectura multitarea, que estudia la predicción del futuro camino de un peatón junto con actividades futuras. Esta arquitectura multitarea utiliza características visuales ricas sobre la información del comportamiento humano y la interacción con su entorno.

1.3. Contribuciones de la tesis

Nuestras contribuciones son las siguientes:

- proponemos una nueva forma de tomar en cuenta la interacción social haciendo uso del flujo óptico simulado; de esa manera, modelamos la percepción humana de manera más natural y así tomamos en cuenta cada uno de los obstáculos que se encuentran en su alrededor;

- evaluamos cuantitativamente el uso de la información de pose, en conjuntos de datos donde esta información se puede extraer (es decir, con imágenes de suficientemente alta resolución);
- mostramos que la precisión de los modelos es sensible a la forma de representar el output de la predicción.

Para abordar este problema de predicción, usamos esencialmente redes neuronales recurrentes. En los últimos años, han demostrado tener éxito en problemas de esta naturaleza. Veremos que, en la gran mayoría de los casos, nuestros modelos son entrenados con datos extraídos de secuencias de vídeo.

1.4. Organización del documento

A continuación se presenta un esquema de la estructura de la tesis, que consta de seis capítulos, el contenido de cada capítulo así como las principales contribuciones.

1. En el Capítulo 1 se presenta una introducción general de la literatura revisada de los resultados del estado del arte en la predicción de trayectorias de peatones y se describe las contribuciones de la tesis.
2. En el Capítulo 2 se introduce la notación que se usará en los siguientes capítulos. Se plantea el problema, así como se proporcionan conceptos importantes sobre Redes Neuronales, en particular sobre Redes Neuronales Recurrentes. Se describe que es un mecanismo de atención. Se explica brevemente el sistema con el cual se detectan los puntos claves del esqueleto humano, los cuales serán usados en el Capítulo 4. Posteriormente se describe la naturaleza de los datos con los que se va a trabajar y finalmente se dan las métricas con las cuales serán medidas las trayectorias predichas.
3. En el Capítulo 3 se presenta la arquitectura de la red con la cual se evaluarán las distintas representaciones de entrada/salida del sistema. Se describe cada una de las representaciones de entrada/salida propuestas. Se evalúan las representaciones y por último de comparan los resultados obtenidos con el estado del arte.

4. En el Capítulo 4 se evalúa la contribución de la pose en la predicción de trayectorias de peatones. Así que primero se explica la preparación de los datos que serán usados para el entrenamiento de la red. Se describe la red neuronal con la cual se medirá la contribución de la pose. Se da la arquitectura de la red neuronal. Se presentan los experimentos con los que se mide la contribución de la pose al predecir.

5. En el Capítulo 5 se presenta como se ha modelado la interacción de los peatones. Se describe que es el flujo óptico y como simular el flujo óptico percibido por cada peatón a partir de los movimientos relativos de sus vecinos. Se explica como modelar la interacción social del peatón con sus vecinos por medio del flujo óptico. Se incorporan los mapas semánticos en el modelo de interacción social. Se preparan los datos para entrenar la red neuronal y se ilustra la arquitectura de la red. Se evalúa el modelado de interacción social.

6. En el Capítulo 6 se resume los principales resultados y contribuciones de esta tesis, y también se indican algunas posibles direcciones de trabajo a futuro.

Capítulo 2

Marco Teórico

En este capítulo revisaremos diferentes conceptos y notaciones necesarias para leer con más facilidad el desarrollo de esta tesis.

Por lo que el resto del capítulo estará organizado en las siguientes secciones. En la Sección 2.1 se describe la notación que se usará en el escrito. En la Sección 2.2 se plantea el problema. En la Sección 2.3 se describen conceptos importantes sobre Redes Neuronales. Mientras que en la Sección 2.4 se explica brevemente como la información de pose es extraída de las imágenes. Posteriormente en la Sección 2.5 se describe brevemente la naturaleza de los conjuntos de datos con los cuales se evaluarán nuestros modelos. En la Sección 2.6 se plantea los dos esquemas de entrenamiento que serán usados al momento de entrenar los modelos y en la Sección 2.7 se definen las métricas que evaluarán las trayectorias predichas por los modelos.

2.1. Notación

Esta sección describe las principales convenciones que usaremos para nuestras formulaciones matemáticas, así como notaciones usuales:

- Para referirnos a vectores, usamos letras negritas.
- Para referirnos a escalares, usamos letras minúsculas normales.
- Los índices $i, j \in \{1, ..., N\}$ se refieren generalmente a la identidad de peatones, donde N es el número total de peatones en una escena bajo consideración.

- El tiempo se considera discretizado y una **observación** del peatón i en el tiempo t es denotada como:

$$\mathbf{x}_t^i \triangleq (x_t^i, y_t^i)^T \in \mathbb{R}^2.$$

- $\mathbf{x}_{1:T_{obs}}^i \triangleq \{\mathbf{x}_1^i, ..., \mathbf{x}_{T_{obs}}^i\}$ es el conjunto de observaciones **recolectadas** sobre el peatón $i \in \{1, ..., N\}$ en todo el intervalo $[1, T_{obs}]$.

- $\hat{\mathbf{x}}_{T_{obs}+1:T_{pred}}^i \triangleq \{\hat{\mathbf{x}}_{T_{obs}+1}^i, ..., \hat{\mathbf{x}}_{T_{pred}}^i\}$ es el conjunto de las posiciones **predichas** por un algoritmo de predicción para el peatón $i \in \{1, ..., N\}$ en el intervalo $[T_{obs} + 1, T_{pred}]$.

2.2. Planteamiento del problema

Figura 2.1: La definición de los Bounding Boxes asociados a peatones se hace con los cuatro escalares **bL**-bodyLeft, **bR**-bodyRight, **bT**-bodyTop y **bB**-bodyBottom.

Se supone que cada escena (típicamente, proveniente de una secuencia de vídeo) es preprocesada para obtener las coordenadas espaciales de todos los peatones pre-

sentes, en diferentes instantes de tiempo, a través de sus Bounding Boxes (cajas englobantes), como ilustrado en la Fig. 2.1. Con la información de coordenadas espaciales, se construyen secuencias de posiciones de tamaño T_{obs}, es decir, $\mathbf{x}^i_{1:T_{obs}}$, con $i \in \{1, \ldots, N\}$, los cuales son la información que observamos.

Problema: Predecir la trayectoria de cada peatón para los siguientes $T_{pred} - T_{obs}$ instantes de tiempo, es decir estimar $\mathbf{x}^i_{T_{obs}+1:T_{pred}}$ dado $\mathbf{x}^i_{1:T_{obs}}$.

Hay que mencionar que hay dos formas de predecir (decodificar) los instantes de tiempo $\{T_{obs}+1, ..., T_{pred}\}$ para el peatón i:

- *Consecutiva*: el modelo aprende a predecir un solo paso después de lo observado,

$$\mathbf{x}^i_{1:T_{obs}} \rightarrow \hat{\mathbf{x}}^i_{T_{obs}+1} \tag{2.1}$$

  por lo que la posición $\hat{\mathbf{x}}^i_{T_{obs}+1}$ pasaría a formar parte de las posiciones observadas (nos olvidamos de la posición $\mathbf{x}^i_1$) y aplicamos de nuevo el modelo

$$\mathbf{x}^i_{2:T_{obs}}, \hat{\mathbf{x}}^i_{T_{obs}+1} \rightarrow \hat{\mathbf{x}}^i_{T_{obs}+2} \tag{2.2}$$

  y así se procede sucesivamente, hasta el número de posiciones que se quieran predecir.

- *Bloque*: el modelo aprende a predecir un número constante n de posiciones después de los observados,

$$\mathbf{x}^i_{1:T_{obs}} \rightarrow \hat{\mathbf{x}}^i_{T_{obs}+1:T_{pred}} \tag{2.3}$$

Como datos condicionados adicionales, la información espacial observada $\mathbf{x}^i_{1:T_{obs}}$ podrá estar acompañada por:

- información de apariencia sobre el peatón i (este tema será desarrollado con más detalles en el capitulo 4);

- información espacial de los otros peatones al mismo momento que i, $\mathbf{x}^j_{1:T_{obs}}$ con $j \neq i$ (este tema será desarrollado con más detalles en el capitulo 5).

Para este trabajo, usaremos $T_{obs} = 8$ y $T_{pred} = 12$, esto es porque en el estado de arte son los valores estándar. En el Capítulo 3, se estudiarán otros valores para la variable T_{pred}, mientras que en los capítulos posteriores T_{pred} siempre tomará el valor predeterminado de 12. Lo hacemos así para poder comparar nuestros resultados con los resultados del estado de arte.

2.3. Redes Neuronales Profundas

En lo siguiente, procedemos a revisar brevemente los conceptos de redes neuronales necesarios para seguir con más facilidad los métodos de regresión basados en redes neuronales que proponemos en los siguientes capítulos.

2.3.1. Generalidades

Una red neuronal profunda (DNN por sus iniciales en inglés) es una red neuronal artificial (ANN) con varias capas ocultas entre las capas de entrada y salida. Al igual que en las ANN poco profundas, las DNN pueden modelar relaciones no lineales complejas entre datos de entrada y de salida. Los modelos de aprendizaje profundo han llegado a producir resultados mucho mejores que métodos clásicos de aprendizaje máquina y, en varios problemas, han permitido establecer nuevos desempeños de referencia.

El propósito principal de una red neuronal es recibir un conjunto de datos (un vector, por ejemplo)

$$\{x_1, ..., x_n\} \tag{2.4}$$

de entradas, realizar cálculos progresivamente complejos en ellas y generar como salida

$$\{y_1, ..., y_s\} \tag{2.5}$$

para resolver problemas como clasificación o regresión, que se materializan en el área de visión computacional en reconocimiento de objetos, segmentación de imágenes, estimación de disparidad, etc. En nuestro caso, el problema que queremos resolver es el de regresión sobre datos de trayectorias, es decir queremos predecir los valores (escalares) futuros de las coordenadas x e y de un objeto móvil, dado un histórico

de ellos.

Como se ilustra en la Fig. 2.2, en una red profunda (o aun en las redes mas sencillas), tenemos una entrada, una salida y un procesamiento secuencial ("feedforward") de los datos para transformar la entrada en la salida.

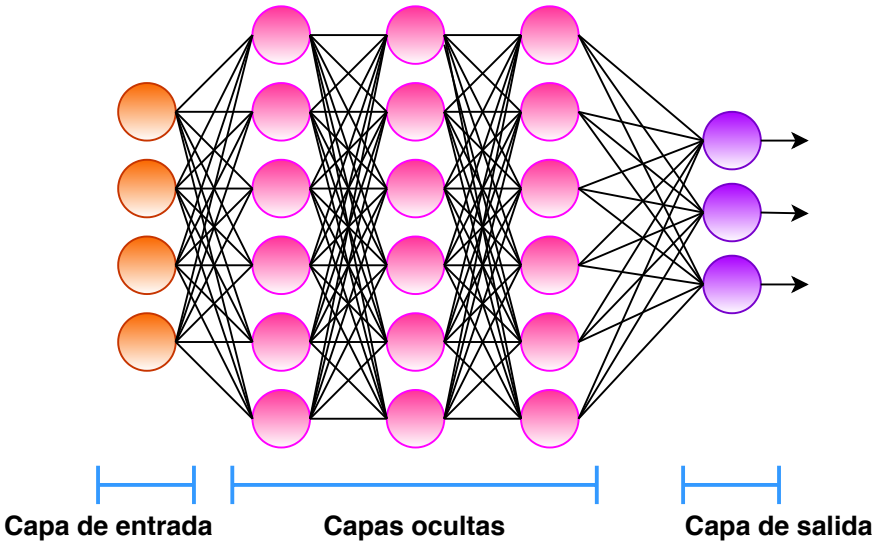


Figura 2.2: Red Neuronal Profunda.

Las redes neuronales se utilizan ampliamente en aprendizaje supervisado y problemas de aprendizaje de refuerzo. Estas redes se basan en un conjunto de capas conectadas entre sí, donde cada capa está compuesta por un conjunto de neuronas.

La i-ésima neurona de la capa oculta recibe un conjunto de entradas $\{x_1, ..., x_n\}$, que pueden ser los datos de entrada de la red o los datos de salida de la j-ésima neurona de la capa anterior. A las entradas de la neurona i se les aplica una función afín como sigue

$$z \triangleq \sum_{j=1}^{n} w_{ij} x_j + b_i, \tag{2.6}$$

donde el elemento w_{ij} de la matriz $\mathbf{W}$ es el peso de conexión entre la neurona j de la capa anterior y la neurona oculta i. En el proceso de aprendizaje, los pesos son los que se van modificando y b_i se denomina el vector de sesgo (*bias*) de la neurona

i. Posteriormente, al valor z, se le aplica una función de activación f (generalmente no lineal), así generando el valor de salida de la neurona i.

Las salidas de una red neuronal se obtienen por la combinación de funciones lineales y no lineales que se van aplicando a través de las diferentes capas, como descrito en el párrafo anterior. En aprendizaje profundo, la cantidad de capas ocultas, en su mayoría no lineales, puede ser grandes, del orden de 1000 capas.

Funciones de activación

Las funciones de activación se aplican en cada neurona de una red neuronal y la utilidad más importante que tienen es indicar cuándo y cuánto una neurona se activa o se apaga. Dependiendo de la función de activación que se use, ésta tendrá ciertos limites. Recordemos que primero se calcula la función z (combinación lineal de la Eq. 2.6) y luego el resultado de esta función se le pasa a la función de activación. Esta última observa si los datos tienen los patrones que requiere la neurona o si no los tienen.

Se buscan funciones que tengan derivadas simples, para minimizar el costo computacional y permitir la optimización de los pesos de la red de forma eficiente.

Las siguientes son algunas funciones de activación, que ilustramos también en la Fig. 2.3:

- **Sigmoide:** La función sigmoide transforma los valores introducidos a una escala $(0, 1)$. Para valores altos de x, la función tiende de manera asintótica a 1 y para valores muy bajos, tiende de manera asintótica a 0,

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.7}$$

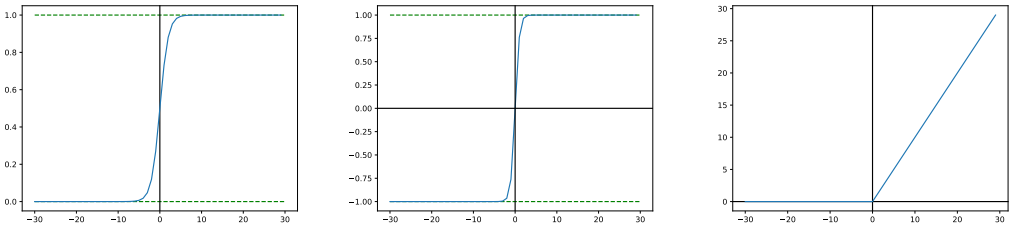
- **Tangente hiperbólica:** La función tangente hiperbólica transforma los valores introducidos a una escala $(-1, 1)$. Para valores altos de x, tiende de manera asintótica a 1 y para valores muy bajos, tiende de manera asintótica a -1,

$$f(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{2.8}$$

  Esta función de activación tiene buen desempeño en redes recurrentes [5].

- **ReLU:** La función ReLU transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran:

$$f(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ \\ x & \text{si} \quad x \geq 0. \end{cases} \tag{2.9}$$



(a) Función Sigmoide (b) Función Tanh (c) Función ReLU

Figura 2.3: Algunas funciones de activación.

Función de pérdida

La función de pérdida, también conocida como función de costo, es la función que nos dice que tan buena es la red para una cierta tarea. Ésta es la función que optimizamos o minimizamos por el algoritmo de *backpropagation* (o simplemente de derivación por regla de la cadena), que describiremos a continuación. Dicha función de pérdida recibe como parámetros los valores que la red predice como salida y los valores que quisiéramos que la red aprendiera (*ground truth*) y evalúa la distancia entre ambos. El algoritmo de *backpropagation* es esencialmente un método de descenso de gradiente que optimiza los parámetros de la red al minimizar la función de pérdida.

A continuación menciono algunas funciones de pérdida, donde x es es el error, la diferencia entre y_{true} y y_{pred}, para cada una de estas funciones.

- **Logcosh:**

$$\text{loss\_logcosh(x)} = \text{reduce\_mean(logcosh(x), axis= -1)} \tag{2.10}$$

donde $\log(\cosh(x)) = \log\left(\frac{\exp(x)+\exp(-x)}{2}\right) = x+\log(\exp(-2x)+1)-\log(2)$.

- **Error cuadrático medio:**

$$\text{loss\_mse(x)} = \text{reduce\_mean}(x^2, \text{axis} = -1). \tag{2.11}$$

La primera es más robusta, ya que valores más altos del error contribuyen de forma lineal, y no cuadrática, al total.

2.3.2. Entrenamiento de una red neuronal

El algoritmo de *backpropagation* permite entrenar a las redes de múltiples capas de manera supervisada, por descenso de gradiente. Optimiza los pesos de cada una de las capas ocultas, para obtener mejores resultados en la función de pérdida.

En el proceso de aprendizaje, a la red se le proporciona un dato p, $\{x_1^p, ..., x_n^p\}$, como entrada de la red. Éste se propaga a través de los pesos w_{ij} desde la capa de entrada hacia las capas ocultas. En el caso particular de la capa de salida, denotaremos sus pesos con v_{ki}. Las neuronas de las capas intermedias transforman las señales recibidas por medio de la aplicación de la función de activación, generando de este modo, un valor de salida. Posteriormente, éste se transmite a través de los pesos v_{ki}, hacia la capa de salida, donde, aplicando la misma operación que en el caso anterior, las neuronas de esta última capa proporcionan la salida de la red. A continuación, este proceso se resume:

1. La entrada que recibe una neurona oculta i es $\{x_1^p, ..., x_n^p\}$. A esa entrada se le aplica la función afín z con la cual se obtiene $\sum_{j=1}^{n} w_{ij}x_j + b_i$.

2. El valor de salida y_i^p de la neurona oculta i se obtiene aplicando una función de activación f sobre el valor antes obtenido:

$$y_i^p = f(\sum_{j=1}^{n} w_{ij}x_j^p + b_i). \tag{2.12}$$

3. De manera análoga, la entrada que recibe una neurona k de la capa de salida se transforma como sigue:

$$\sum_{i=1}^{l} v_{ki} y_i^p + c_k. \tag{2.13}$$

   donde c_k es el bias de la capa de salida.

4. Por último, el valor de salida de la neurona k de la capa de salida es:

$$y_k^p = f(\sum_{i=1}^{l} v_{ki} y_i^p + c_k). \tag{2.14}$$

Una función de error clásica (pero lejos de ser única) que se puede minimizar está dada por un promedio del error cuadrático sobre todo el conjunto de entrenamiento (es decir, iterando sobre los m datos p de este conjunto de entrenamiento):

$$E = \frac{1}{2} \sum_{p=1}^{m} \sum_{k=1}^{s} (d_k^p - y_k^p)^2, \tag{2.15}$$

donde d_k^p es la salida deseada para la neurona k de la capa de salida.

La base del algoritmo de *backpropagation* para modificar los pesos es el gradiente descendente. Si calculamos el gradiente de $E^p \triangleq \frac{1}{2} \sum_{k=1}^{s} (d_k^p - y_k^p)^2$, con respecto a cada uno de los pesos, se encontraría la dirección que determina el incremento más rápido en el error para este dato p, mientras que la dirección opuesta determina el decremento más rápido del error. Por esa razón, el error puede reducirse ajustando cada peso en la dirección:

$$-(\sum_{p=1}^{m} \frac{\partial E^p}{\partial w_{ij}})_{i,j}. \tag{2.16}$$

Por lo tanto, la forma de actualizar los pesos de forma iterativa consiste en aplicar la regla de la cadena a la expresión del gradiente y multiplicar el gradiente calculado por una tasa de aprendizaje. Así, para una neurona de salida:

$$\Delta v_{ki}(r+1) = -\eta \frac{\partial E}{\partial v_{ki}} = -\eta \sum_{p=1}^{m} \frac{\partial E^p}{\partial v_{ki}} = \eta \sum_{p=1}^{m} \delta_k^p y_i^p, \tag{2.17}$$

donde $\delta_k^p = (d_k^p - y_k^p) f'(a_k^p)$, $a_k^p = \sum_{i=1}^{l} v_{ki}(r) y_i^p + c_k$, y r indica la iteración.

En cambio, para una neurona de una capa oculta, la actualización es:

$$\Delta w_{ij}(r+1) = \eta \sum_{p=1}^{m} \delta_i^p x_j^p, \tag{2.18}$$

donde $\delta_i^p = f'(a_i^p) \sum_{k=1}^{s} \delta_k^p v_{ki}$, $a_i^p = \sum_{j=1}^{n} w_{ij}(r) x_j^p + b_i$.

Se puede observar que el error o valor delta asociado a una neurona oculta i, está determinado por la suma de los errores que se cometen en las k neuronas de salida que reciben como entrada la salida de esa neurona oculta i. Por esa razón, el algoritmo también recibe el nombre de propagación hacía atrás.

A continuación, presentamos las *Redes Neuronales Recurrentes* (RNN por sus siglas en inglés), que son redes neuronales profundas adecuadas para procesar datos secuenciales (series de tiempo).

2.3.3. Redes Neuronales Recurrentes

Los seres humanos tenemos la capacidad de persistencia de memoria, lo cual es de ayuda para tomar decisiones futuras tomando en cuenta experiencias previas.

Desafortunadamente, las redes neuronales tradicionales no dan un cuadro natural para implementar esa idea de persistencia. Por ejemplo, si se imagina que se desea clasificar qué tipo de evento está ocurriendo en cada punto de una película, no es claro cómo una red neuronal tradicional podría usar su razonamiento sobre eventos anteriores en la película para "informar" a los posteriores. Las redes neuronales recurrentes, ilustradas en la Fig. 2.4, permiten atender este problema. Son redes con bucles sobre sí misma, es decir tal que su output es usado como input en iteraciones consecutivas, permitiendo que la información persista y se propague a tiempos consecutivos.

Estos bucles hacen que las redes neuronales recurrentes parezcan algo más difícil de conceptualizar. Sin embargo, no son tan diferentes a una red neuronal normal. Una red neuronal recurrente se puede considerar como copias múltiples de la misma red, donde cada una de las instancias de ella pasa un mensaje a un sucesor. Considera por ejemplo en la figura 2.5 lo que pasa si desenrollamos el bucle.

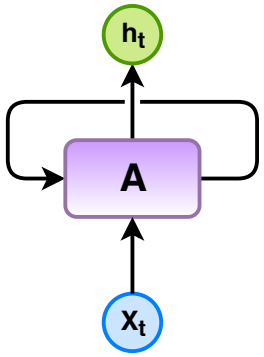


Figura 2.4: Las redes neuronales recurrentes tienen bucles (figura inspirada por Colah's blog [19]).

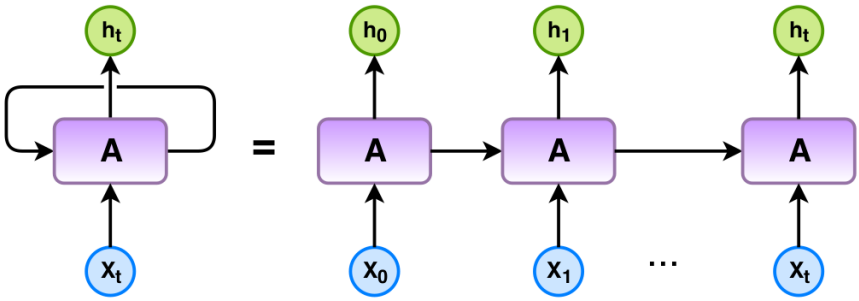


Figura 2.5: Una red neuronal recurrente desenrollada (figura inspirada por Colah's blog [19]).

Ésta es una RNN típica. Obsérvese que las imágenes de la derecha no son capas múltiples, sino la misma capa que se desenrolla en el tiempo en que las salidas se devuelven a la capa oculta.

Las redes neuronales recurrentes recuerdan el estado de una entrada de tiempos anteriores, lo que le ayuda a tomar una decisión para el tiempo futuro.

Ahora, hay un pequeño problema con este concepto. El modelo se entrena utilizando el algoritmo de propagación hacia atrás (*backpropagation*), en el que acumulamos los gradientes (errores) de la capa de salida y los devolvemos a toda la red. A medida que el número de las funciones de activación y la cantidad de pasos de tiempo en problemas del mundo crecen (por ejemplo, si consideramos secuencias muy largas), los gradientes tienden a volverse pequeños (y, en un caso extremo, a acercarse de cero). Esto se llama el problema de degradación del gradiente (*vanishing gradient*).

Para superar esto, se desarrolló variantes mejoradas de las RNNs. Una de esas se denomina redes de memoria a corto y largo plazo (LSTM por sus siglas en inglés). Las LSTM fueron introducidos por Hochreiter y Schmidhuber (1997) y se desarrollaron para resolver el problema de los gradientes, mediante la introducción de varias compuertas que dan al modelo la posibilidad de decidir qué información conservar y qué olvidar. Dichas redes trabajan bien en una gran variedad de problemas, tales como reconocimiento de voz, modelado del lenguaje, traducción, generación de subtítulos de imágenes.

Long short-term memory

Las RNNs tienen la forma de una cadena de módulos repetidos de una misma red neuronal. En la RNN estándar, este módulo repetido tiene una estructura muy simple, como una sola capa no-linear de tipo tangente hiperbólica.

En la figura 2.6, se puede observar una celda RNN básica, donde $\mathbf{h}_{t-1}$ es la salida del paso de tiempo anterior. Esa salida se pasa a la función de activación tanh junto con la nueva entrada $\mathbf{x}_t$.

Los LSTMs tienen la misma estructura tipo cadena de los RNNs, pero el módulo repetido tiene una estructura diferente. En lugar de tener una sola capa interna, la red neuronal tiene cuatro capas que interactúan de una manera especial.

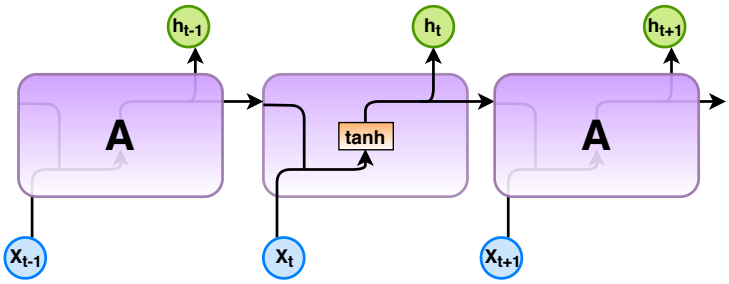


Figura 2.6: El módulo repetido en una RNN estándar contiene una sola capa (figura inspirada por Colah's blog [19]).

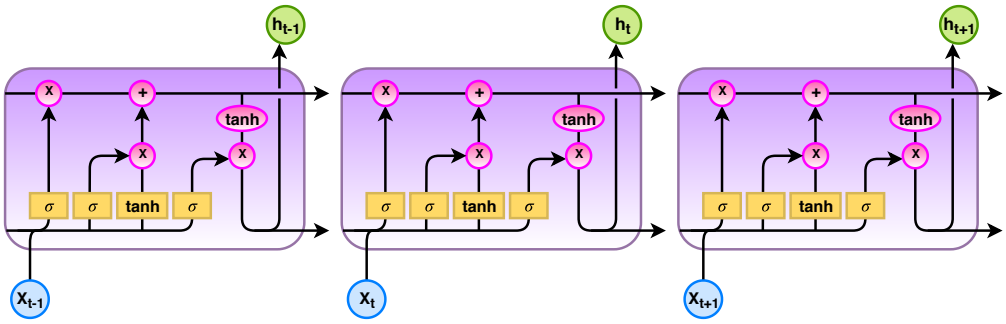


Figura 2.7: Una LSTM estándar desenrollada (figura inspirada por Colah's blog [19]).

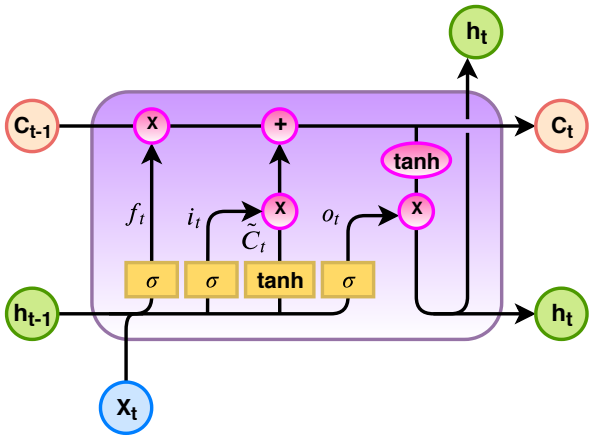


Figura 2.8: Celda básica de una LSTM (figura inspirada por Colah's blog [19]).

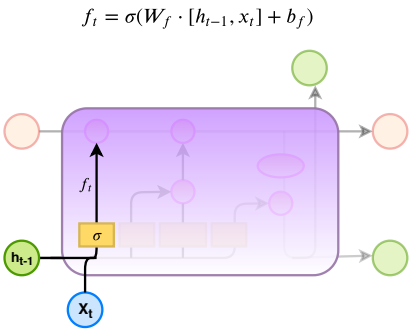


Figura 2.9: Capa de olvido de una celda LSTM (inspirado por Colah's blog [19]).

Descripción paso a paso de una celda LSTM

A diferencia de la arquitectura de RNN básica de la Fig. 2.6, se tiene una nueva variable $\mathbf{C}_{t-1}$, la cual es conocida como estado de la celda anterior.

La celda LSTM combina cuatro pasos:

Capa de olvido. Ilustrada en la Fig. 2.9, es la responsable de decidir qué información retener del estado de la celda anterior y qué información se debe olvidar o eliminar. Esta decisión la toma una capa **sigmoide**, la cual concatena a $\mathbf{h}_{t-1}$ y $\mathbf{x}_t$, lo multiplica por un operador lineal $\mathbf{W}_f$, le agrega un bias $\mathbf{b}_f$, y genera (por la función sigmoide) un número entre 0 y 1, eso para cada dimensión en el estado de la celda anterior $\mathbf{C}_{t-1}$. Uno significa mantener completamente presente la información, mientras 0 significa deshacerse completamente de ella.

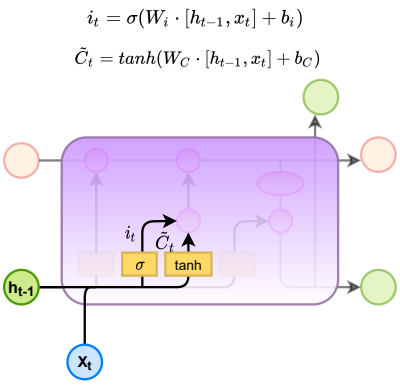


Figura 2.10: Capa de permanencia (inspirado por Colah's blog [19]).

Capa de permanencia. Ilustrada en la Fig. 2.10, decide qué información *nueva* se va a incorporar en el estado de la celda. Consta de dos partes. Primero, una capa sigmoide llamada puerta de entrada decide qué componentes/dimensiones del estado se actualizarán. A continuación, una capa tanh crea un vector de nuevos candidatos $\widetilde{\mathbf{C}}_t$ que pueden agregarse al estado. En el siguiente paso, se combinan estos dos ($\mathbf{C}_{t-1}$ y $\widetilde{\mathbf{C}}_t$) para crear una actualización del estado.

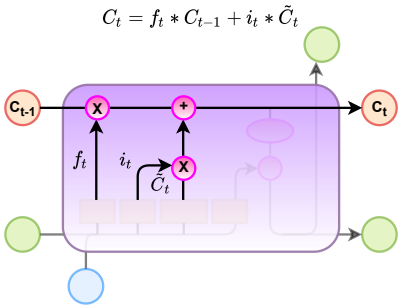


Figura 2.11: Capa de actualización (inspirado por Colah's blog [19]).

Capa de actualización del estado de la celda. Ilustrada en la Fig. 2.11, actualiza el estado de la celda anterior $\mathbf{C}_{t-1}$, en un nuevo estado de celda $\mathbf{C}_t$. Los pasos anteriores decidieron qué hacer, ahora sólo se actualiza. Se multiplica (elemento por elemento) $\mathbf{C}_{t-1}$ por $\mathbf{f}_t$, donde $\mathbf{f}_t$ es la salida de la capa de olvido, para efectivamente olvidar (según la proporción indicada por $\mathbf{f}_t$) las cosas que se decidieron olvidar antes. Luego se agrega la multiplicación elemento-por-elemento de $\mathbf{i}_t$ por $\widetilde{\mathbf{C}}_t$, donde $\mathbf{i}_t$

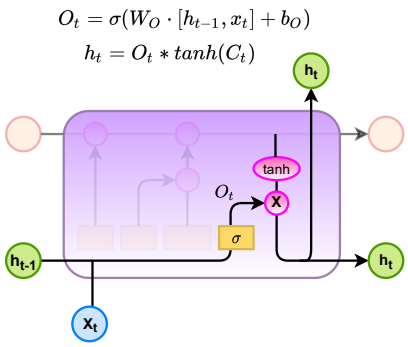


Figura 2.12: Capa de salida (inspirado por Colah's blog [19]).

es la salida de actualización. Éstos son los nuevos valores candidatos, escalados por cuánto se decidió actualizar cada valor del estado.

Capa de salida. Ilustrada en la Fig. 2.12, esta capa hace uso de toda la información recopilada en las últimas tres capas para producir una salida $\mathbf{h}_t$ que tiene características del paso de tiempo actual y también de varios pasos de tiempo anteriores. La salida se basa en el estado de celda $\mathbf{C}_t$, pero es una versión filtrada. Primero, se ejecuta una capa sigmoide que decide qué partes del estado de la celda va a formar parte de la salida (de cierta manera, es una selección de características), y se coloca el estado de la celda a través de tanh y se multiplica por la salida de la sigmoide, de modo que sólo se emite las partes que se decidieron.

El vector de salida $\mathbf{h}_t$, es de la dimensión del número de unidades de una celda LSTM. Dicho vector también es conocido como *estado oculto de la celda LSTM*, en el tiempo t y haremos muchas referencias a él en este documento.

2.3.4. Mecanismo de atención

Los mecanismos de atención tienen un papel importante en muchas aplicaciones de aprendizaje máquina. Una de sus primeras variantes fue propuesta en el trabajo de Xu et al. [27], para la descripción del contenido de una imagen. Este mecanismo de atención fue inspirado por los recientes usos y éxitos de la atención en tareas como reconocimiento de objetos y traducción automática. Lo que hace es generar un

vector de **contexto** $\mathbf{z}_t$, que en el caso particular de la tarea de descripción de una imagen, da una representación dinámica de las partes relevantes de la imagen en un cierto instante de tiempo t.

A continuación explicaremos brevemente la formulación del uso de atención en el caso de descripción de una imagen. Para este problema, se toma como entrada de una red neuronal una imagen $\mathbf{I}$ y se obtiene como salida un subtítulo (texto) $\mathbf{y}$, el cual es codificado como una secuencia de vectores de longitud K que codifican palabras

$$\mathbf{y} = \{\mathbf{y}_1, ..., \mathbf{y}_C\}, \mathbf{y}_i \in \mathbb{R}^K, \tag{2.19}$$

donde K es el tamaño del vocabulario y C la longitud del subtítulo. Se espera que el subtítulo sea una leyenda apropiada de la imagen.

Para la extracción de características de la imagen, se usa (**VGGnet**), una red neuronal convolucional descrita en el trabajo de Simonyan y Zisserman [24], la cual extrae un conjunto $\mathbf{a}$ de L vectores de características de dimensión D, a los cuales se les conoce como vectores de anotación. Dicho extractor obtiene L vectores, y cada uno de ellos es una representación de alguna parte de la imagen

$$\mathbf{a} = \{\mathbf{a}_1, ..., \mathbf{a}_L\}, \mathbf{a}_i \in \mathbb{R}^D. \tag{2.20}$$

Para la decodificación de la leyenda, se usa una red LSTM (ver Sección 2.3.3) que produce de output una palabra $\mathbf{y}_t$ en cada instante de tiempo t, condicionado a un vector contexto $\mathbf{z}_t$, al estado oculto del instante de tiempo anterior y a las palabras generadas anteriormente. En lo siguiente, veremos cómo se genera $\mathbf{z}_t$.

Mecanismo de Atención Suave

Las redes recurrentes como las LSTMs exhiben cierta debilidad al momento de reusar información pasada, es por ello que utilizar un mecanismo de atención es de gran ayuda, ya que estos centran cierta atención en diferentes instantes de tiempo dentro de la información que se le da.

El mecanismo de atención suave usa (implícitamente) el valor esperado del vector de contexto $\mathbf{z}_t$, evaluado entre todas las posiciones posibles de la imagen

$$\mathbb{E}_{p(s_t|\mathbf{a})}[\mathbf{z}_t] = \sum_{i}^{L} \alpha_{t,i}\mathbf{a}_i. \tag{2.21}$$

donde los coeficientes $\alpha_{t,i}$ pueden interpretarse como la probabilidad de que la posición i sea la más adecuada para enfocarse (s_t), al momento de predecir la siguiente palabra. Estos coeficientes de ponderación (o probabilidades) se calculan de dos maneras posibles:

$$e_{t,i} = f_{att}(\mathbf{a}_i, \mathbf{h}_{t-1}) = \begin{cases} \mathbf{a}_i^T \mathbf{W}_a \mathbf{h}_{t-1} & \text{(caso general)} \\ \\ \mathbf{V}_a^T \tanh\left(\mathbf{W}_a[\mathbf{a}_i, \mathbf{h}_{t-1}]\right) & \text{(por concatenación)} \end{cases} \tag{2.22}$$

$$\alpha_{t,i} = \frac{\exp\left(e_{t,i}\right)}{\sum_{j=1}^{L} \exp\left(e_{t,j}\right)}, \tag{2.23}$$

donde $\mathbf{W}_a$, $\mathbf{V}_a$ se aprenden como parte de los parámetros de la red.

La idea de este mecanismo es aprender a **seleccionar la informaciíon relevante** dentro de la imagen (los vectores $\mathbf{a}_i$ más relevantes) para la tarea de predicción de la secuencia de palabras que forma la leyenda de la imagen. Este proceso de selección puede involucrar (en el caso general descrito arriba) un calculo de producto escalar/correlación de versiones transformadas (por $\mathbf{W}_a$) de los features de interés y de los vectores de estado de la RNN.

Mecanismo de Atención Suave con Datos de Trayectorias

Con datos secuenciales, como en esta tesis, procederemos **con un mecanismo similar** al descrito arriba. Para realizar la predicción de una futura posición de una persona, usaremos un **vector de contexto** que nos permitirá seleccionar (de manera suave, como suma ponderada de vectores de características) elementos de contexto. Veremos que este mecanismo procederá en varias dimensiones: en el tiempo, primero, y en un espacio de características, segundo. Esto será una diferencia con el ejemplo desarrollado arriba donde procedía solamente en la dimensión espacial (de la imagen).

2.4. Elementos de procesamiento de imágenes

Veremos en el capítulo 4 que puede ser útil combinar, en el esquema de predicción, información espacial con información de apariencia. En esta sección, describimos de manera muy breve cómo esta información de apariencia está extraída de las imágenes.

2.4.1. Detección de poses humanas en imágenes

Un elemento importante en la apariencia de personas observadas en imágenes es su postura, i.e., la configuración de sus articulaciones. Intuitivamente, esta postura informa sobre la acción en curso, o la intención de la persona. Típicamente, se infiere a través de un modelo de detección de puntos claves del esqueleto humano.

El modelo que se usó en este trabajo es el de Cao et al. [6]. Le hicimos algunos ajustes, ya que este sistema originalmente recibe una imagen y regresa los puntos claves de todas las personas en la imagen. No obstante, no siempre regresa todos los puntos claves para cada una de las personas en la imagen. En nuestro caso, es importante contar con todos los puntos del esqueleto de la persona. Por ende, lo modificamos para realizar la detección en la caja delimitadora de la persona (esa información por lo general hace parte del Ground Truth que tenemos, como descrito todo al inicio de este capítulo).

Como lo ilustramos en la Fig. 2.13, los puntos claves que este sistema detecta son: Nariz, Cuello, Hombro derecho, Muñeca derecha, Hombro izquierdo, Codo izquierdo, Muñeca izquierda, Cadera derecha, Rodilla derecha, Tobillo derecho, Cadera izquierda, Rodilla izquierda, Tobillo izquierdo, Ojo derecho, Ojo izquierdo, Oreja derecha y Oreja izquierda.

Después de obtener todos los puntos claves con el sistema aplicado a la caja delimitadora, se les aplica una transformación para que las coordenadas estén expresadas en el marco de la imagen.

Breve descripción del método

Para nuestro uso, al sistema se le proporciona como entrada cada uno de los cuadros delimitadores del histórico de la persona, de tamaño $w \times h$, y nos regresa como salida una matriz $\mathbf{p}$ con $J = 18$ tripletas (x, y, π) para cada cuadro delimitador:

Figura 2.13: Puntos claves del esqueleto humano.

$$\mathbf{p} = \begin{pmatrix} x_1 & y_1 & \pi_1 \\ x_2 & y_2 & \pi_2 \\ \dots & \dots & \dots \\ x_{18} & y_{18} & \pi_{18} \end{pmatrix}. \tag{2.24}$$

Las primeras dos coordenadas x, y representan las coordenadas del punto clave del esqueleto en coordenadas píxel y π es un indicador de confianza sobre dicho punto clave.

Primero, una red neuronal predice simultáneamente un conjunto $\mathcal{S}$ de mapas de confianza $2D$. Esos mapas indican la probabilidad de ubicación en cada lugar de la parte del cuerpo $j \in [1, J]$ (puntos clave). Predice igualmente un conjunto $\mathcal{L}$ de campos vectoriales de afinidad de partes $2D$, que codifica el grado de asociación entre partes.

El conjunto $\mathcal{S}$ tiene J mapas de confianza, uno por cada punto clave del cuerpo, donde $\mathcal{S}_j \in \mathbb{R}^{w \times h}, j \in \{1, ..., J\}$. El conjunto $\mathcal{L}$ tiene C campos vectoriales $\mathcal{L}_c$, uno por miembro (antebrazo, etc.), donde $\mathcal{L}_c \in \mathbb{R}^{w \times h \times 2}, c \in \{1, ..., C\}$. En cada píxel de imagen en $\mathcal{L}_c$, se codifica un vector $2D$ el cual representa la posición y orientación de los miembros. Finalmente, los mapas de confianza y los campos de afinidad son analizados por inferencia glotona, para generar los puntos clave para todas las personas en la imagen.

El modelo fue entrenado con el conjunto de datos MSCOCO.

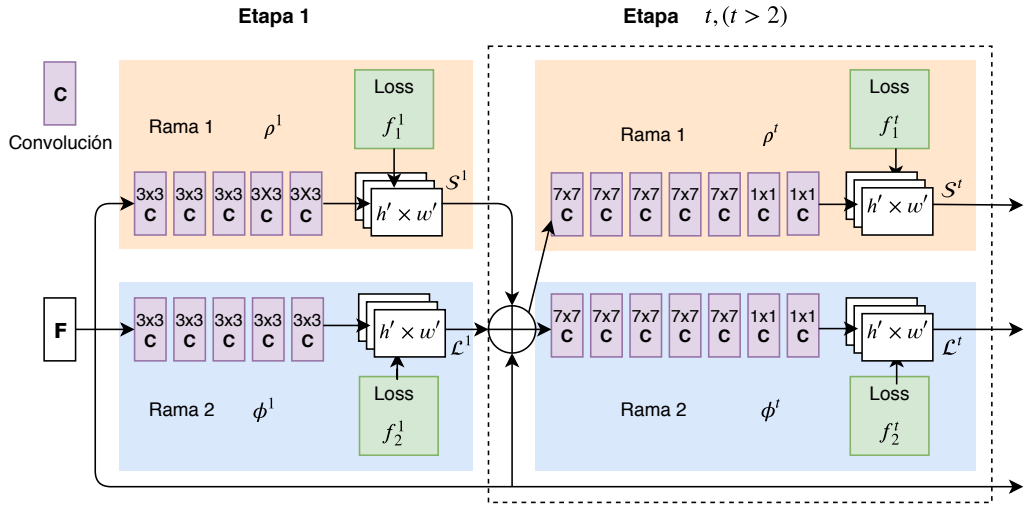


Figura 2.14: Arquitectura de la red CNN multi-etapa de dos ramas [6].

Detección y asociación simultaneas

La arquitectura de la red se ilustra en la Fig.2.14. Ésta predice simultáneamente los mapas de confianza y los campos de afinidad, que codifican la asociación de un miembro con otro. La red está dividida en dos ramas: la rama superior predice los mapas de confianza $\mathcal{S}_j$ y la rama inferior predice los campos de afinidad $\mathcal{L}_c$. Cada rama es una arquitectura de predicción iterativa que refina las predicciones en etapas sucesivas, $t \in \{1, ..., T\}$, con supervisión en cada etapa.

La imagen de entrada es primero analizada con las primeras capas de la red VGG-19 que se ilustra en la Fig. 2.15, la cual es descrita en [24]. Estas capas generan como salida un conjunto de F mapas de características, que dan la entrada a la primera etapa de cada rama.

2.5. Conjuntos de datos de trayectorias

En esta sección, presentamos brevemente los conjuntos de datos de trayectorias peatonales con los cuales se evaluarán nuestros modelos.

Los detalles de estos conjuntos de datos son resumidos en la Tabla 2.1.

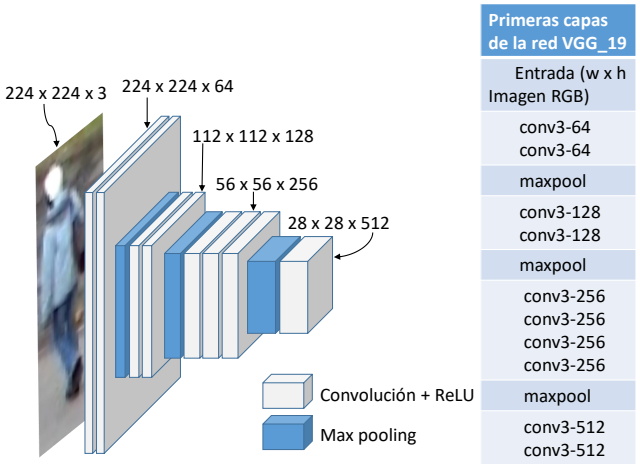


Figura 2.15: Primeras capas de la red VGG-19 [24].

Nombre	Ref.	Resolución	# Peatones	Framerate	ROI
Crowds Zara01	[15]	720 × 576	148	2.5	
Crowds Zara02	[15]	720 × 576	204	2.5	
PETS2009 S2L1	[8]	768 × 576	19	2.5	✓
UCY Univ	[15]	720 × 576	434	2.5	
ETH Hotel	[20]	720 × 576	390	2.5	
ETH Univ	[20]	640 × 480	360	2.5	
TownCentre	[3]	1080 × 1920	230	2.5	✓
ActEV	[2]	1080 × 1920		2.5	✓

Tabla 2.1: Detalles de los conjuntos de datos de trayectorias de peatones.

A continuación, se describe brevemente la naturaleza de cada conjunto de datos y en las imágenes (Figs. 2.16, 2.17, 2.18), se dan algunos ejemplos de los frames de algunos de estos conjuntos.

El conjunto de datos PETS2009 que se describe en [8], consta de tres conjuntos, que fueron registrados para un taller en el Campus Whiteknights de la universidad de Reading, Reino Unido. Los conjuntos de datos comprenden secuencias multisensoriales con escenarios de multitudes, con una creciente complejidad de la escena. El conjunto de datos S1 se enfoca al recuento de personas y la estimación de densidad. El conjunto de datos S2 aborda el seguimiento de personas. El conjunto de datos S3 implica análisis de flujo y reconocimiento de eventos. Para nuestra aplicación, usamos el subconjunto de datos L1 del conjunto S2, en el cual las personas están caminando y la densidad es menor que en los subconjuntos L2 y L3.

El conjunto de datos de peatones caminando ETH(EWAP) [20] se divide en dos conjuntos (ETH y Hotel). Las escenas fueron tomadas con vista panorámica en la entrada de un edificio.

Otro de los conjunto de datos es Crowds UCY [15], el cual contiene tres escenas desde una vista oblicua. La primera (Zara) muestra una parte de una calle comercial, la segunda (Univ) muestra una parte de un Campus y la tercera escena (Arxiepiskopi) muestra otra parte del Campus.

El conjunto de datos de Oxford Town Centre [3] es un vídeo de CCTV de peatones en un área concurrida del centro de Oxford que se utiliza para investigación y desarrollo de sistemas de reconocimiento facial y de actividad. El vídeo de CCTV se obtuvo de una cámara de vigilancia en la esquina de Cornmarket y Market St. en Oxford, Inglaterra.

El conjunto de datos ActEV [2], es un conjunto de datos publico, el cual fue realizado por el NIST en 2018 para la investigación de detección de actividades en vídeos (`http://actev.nist.gov/`). Este conjunto de datos es una versión mejorada de VIRAT [18], con más vídeos y anotaciones. Incluye 455 vídeos, capturadas a 30 frames por segundo, de 12 escenas, lo que totaliza más de 12 horas de grabaciones. La mayoría de los vídeos tienen una alta resolución de 1920×1080. Usamos el conjunto de entrenamiento oficial para el entrenamiento y el conjunto de prueba oficial para las pruebas.

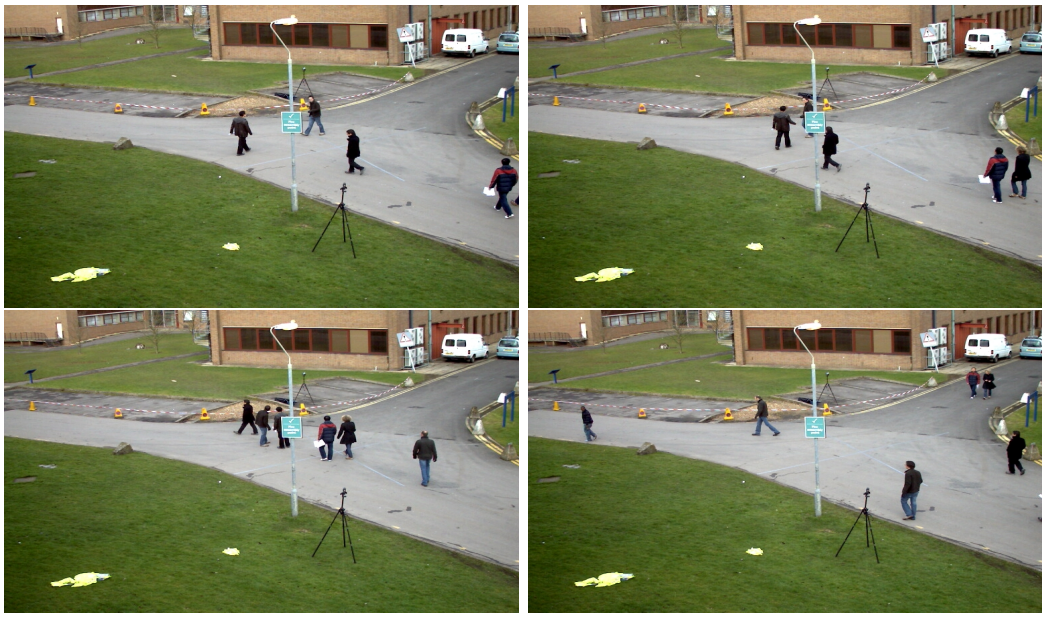


Figura 2.16: Ejemplos de frames del conjunto de datos PETS2009 S2L1.

Figura 2.17: Ejemplos de frames del conjunto de datos ETH Hotel/Univ.

Figura 2.18: Ejemplos de frames del conjunto de datos Crowds Zara01/Zara02.

2.6. Esquemas de entrenamiento

Para proceder al entrenamiento de los modelos con los diferentes datasets que hemos mencionado, procederemos de dos formas. El primer enfoque es el leave-one-out y el segundo individual, los cuales explicamos a continuación.

Leave-one-out (LOO): En este esquema, si tenemos n conjuntos de datos en total, entrenamos y validamos nuestro modelo con $n-1$ conjuntos y probamos en el conjunto restante. Repetimos esto para todos los n conjuntos. Si los datasets son suficientemente distintos, podemos de esa forma evaluar las capacidades de generalización de nuestro modelo.

Proporcional (PRO): Con este enfoque, los conjuntos de datos con los que se cuenten se dividen cada uno en conjunto de entrenamiento (70 %), validación (10 %) y prueba (20 %). Se unen todos los conjuntos de entrenamientos para así formar el conjunto de entrenamiento total, y así se procede con los otros dos tipos de conjuntos.

2.7. Métricas de evaluación

Para medir cuantitativamente el error realizado en las trayectorias predichas y para comparar con otros modelos, se utilizan dos métricas.

Error de Desplazamiento Promedio (ADE): Es la distancia promedio (entre índices i) entre las coordenadas verdaderas y coordenadas predichas por el sistema, sobre todos los instantes de tiempo de la ventana de predicción (índices t):

$$ADE = \frac{\sum_{i=1}^{N} \sum_{t=T_{obs}+1}^{T_{pred}} ||\hat{\mathbf{x}}_t^i - \mathbf{x}_t^i||_2}{N * T_{pred}}, \tag{2.25}$$

donde N es el número de datos (trayectorias) evaluados.

Error de Desplazamiento Final (FDE): Es la distancia Euclidiana entre las coordenadas predichas y las verdaderas coordenadas del último instante de tiempo predicho:

$$FDE = \frac{\sum_{i=1}^{N} ||\hat{\mathbf{x}}_{T_{pred}}^i - \mathbf{x}_{T_{pred}}^i||_2}{N}, \tag{2.26}$$

donde N es el número de peatones.

Capítulo 3

Predicción de trayectorias con redes recurrentes

En este capítulo, se describe una arquitectura básica de encodificación y decodificación de trayectorias para predicción, haciendo uso de sólo coordenadas. Esta arquitectura simple servirá de base para las arquitecturas descritas en los siguientes capítulos.

Con esta arquitectura, se comparan tres representaciones posibles de la salida de la red, para predecir trayectorias futuras de peatones y una representación tanto de la entrada como de la salida. Se muestra que el aprendizaje de la red *sí es sensible a la forma de representar la salida o la entrada.*

El resto de este capítulo está organizado de la siguiente manera. En la Sección 3.1, se describe la arquitectura general de la red que se usará para evaluar cada uno de los enfoques, mientras que en la Sección 3.2, se explica cómo se procesan los datos para realizar el entrenamiento. En la Sección 3.3, se presentan las diferentes representaciones de la salida que hemos evaluado; después, en la Sección 3.3.4, se describe una representación usada tanto para la entrada como para la salida de la red. La evaluación de cada una de éstas con un conjunto de datos se presenta en la Sección 3.4. En la Sección 3.5 se proporcionan algunas comparaciones con otros modelos de la literatura, y finalmente se extraen conclusiones de este capítulo en la Sección 3.6.

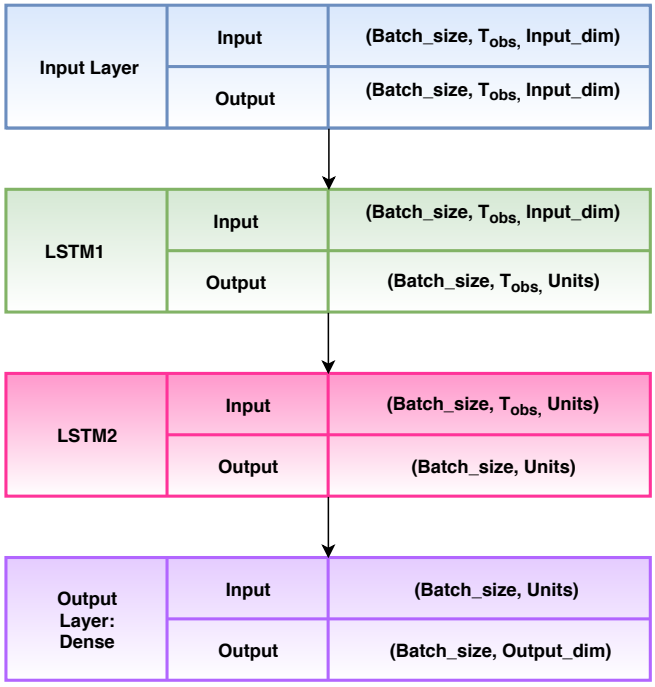


Figura 3.1: Esquema de la red neuronal básica usada para la predicción de trayectoria. Las dos capas intermedias son redes recurrentes de tipo LSTM: LSTM1 y LSTM2.

3.1. Arquitecturas de predicción

La arquitectura de las redes descritas en este capítulo constan de capas LSTM apiladas, similares a otras propuestas de la literatura [1, 10, 14]. La ilustramos en la Figura 3.1 y la describimos más a detalles a continuación.

3.1.1. Descripción de la red

Recordamos que este esquema de red neuronal permite predecir la trayectoria futura de un peatón i para $i \in \{1, ..., N\}$: dada una entrada $\mathbf{x}^i_{1:T_{obs}}$, se trata de dar un estimado de $\mathbf{x}^i_{T_{obs}+1:T_{pred}}$.

La primera capa es la de **entrada**, la cual se encarga de recibir $\mathbf{x}^i_{1:T_{obs}}$ que es la secuencia de las T_{obs} posiciones observadas, en coordenadas absolutas. Los datos de entrada se pueden representar como un tensor $N \times T_{obs} \times 2$, donde N es el número

de datos (típicamente, el tamaño de un *batch*) y el 2 representa la dimensión de la entrada (posiciones x y y).

Las capas LSTM1 y LSTM2 son las que actúan como **encoder** (es decir, de proyección a un espacio de dimensión más chica), y se encargan de codificar las secuencias observadas en vectores de tamaño fijo, llamados estados ocultos. La dimensión de este estado oculto (*units*) es el número de unidades de la celda LSTM. La salida de la capa LSTM1 se puede representar como un tensor $N \times T_{obs} \times units$ (la serie de estados ocultos). La tercera capa LSTM2 regresa solamente su último estado oculto $\tilde{\mathbf{h}}_{T_{pred}}$ (usamos aquí el tilde para referirnos al estado oculto de esta segunda LSTM), mientras que recibe como entrada la secuencia de los estados ocultos construidos a partir de la información de la secuencia observada, es decir una de las salidas de la LSTM1.

Los vectores de estado oculto $\mathbf{h}_t$ tienen, en general, más presente la información del último paso analizado. Por ejemplo, si la sucesión observada es de longitud T_{obs}, el estado oculto $\mathbf{h}_1$ tiene la información de la observación $\mathbf{x}_1^i$, mientras que el estado oculto $\mathbf{h}_2$ tiene más presente la información de la posición $\mathbf{x}_2^i$ y menos la de $\mathbf{x}_1^i$. Por lo tanto, el último estado oculto $\mathbf{h}_{T_{obs}}$ que sale de esta capa tiene la información de cada una de las posiciones observadas pero conserva con mayor precisión la de $t = T_{obs}$.

La cuarta capa es una densa, la cual funciona como **decoder**. Esta capa recibe como entrada el último estado oculto de la capa LSTM2, a la cual se le aplica una capa densa de tamaño 2, para calcular las coordenadas x y y de la representación elegida para la salida (veremos en la Sección 3.3 que podemos definirla de diferentes maneras).

En nuestros experimentos, esta red es optimizada con el optimizador RMSprop y entrenada con los siguientes hiperparámetros: 250 épocas, batches de tamaño $N = 64$, función de pérdida logcosh vista en la Ec. 2.10, tasa de aprendizaje (*learning rate*) inicial de 0.01. Hemos usado $units = 9$.

3.1.2. Aplicación recursiva

Con la red descrita arriba, obtenemos típicamente un mapeo de la trayectoria observada hacia la posición siguiente:

$$\mathbf{x}^i_{1:T_{obs}} \rightarrow \hat{\mathbf{x}}^i_{T_{obs}+1}. \tag{3.1}$$

Ahora, como en casi todas las arquitecturas de este estilo, se puede aplicar consecutivamente para generar varias posiciones futuras. La posición *predicha* $\hat{\mathbf{x}}^i_{T_{obs}+1}$ se incorpora a la secuencia $\mathbf{x}^i_{2:T_{obs}}$ (olvidándonos de la posición $\mathbf{x}^i_1$) y podemos aplicar la *misma red*

$$\mathbf{x}^i_{2:T_{obs}}, \hat{\mathbf{x}}^i_{T_{obs}+1} \rightarrow \hat{\mathbf{x}}^i_{T_{obs}+2}. \tag{3.2}$$

Sucesivamente, iterando el mecanismo anterior, podemos reconstruir de esta manera una secuencia completa predicha.

3.2. Procesamiento de los datos

En esta sección, se explicará brevemente cómo se procesan los datos que se usan para el entrenamiento del modelo presentado en la Fig. 3.1. En dicho modelo, se predice la trayectoria de un peatón de manera independiente a los otros, es decir que no tomamos en cuenta la información de los demás peatones que se encuentren en su entorno. Por lo tanto, al momento de procesar los datos (*trayectorias en marco píxel*), se obtienen las trayectorias $\mathbf{x}^i$ donde el peatón $i \in \{1, ..., N\}$. Posteriormente, dividimos el conjunto de todos los peatones presentes en el conjunto de datos, en conjunto de entrenamiento y conjunto de prueba.

Aquellas trayectorias $\mathbf{x}^i$ del conjunto de entrenamiento, que cumplan que su longitud es mayor o igual a $T_{obs} + 1$, las subdividimos en pedazos de longitud $T_{obs} + 1$, donde $\mathbf{x}^i_{1:T_{obs}}$ representa la entrada de la red y $\mathbf{x}^i_{T_{obs}+1}$ representa la salida deseada de la red.

3.3. Definición de la representación de salida

En esta sección, presentaremos diferentes enfoques para representar la salida de nuestra red así como una representación que es tanto para la entrada y salida de la red. Cada una de estas representaciones nos es útil para predecir la trayectoria futura de peatones. En la mayoría de los trabajos revisados, los sistemas de predicción usan una representación en coordenadas absolutas de la información de las posiciones (Subsección 3.3.1), pero veremos al menos dos alternativas.

3.3.1. Coordenadas Absolutas

En este caso, el modelo recibe sucesiones $\mathbf{x}^i_{1:T_{obs}}, i \in \{1, ..., N\}$ y para el entrenamiento, recibe la información de referencia como salida $\mathbf{x}^i_{T_{obs}+1}$, el cual es el siguiente paso de la trayectoria i después de los T_{obs} observados, en *coordenadas absolutas*, así que la red aprende cómo calcular la siguiente posición de manera absoluta, al proporcionarle la parte observada.

Una intuición que queremos comprobar a continuación es que, para hacer correctamente la regresión, la red necesitaría aprender "menos" que eso, es decir principalmente desviaciones (con respecto a la última posición observada, por ejemplo).

3.3.2. Desplazamientos

En este segundo caso, el modelo recibe de entrada sucesiones $\mathbf{x}^i_{1:T_{obs}}$ y, para su entrenamiento, para comprobar la intuición que enunciamos arriba, le proporcionamos como salida $\mathbf{d}^i_{T_{obs}+1}$, el cual es la diferencia entre las coordenadas $\mathbf{x}^i_{T_{obs}+1}$ y $\mathbf{x}^i_{T_{obs}}$ (es decir, el *desplazamiento* observado entre los tiempos T_{obs} y $T_{obs}+1$).

Por lo tanto, para predecir la posición de la persona i, en el instante de tiempo $T_{obs}+1$, se le proporciona a la red una sucesión de la parte observada $\mathbf{x}^i_{1:T_{obs}}$ y la red proporcionará como salida $\hat{\mathbf{d}}_{T_{obs}+1}$, por lo que la posición predicha de la persona i en $t = T_{obs}+1$ es:

$$\hat{\mathbf{x}}_{T_{obs}+1} = \mathbf{x}_{T_{obs}} + \hat{\mathbf{d}}_{T_{obs}+1}. \tag{3.3}$$

3.3.3. Variación al modelo de regresión lineal

En esta representación, lo que queremos que la red aprenda es a evaluar una desviación entre la posición verdadera y la posición predicha haciendo uso de la *regresión lineal en las coordenadas observadas x e y*. La motivación es que el modelo más simple entre todos es el modelo lineal y que este modelo en sí puede predecir razonablemente bien en muchos casos. Aprendemos entonces a predecir el **residuo** con respecto a este modelo.

A continuación se explica a detalle como se calcula cada una de las regresiones para las dos variables, haciendo uso de datos observados.

Suponemos que x e y son variables que dependen linealmente del tiempo t, es decir $x(t) = x_0 + v_x t,\ y(t) = y_0 + v_y t$ y suponemos que tenemos como observaciones los conjuntos $\{x_1, ..., x_{T_{obs}}\}$ y $\{y_1, ..., y_{T_{obs}}\}$.

Para poder predecir la posición $\mathbf{x}_t = (x_t, y_t)^T$, para el instante de tiempo $t > T_{obs}$, podemos ajustar un modelo lineal en las dos variables. Por lo tanto, habría que minimizar la función 3.4 con mínimos cuadrados:

$$s = \sum_{i=1}^{T_{obs}} (x_i - (x_0 + v_x t_i))^2 + (y_i - (y_0 + v_y t_i))^2. \tag{3.4}$$

Para minimizar la función anterior, se calcula las derivadas parciales de s. Calculamos la derivada parcial de la función s con respecto a x_0 y la igualamos a 0:

$$\frac{\partial s}{\partial x_0} = -2 \sum_{i=1}^{T_{obs}} (x_i - v_x t_i - x_0) = 0 \tag{3.5}$$

por lo que

$$-\sum_{i=1}^{T_{obs}} x_i + v_x \sum_{i=1}^{T_{obs}} t_i + \sum_{i=1}^{T_{obs}} x_0 = 0 \tag{3.6}$$

y finalmente

$$\sum_{i=1}^{T_{obs}} x_i = T_{obs} x_0 + v_x \sum_{i=1}^{T_{obs}} t_i. \tag{3.7}$$

Dividiendo todo entre el número de observaciones, tenemos:

$$\bar{x} = x_0 + v_x\bar{t}, \tag{3.8}$$

y despejando x_0

$$x_0 = \bar{x} - v_x\bar{t}, \tag{3.9}$$

donde hicimos aparecer los valores promedios $\bar{x}$ y $\bar{t}$. Calculando la derivada parcial de la función s con respecto a v_x, obtenemos

$$\frac{\partial s}{\partial v_x} = \sum_{i=1}^{T_{obs}} -2t_i(x_i - v_x t_i - x_0) = 0, \tag{3.10}$$

y simplificando

$$-\sum_{i=1}^{T_{obs}} t_i x_i + \sum_{i=1}^{T_{obs}} v_x t_i^2 + \sum_{i=1}^{T_{obs}} t_i x_0 = 0. \tag{3.11}$$

Despejando

$$\sum_{i=1}^{T_{obs}} t_i x_i = v_x \sum_{i=1}^{T_{obs}} t_i^2 + x_0 \sum_{i=1}^{T_{obs}} t_i \tag{3.12}$$

y sustituyendo la ecuación 3.9, en la ecuación. 3.12, se tiene

$$\sum_{i=1}^{T_{obs}} t_i x_i = (\bar{x} - v_x\bar{t}) \sum_{i=1}^{T_{obs}} t_i + v_x \sum_{i=1}^{T_{obs}} t_i^2 \tag{3.13}$$

despejando

$$\sum_{i=1}^{T_{obs}} t_i x_i - \bar{x} \sum_{i=1}^{T_{obs}} t_i = v_x \Big(\sum_{i=1}^{T_{obs}} t_i^2 - \bar{t} \sum_{i=1}^{T_{obs}} t_i \Big). \tag{3.14}$$

Por otro lado tenemos:

$$\begin{aligned}\sum_{i=1}^{T_{obs}}(x_i-\bar{x})(t_i-\bar{t}\,) &= \sum_{i=1}^{T_obs}(x_it_i - x_i\bar{t} - \bar{x}t_i + \bar{x}\bar{t}\,)\\ &= \sum_{i=1}^{T_{obs}} x_it_i - T_{obs}\bar{t}\bar{x} - \bar{x}\sum_{i=1}^{T_{obs}} t_i + T_{obs}\bar{x}\bar{t} \qquad (3.15)\\ &= \sum_{i=1}^{T_{obs}} x_it_i - \bar{x}\sum_{i=1}^{T_{obs}} t_i\end{aligned}$$

y

$$\begin{aligned}\sum_{i=1}^{T_{obs}}(t_i-\bar{t}\,)^2 &= \sum_{i=1}^{T_{obs}}(t_i^2 - 2t_i\bar{t} + \bar{t}^2)\\ &= \sum_{i=1}^{T_{obs}} t_i^2 - 2\bar{t}\sum_{i=1}^{T_{obs}} t_i + T_{obs}\bar{t}^2 \qquad (3.16)\\ &= \sum_{i=1}^{T_{obs}} t_i^2 - \bar{t}\sum_{i=1}^{T_{obs}} t_i.\end{aligned}$$

Sustituyendo la ecuación 3.15 y ecuación 3.16 en 3.14

$$\sum_{i=1}^{T_{obs}}(x_i-\bar{x})(t_i-\bar{t}\,) = v_x\sum_{i=1}^{T_{obs}}(t_i-\bar{t}\,)^2. \qquad (3.17)$$

Despejando v_x, se obtiene finalmente:

$$v_x = \frac{\sum_{i=1}^{T_{obs}}(x_i-\bar{x})(t_i-\bar{t}\,)}{\sum_{i=1}^{T_{obs}}(t_i-\bar{t}\,)^2} \qquad (3.18)$$

Entonces para poder calcular las siguientes posiciones de la secuencia con este modelo simple ajustado sobre los datos observados $\mathbf{x}^i_{1:T_{obs}}$, lo hacemos con las siguientes formulas:

$$
\begin{aligned}
x(t) &= x_0 + v_x t & & & y(t) &= y_0 + v_y t \\
v_x &= \frac{\sum_{i=1}^{T_{obs}} (x_i - \bar{x})(t_i - \bar{t})}{\sum_{i=1}^{T_{obs}} (t_i - \bar{t})^2} & \text{y} & & v_y &= \frac{\sum_{i=1}^{T_{obs}} (y_i - \bar{y})(t_i - \bar{t})}{\sum_{i=1}^{T_{obs}} (t_i - \bar{t})^2} \\
x_0 &= \bar{x} - v_x \bar{t} & & & y_0 &= \bar{y} - v_y \bar{t}.
\end{aligned}
\tag{3.19}
$$

Entrenamiento

Para el entrenamiento de la red, se le proporciona como entrada secuencias de posiciones absolutas $\mathbf{x}^i_{1:T_{obs}}$ de tamaño T_{obs}. Con dicha sucesión, se calculan las regresiones lineales de $x(t)$ y $y(t)$ con las formulas 3.19 descritas más arriba. Con las regresiones, se calcula en particular la posición para $t = T_{obs} + 1$, es decir $r(T_{obs} + 1) = (x(T_{obs} + 1), y(T_{obs} + 1))$. De salida deseada, se proporciona $\mathbf{d}^r_{T_{obs}+1}$, el cual es la diferencia entre las coordenadas de $\mathbf{x}^i_{T_{obs}+1}$ y $r(T_{obs}+1)$. La red proporciona como salida $\hat{\mathbf{d}}^r_{T_{obs}+1}$, por lo que la posición predicha de la persona i en $t = T_{obs} + 1$ se deduce por:

$$\hat{\mathbf{x}}_{T_{obs}+1} = r(T_{obs} + 1) + \hat{\mathbf{d}}^r_{T_{obs}+1}. \tag{3.20}$$

3.3.4. Representación por desplazamientos en entrada/salida

En esta subsección explicaremos brevemente cómo usar la información de coordenadas por medio de desplazamientos, para representar tanto los datos de entrada y salida con la arquitectura ilustrada en la Fig. 3.1.

Para esta propuesta alternativa, el modelo recibe como entrada sucesiones de la forma $\mathbf{d}^i_{1:T_{obs}}$, donde $\mathbf{d}^i_{2:T_{obs}}$ es la diferencia entre coordenadas $\mathbf{x}^i_{2:T_{obs}}$ y $\mathbf{x}^i_{1:T_{obs}-1}$. Usamos $\mathbf{d}^i_1 = [0, 0]$, porque se considera que en el primer instante de tiempo el peatón aún no se ha movido.

Como salida deseada, usamos $\mathbf{d}^i_{T_{obs}+1} = \mathbf{x}^i_{T_{obs}+1} - \mathbf{x}^i_{T_{obs}}$, como explicado en la sección anterior.

Para predecir la posición de la persona i, en el instante de tiempo $T_{obs} + 1$, a la red se le proporciona la sucesión $\mathbf{d}^i_{1:T_{obs}}$ y la red proporciona como salida $\hat{\mathbf{d}}_{T_{obs}+1}$. La

posición predicha del peatón i en el instante de tiempo $t = T_{obs} + 1$ es:

$$\hat{\mathbf{x}}_{T_{obs}+1} = \mathbf{x}_{T_{obs}} + \hat{\mathbf{d}}_{T_{obs}+1}. \tag{3.21}$$

3.4. Evaluación de las representaciones

Se evaluarán las tres representaciones de salida y la representación de entrada/salida con desplazamientos presentadas anteriormente. Para eso, usamos el conjunto de datos PETS2009-S2L1, con diferentes frecuencias de muestreo temporal.

3.4.1. Ajuste de los hiperparámetros

Para ajustar los hiperparámetros (épocas, *batch size, hidden state size* y *learning rate*), para los modelos con las diferentes representaciones de salida, se siguió un procedimiento estándar [4]. Lo primero que se hace es ajustar el número de *épocas*, y esto se realiza como se explica a continuación y, para mayor claridad, en el Algorithm 1.

Se divide el conjunto de datos en conjunto de entrenamiento y prueba. Posteriormente, definimos la lista de valores de épocas que vamos a evaluar, mientras que los demás hiperparámetros los iniciamos en valores pequeños o en los valores por defecto. A continuación realizamos 30 repeticiones de pasos de entrenamiento del modelo por cada valor de la lista de épocas, para evaluar cada repetición. Al finalizar el entrenamiento, evaluamos el modelo con el conjunto de prueba. Cada evaluación se refleja en el valor de una métrica (*MSE*), midiendo cómo fue el entrenamiento.

Por lo que para cada valor de *época*, tenemos 30 valores de *MSE*, que son distintos porque los parámetros del modelo se inicializan aleatoriamente, lo que nos permite realizar un boxplot. Análogamente, se realiza este proceso con los otros hiperparámetros, ahora fijando el valor del hiperparámetro que haya resultado mejor del experimento antes hecho. En las imágenes (Figs. 3.2, 3.3, 3.4, 3.5, 3.6 y 3.7) se muestran los resultados de los experimentos para cada una de las diferentes representaciones de salida con un boxplot. Los experimentos se hicieron con dos valores de framerate, 3.75fps y 7.5fps.

Algorithm 1 Búsqueda de hiperparámetros

```
repeats = 30    \\Repeticiones del entrenamiento
b = 8                          \\Tamaño del bache
h = 1   \\Dimensión del estado oculto de la LSTM
lr = default                      \\Learning rate
epochs = [100,200,250,300]

for e in epochs:
    results[str(e)] = experiment(repeats,train,test,e,b,h,lr)

\\Se calcula el boxplot con los resultados de MSE obtenidos de
las 30 repeticiones de cada uno de los valores de las épocas.

\\Se toma el mejor valor de época de acuerdo al valor de MSE,
representado en los gráficos del boxplot.

\\Se fija el valor de época y se repite el experimento pero
variando sobre el siguiente parámetro (tamaño de bache), y así
sucesivamente hasta acabar con los 4 parámetros.
```

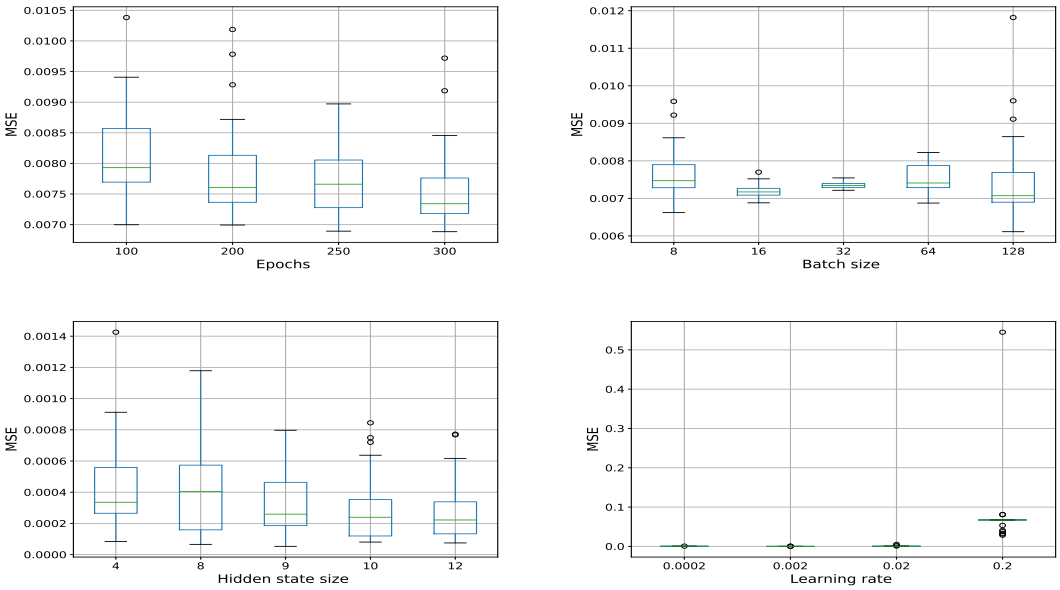


Figura 3.2: MSE sobre el conjunto de prueba para diferentes valores de parámetros, para el modelo de coordenadas absolutas, con un framerate de 7.5 frames por segundo.

Parámetros para el modelo coordenadas absolutas De la Fig. 3.2, se puede observar que para este caso el número de épocas es un parámetro crítico porque se observa que, al crecer el número de épocas, la métrica (MSE) tiende a descender en la mayoría de los valores que éste toma. Mientras tanto, el tamaño de batch no muestra un comportamiento estable, por lo que podríamos tomarnos un tamaño default. Para la dimensión del estado oculto de la celda LSTM, se observa que, a partir del valor 9, el MSE decrece. Finalmente, en la tasa de aprendizaje no hay tanta variación. De la Fig. 3.3, se deduce que podemos tomar un valor de época entre 200 y 300, ya que el promedio de las cajas no varían tanto, por lo que el comportamiento del tamaño del bache sería similar con cualquier valor de época, además que el promedio de las cajas de los tamaños de bache no varían tanto.

Parámetros para el modelo de desplazamientos De la Fig. 3.4, se puede concluir que los parámetros que sí son importantes de ajustar son el tamaño del batch y la tasa de aprendizaje.

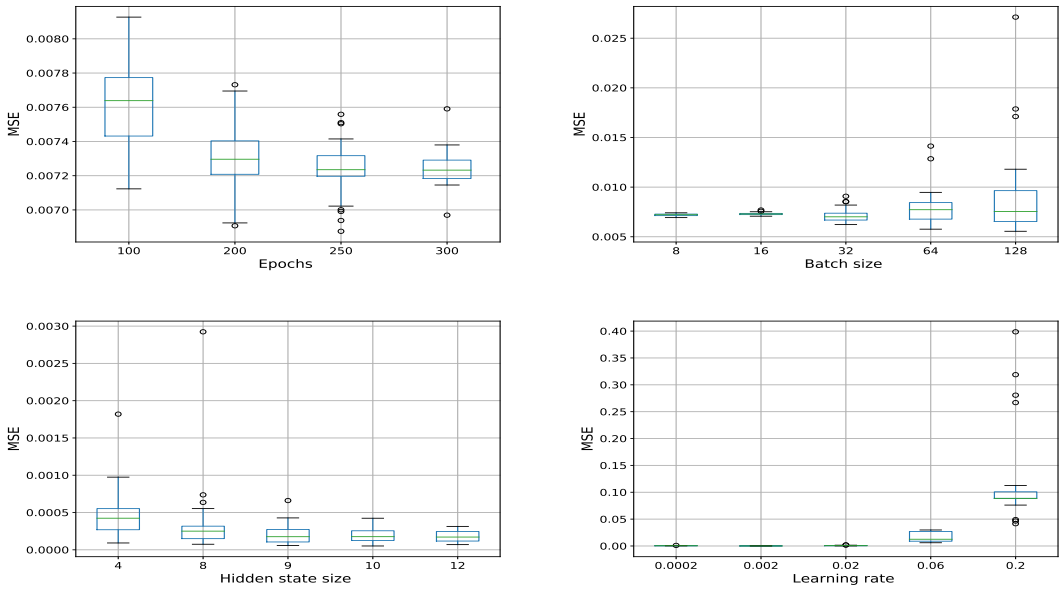


Figura 3.3: MSE sobre el conjunto de prueba para diferentes valores de parámetros, para el modelo de coordenadas absolutas, con un framerate de 3.75 frames por segundo.

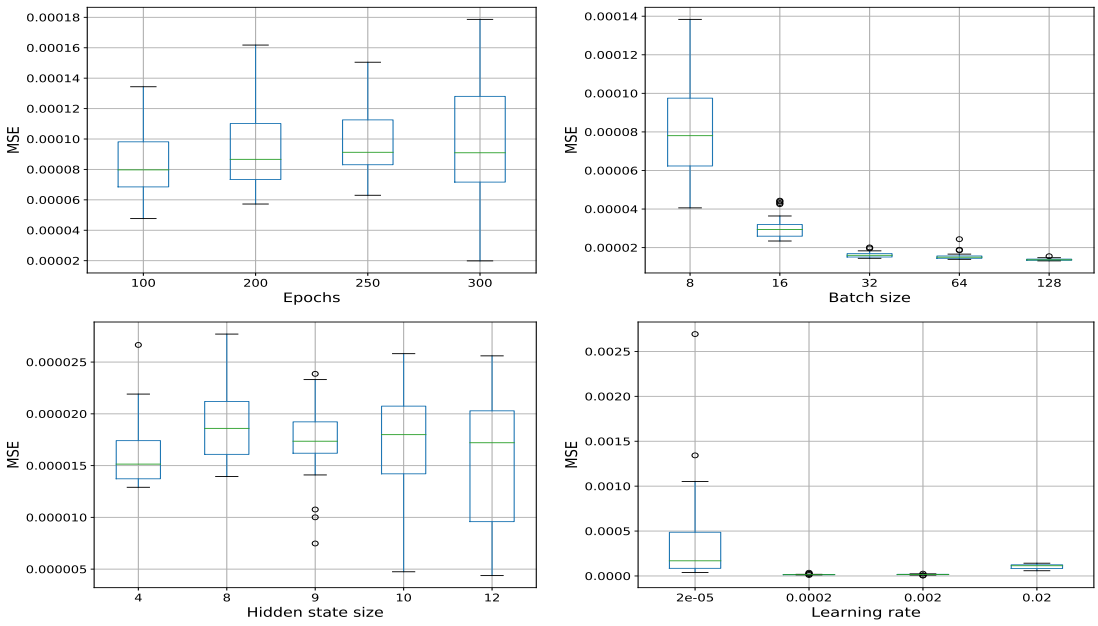


Figura 3.4: MSE sobre el conjunto de prueba para diferentes valores de los parámetros, para el modelo de desplazamientos, con un framerate de 7.5 frames por segundo.

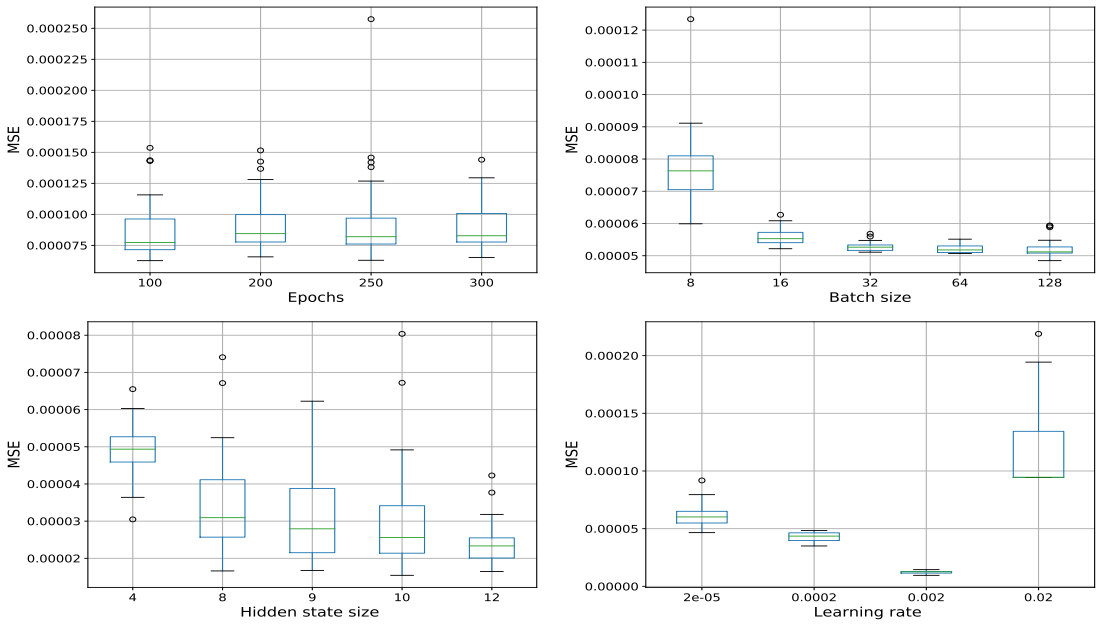


Figura 3.5: MSE sobre el conjunto de prueba para diferentes valores de parámetros, para el modelo de desplazamientos, con un framerate de 3.75 frames por segundo.

De la Fig. 3.5, se deduce que el número de épocas no es un parámetro que afecte de manera importante el entrenamiento (más allá de 100 épocas), pero el que sí es crítico, es el tamaño de batch.

Parámetros para el modelo de variación a regresión De la Fig. 3.6, se puede observar de la gráfica del número de épocas que el promedio de cada una de las cajas no varia tanto, por lo que el comportamiento del tamaño del lote sería similar con cualquier valor de épocas.

De la Fig. 3.7, se concluye que podemos tomar un valor entre 100 a 300 para el número de épocas, ya que tienen un comportamiento similar. El tamaño del batch es importante ajustarlo.

Como se puede observar, la búsqueda de los hiperparámetros es un trabajo difícil. Si realmente se quiere encontrar los mejores conjuntos de parámetros para ciertos modelos, en teoría tendríamos que probar todas las combinaciones posibles, lo cual es muy tardado. Si lo hacemos con el protocolo mencionado anteriormente, éste no nos asegura encontrar el mejor (es greedy).

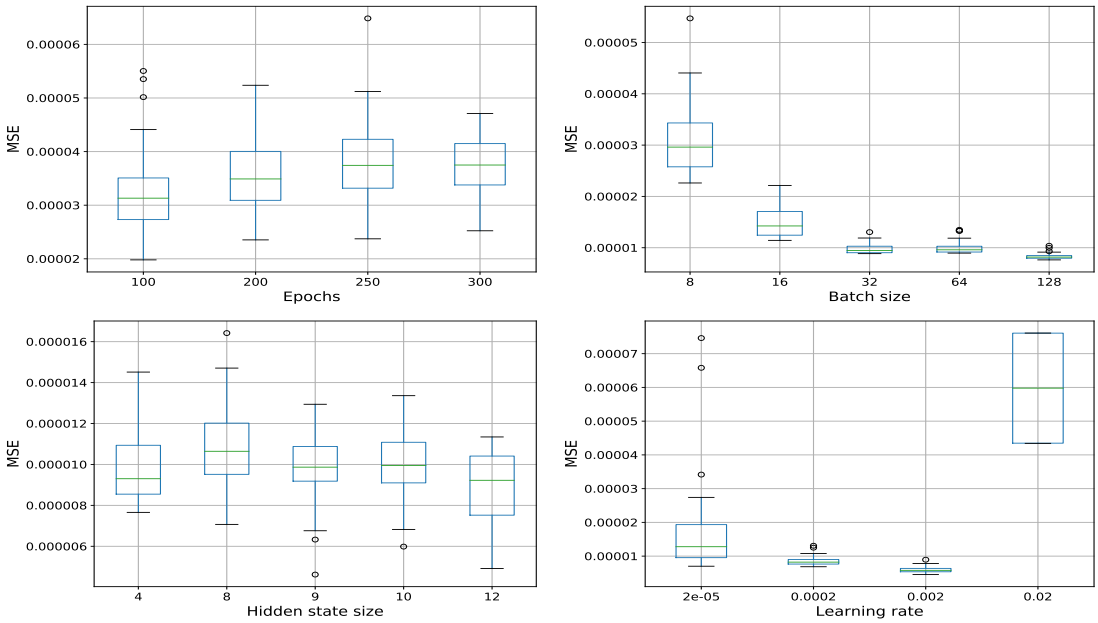


Figura 3.6: MSE sobre el conjunto de prueba para diferentes valores de parámetros, para el modelo de variación a regresión, con un framerate de 7.5 fps.

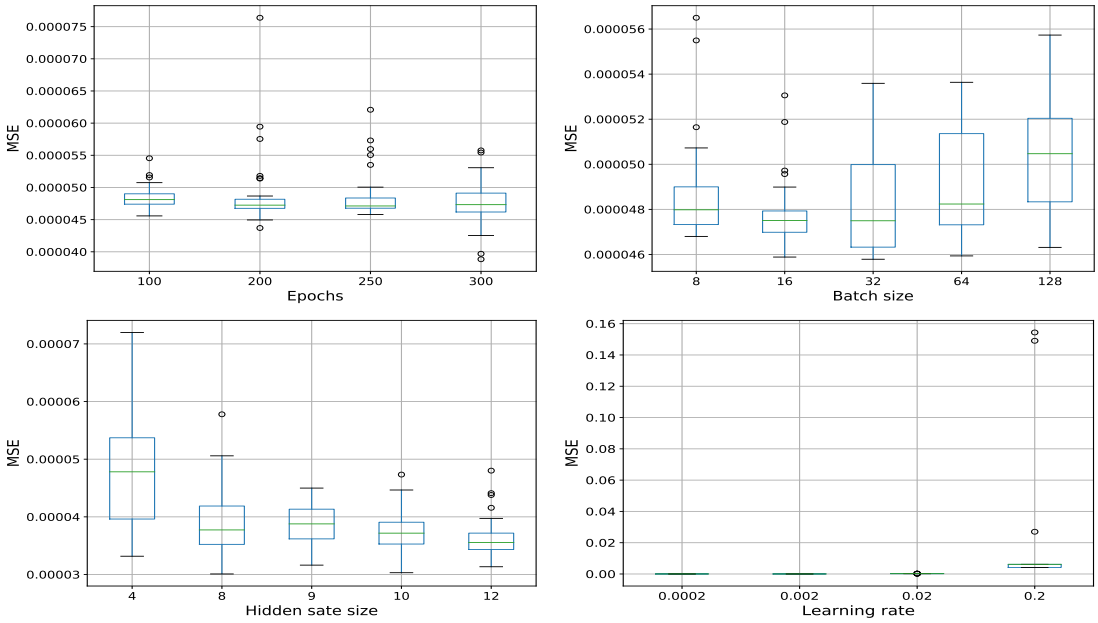


Figura 3.7: MSE sobre el conjunto de prueba para diferentes valores de parámetros, para el modelo de variación a regresión, con un framerate de 3.75 fps.

Finalmente, para cada uno de nuestros experimentos en este capítulo, utilizaremos los siguientes hiperparámetros: número de épocas 300, tamaño de batch 64, dimensión del estado oculto de la celda LSTM 9 y tasa de aprendizaje 0.01. Con esta combinación de valores hemos encontrado buenos resultados.

3.4.2. Evaluación Cuantitativa

Para evaluar las distintas representaciones descritas en las secciones anteriores, se toma como longitud de la secuencia observada $T_{obs} = 8$ posiciones y se predicen respectivamente 2, 4 y 8 posiciones futuras (con la aplicación *recursiva* de nuestra red). La unidad en las cuales se reportan las pruebas es píxeles (en esta base de datos, las trayectorias son expresadas en términos de coordenadas de la imagen).

La división de los datos se hizo de la siguiente manera. Como el conjunto de datos PETS2009-S2L1 cuenta con 19 peatones, entonces se dividió en 5 conjuntos los peatones. Luego, se entrenó la red con 4 de los 5 conjuntos y se probó con el sobrante.

Para analizar los resultados de los experimentos hay que mirar las tablas por pares. En las Tablas 3.1 y 3.2, se puede observar cada uno de los valores de ADE y FDE (ver Sección 2.7) al predecir 2, 4 y 8 posiciones futuras. Esos valores se evalúan para cada una de las diferentes representaciones (a las cuales nos referimos respectivamente por LSTM-CA, LSTM-D, LSTM-VR y LSTM-SD), con una frecuencia de muestreo de 7.5 frames por segundo, donde los valores en negritas representan los mejores resultados. En las Tablas 3.3 y 3.4, en cambio, las muestras se tomaron a 3.75 frames por segundo (es decir, la duración de la trayectoria es más larga, lo que hace que la predicción sea más difícil).

De las Tablas 3.1 y 3.2, se puede concluir que, cuando el muestreo es más frecuente (y que entonces la trayectoria es más corta) y la entrada de la red son sucesiones de coordenadas absolutas, el modelo de salida con variación a regresión da el mejor resultado en este caso, con una mejora del casi 55 % en ADE y un 46.40 % en FDE al predecir 4 pasos a futuro, mientras que mejora 48 % en ADE y 42.55 % FDE al predecir 8 posiciones futuras. Estas mejoras son calculadas con respecto a los promedios obtenidos del modelo LSTM-CA y del modelo LSTM-VR. Pero, de forma general, es mejor el modelo (LSTM-SD), el cual tiene como entrada y salida los

Error de Desplazamiento Promedio (**ADE**) /Error de Desplazamiento Final (**FDE**)						
Conjunto	LSTM-CA			LSTM-D		
de prueba	2PF	4PF	8PF	2PF	4PF	8PF
1	3.753/4.219	5.063/7.154	8.444/14.807	1.947/2.613	3.399/5.634	6.798/13.279
2	4.31/4.819	5.74/8.073	9.589/17.044	1.514/2.034	2.654/4.403	5.349/10.425
3	3.018/3.487	4.296/6.356	7.604/13.961	1.61/2.166	2.812/4.684	5.674/11.103
4	3.469/3.956	4.802/6.944	8.252/14.871	1.201/1.615	2.115/3.523	4.273/8.37
5	7.942/8.958	11.009/15.998	19.644/36.25	2.499/3.346	4.283/7.019	8.38/16.046
Promedio	4.498/5.088	6.182/8.905	10.707/19.387	1.754/2.355	3.053/5.053	6.095/11.845

Tabla 3.1: Comparación cuantitativa entre las diferentes representaciones para el conjunto de datos PETS2009-S2L1 (con una frecuencia alta de muestreo).

Error de Desplazamiento Promedio (**ADE**)/Error de Desplazamiento Final (**FDE**)						
Conjunto	LSTM-VR			LSTM-SD		
de prueba	2PF	4PF	8PF	2PF	4PF	8PF
1	2.189/2.64	3.307/5.12	6.617/13.293	**0.588/0.831**	**1.213/2.258**	**3.163/7.103**
2	1.491/1.797	2.242/3.471	4.46/8.956	**0.432/0.613**	**0.905/1.697**	**2.393/5.457**
3	1.830/2.205	2.766/4.285	5.569/11.251	**0.429/0.606**	**0.886/1.642**	**2.248/4.997**
4	1.238/1.493	1.841/2.841	3.563/7.064	**0.38/0.542**	**0.81/1.534**	**2.158/4.981**
5	2.497/3.003	3.768/5.827	7.523/15.119	**0.77/1.073**	**1.542/2.792**	**3.774/8.233**
Promedio	1.849/2.228	2.785/4.309	5.546/11.137	**0.52/0.73**	**1.071/2.324**	**2.747/6.154**

Tabla 3.2: Comparación cuantitativa entre las diferentes representaciones para el conjunto de datos PETS2009-S2L1 (con una frecuencia alta de muestreo).

Error de Desplazamiento Promedio (**ADE**)/Error de Desplazamiento Final (**FDE**)						
Conjunto	LSTM-CA			LSTM-D		
de prueba	2PF	4PF	8PF	2PF	4PF	8PF
1	5.408/6.603	8.741/13.951	17.718/34.16	4.833/6.492	8.462/14.076	17.515/34.55
2	5.285/6.149	7.753/11.703	14.458/27.032	2.599/3.521	4.733/8.087	**10.324/21.072**
3	8.386/9.869	12.478/19.03	23.43/44.841	1.821/2.513	3.546/6.299	8.266/17.623
4	5.169/6.035	7.452/11.157	12.998/23.355	**1**.877/2.578	3.689/6.61	8.936/**19.4**
5	8.952/10.871	13.913/21.86	27.041/52.371	5.891/7.949	10.548/17.791	22.192/44.125
Promedio	6.64/7.905	10.067/15.540	19.129/36.352	3.404/4.611	6.196/10.573	13.447/27.354

Tabla 3.3: Comparación cuantitativa entre las diferentes representaciones para el conjunto de datos PETS2009-S2L1 (con una baja frecuencia de muestreo).

Error de Desplazamiento Promedio (**ADE**)/Error de Desplazamiento Final (**FDE**)						
Conjunto	LSTM-VR			LSTM-SD		
de prueba	2PF	4PF	8PF	2PF	4PF	8PF
1	7.306/8.81	11.013/16.851	21.32/41.037	**1.543/2.338**	**3.898/7.868**	**11.84/28.413**
2	4.196/5.027	6.263/9.491	12.012/23.373	**1.112/1.715**	**3.113/6.541**	10.651/26.965
3	4.998/5.992	7.451/11.323	14.238/27.699	**0.877/1.327**	**2.168/4.32**	**6.196/14.267**
4	3.855/4.653	5.739/8.756	10.859/20.738	**1.015/1.558**	**2.726/5.657**	**8.882**/22.083
5	7.658/9.168	11.354/17.182	21.446/40.926	**1.999/2.965**	**4.831/9.544**	**14.241/33.955**
Promedio	5.603/6.73	8.364/12.721	15.975/30.755	**1.309/1.981**	**3.347/6.786**	**10.362/25.137**

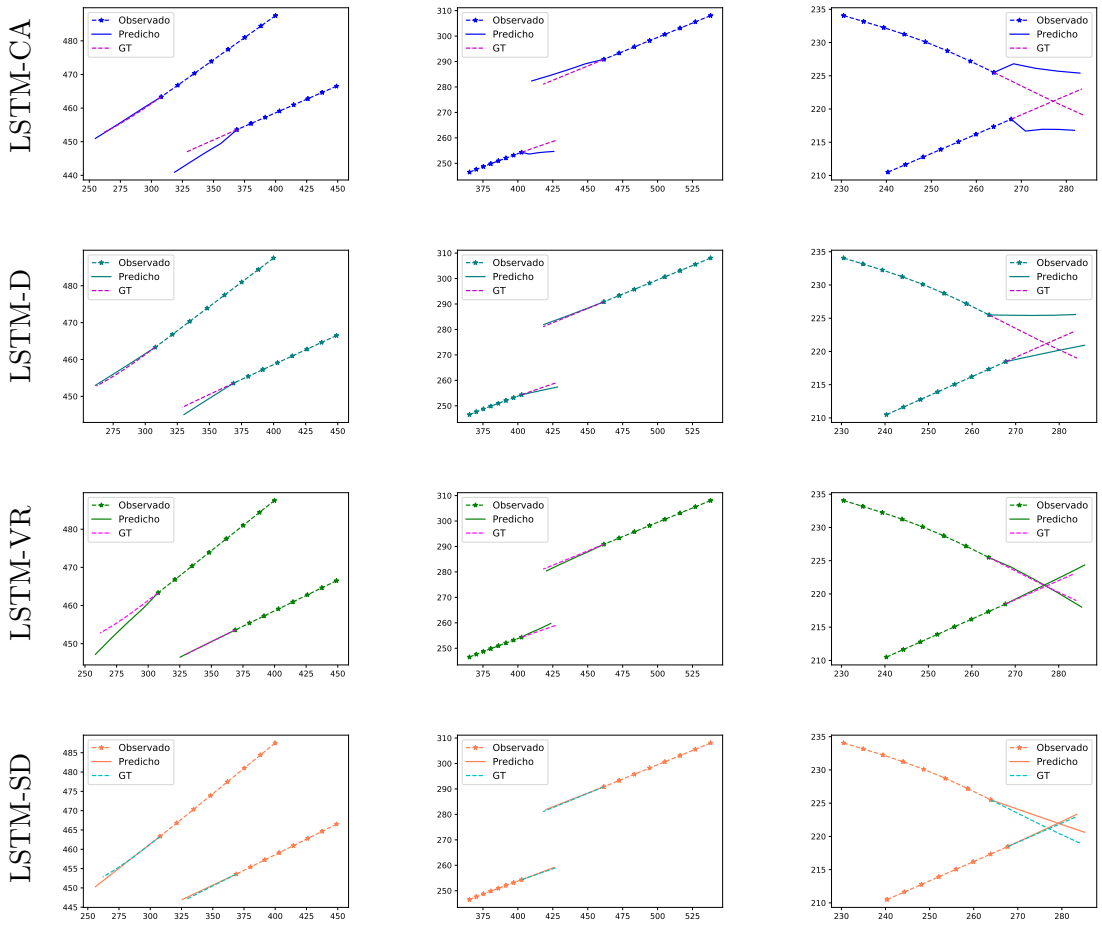
Tabla 3.4: Comparación cuantitativa entre las diferentes representaciones para el conjunto de datos PETS2009-S2L1 (con una baja frecuencia de muestreo).

desplazamientos, ya que al predecir 2 pasos al futuro mejora un 88.44 % en ADE y 85.65 % en FDE, mientras que al predecir 4 posiciones futuras las mejoras son de 82.68 % ADE y 73.90 % FDE respectivamente, y finalmente al predecir 8 posiciones futuras se mejora 74.34 % en ADE y un 68.26 % en FDE, donde estas mejoras son calculadas con respecto a los promedios obtenidos del modelo LSTM-CA y del modelo LSTM-SD.

En las Tablas 3.3 y 3.4, donde el muestreo es inferior (la duración de la trayectoria es más larga), el mejor modelo de salida, cuando la entrada de la red es una sucesión de coordenadas absolutas, resulta ser el modelo de desplazamientos, ya que para este enfoque del modelado de la salida se obtuvieron mejoras del 48.73 % en ADE y 41.67 % en FDE al predecir 2 posiciones futuras, y al predecir 4 pasos se tuvo una mejora del 38.45 % y 31.96 % en ADE y FDE respectivamente. Al predecir 8 pasos futuros, las mejoras fueron de 29.70 % en ADE y 24.75 % FDE, respecto a los promedios obtenidos en el modelo LSTM-CA con una frecuencia de muestreo de 3.75fps. No obstante, si observamos los resultados de forma general, la representación que da los mejores resultados es la en que se le proporciona a la red, como entrada y salida, sucesiones de desplazamientos (LSTM-SD). Las mejoras en este enfoque fueron de 80.29 % en ADE y 74.94 % en FDE al predecir 2 pasos futuros, 66.75 % ADE y 56.33 % FDE al predecir 4 pasos al futuro, mientras que al predecir 8 pasos la mejora fue de 45.83 % en ADE y 30.85 % en FDE. Estos porcentajes fueron calculados con respecto a los promedios de los errores ADE y FDE de los modelo LSTM-CA y LSTM-SD.

3.4.3. Evaluación Cualitativa

En las siguientes imágenes, podemos observar varios casos de trayectorias, donde las personas se cruzan, caminan de manera paralela a otra o caminan de frente. Lo que queremos mostrar con las imágenes, es cómo cada representación (apareciendo en las filas de la tabla) actúa en estos casos. Las pruebas se hicieron con muestras tomadas a 7.5 frames por segundo.



Paralelos De frente Cruce

De las gráficas y tablas presentadas, se concluye que la red que aprende a generar una salida en base a coordenadas absolutas o a desplazamientos necesita más tiempo para aprender sus parámetros, comparado con el modelo de variación a regresión,

cuando el muestreo es más frecuente. Una explicación es que el modelo LSTM-VR tiene que aprender a calcular un error residual chico con respeto a un "buen modelo" a priori (el modelo lineal). No obstante, cuando la frecuencia de muestreo es mas baja, el modelo con desplazamientos aprende mucho mejor, probablemente porque el modelo lineal es menos correcto a largo término.

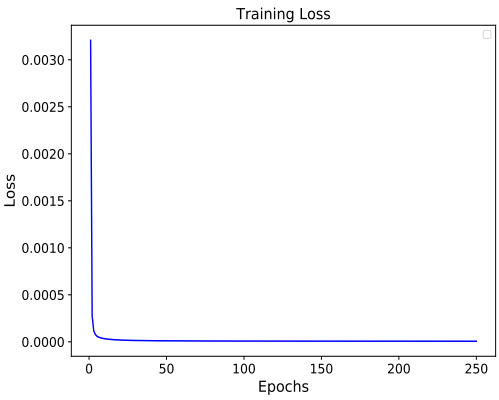
En general, usar la representación de entrada y salida por medio de desplazamientos es mejor, tanto cuando la frecuencia de muestreo es mayor o menor. Esto se debe a que es más fácil aprender los parámetros del modelo que expresa toda la secuencia en desplazamientos que aquella que lo que tiene que aprender es a calcular el error entre una regresión y la posición verdadera, a partir de coordenadas absolutas.

3.5. Comparación con otros modelos

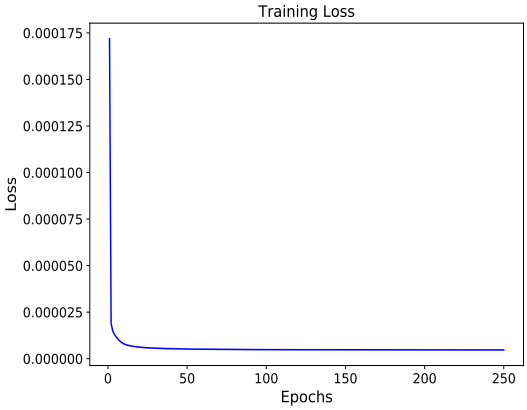
En esta sección, compararemos el modelo simple que ha servido de base a los tres enfoques que hemos presentado anteriormente, con otros modelos de la literatura tales como *Vanilla LSTM*, en el cual se codifican las coordenadas de cada peatón con una sola LSTM, y por medio de una distribución Gaussiana Bivariada. Para ello, entrenamos la red, otra vez, con 4 conjuntos de datos y probamos con el sobrante. En este caso, cada conjunto de datos se forma con una frecuencia de muestreo de 2.5 frames por segundo (es decir que investigamos la predicción a más largo término, lo que, probablemente, desfavorece los modelos lineales). En particular, comparamos los resultados sobre los conjuntos de datos: Crowds Zara1, Crowds Zara2, UCY Univ, ETH Hotel y ETH Univ. Los resultados de las pruebas con *Vanilla LSTM*, son generalmente más grandes que los propuestos por nosotros, esto es por que se usan coordenadas absolutas en el entrenamiento y se usa una sola LSTM para cada peatón, en cambio nosotros usamos dos LSTM.

En la Fig. 3.8, se presentan las gráficas de entrenamiento de cada uno de los modelos. El conjunto de entrenamiento son los datos de ETH/UCY a excepción de ETH-Hotel que en este caso sirvió de conjunto de prueba.

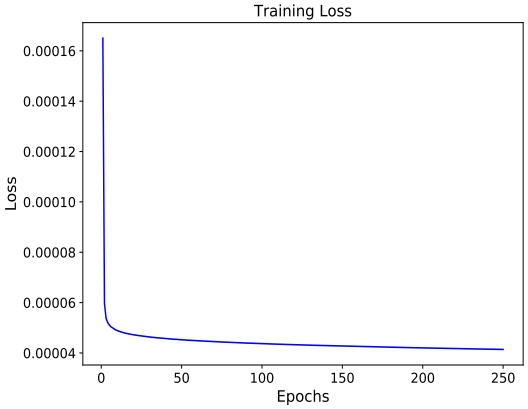
Para el modelo *Vanilla LSTM* la función de pérdida es el negativo de log-verosimilitud y en los modelos que proponemos, la función de pérdida a optimizar es logcosh la cual se explica en la Ecuación 2.10. De las gráficas, se puede observar que



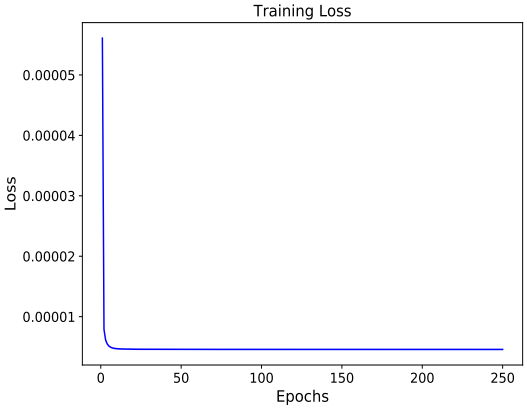
(a) LSTM-CA



(b) LSTM-D



(c) LSTM-VR



(d) LSTM-SD

Figura 3.8: Gráficas de pérdida vs épocas.

Métricas	Conjunto de prueba	Baselines		Nuestros			
		Lineal	LSTM	LSTM-CA	LSTM-D	LSTM-VR	LSTM-SD
Error de desplazamiento promedio (**ADE**)	Zara1	0.62	**0.41**	0.717	0.47	0.539	0.42
	Zara2	0.77	0.52	0.538	0.415	0.487	**0.337**
	UCY Univ	0.82	**0.61**	0.742	0.721	0.871	**0.607**
	ETH Hotel	0.39	0.86	0.469	0.357	**0.31**	0.36
	ETH Univ	1.33	1.09	1.306	1.19	1.197	**1.066**
AVG ADE		0.786	0.698	0.754	0.631	0.681	**0.558**
Error de desplazamiento final (**FDE**)	Zara1	1.21	**0.88**	1.636	1.03	1.068	0.925
	Zara2	1.48	1.11	1.231	0.93	0.985	**0.747**
	UCY Univ	1.59	**1.31**	1.597	1.58	1.737	**1.307**
	ETH Hotel	0.72	1.91	0.979	0.716	**0.608**	0.674
	ETH Univ	2.94	2.41	2.812	2.471	2.239	**2.128**
AVG FDE		1.588	1.524	1.651	1.345	1.327	**1.156**

Tabla 3.5: Comparación de modelos entrenados con el enfoque LOO (ver Sección 2.6) para predecir 12 posiciones futuras. Los errores son dados en metro.

entre las diferentes representaciones, la que usa sólo desplazamientos (LSTM-SD) inicia el entrenamiento con un error más pequeño. Cabe mencionar que los resultados con los cuales comparamos nuestros modelos en la Tabla 3.5, fueron tomados del artículo [23]

3.6. Conclusión

De las diferentes pruebas realizadas en este capítulo, con los diversos conjuntos de datos, se puede concluir que el entrenamiento es bastante sensible a la forma que elegimos para representar la entra y la salida del modelo. Basta con recordar que las mejoras del modelo LSTM-SD con respecto al modelo LSTM-CA para predecir 8 posiciones futuras fueron de 74.34 % en ADE y 68.26 % en FDE para una frecuencia de muestreo de 7.5fps. Para el mismo número de pasos a predecir pero con una frecuencia de muestreo de 3.75fps, las mejoras fueron de 45.83 % en ADE y 30.85 % en FDE. Además, concluimos que trabajar con datos expresados en coordenadas absolutas no es una buena opción.

Capítulo 4

Contribución en el uso de la pose para la inferencia de trayectorias

En este capítulo, estudiaremos cómo la información de pose puede contribuir en la predicción de posiciones futuras. Creemos que, a partir de la pose de una persona, podemos predecir comportamientos futuros de ella, como se ilustra en la Fig. 4.1. La posición de las piernas y de los brazos nos pueden dar información para determinar si una persona está caminando (ilustración de la derecha) o si se encuentra en estado de reposo (ilustración de la izquierda), así como la dirección hacia la cual se está moviendo.

Es por ello que para estudiar, si la pose contribuye en la predicción de trayectorias de peatones, dividimos este capítulo de la siguiente manera. Primero en la Sección 4.1 se describe como se procesan las secuencias de trayectorias históricas y las secuencias de pose, para el entrenamiento del modelo. En seguida en la Sección 4.2 se ilustra y describe cada una de las partes que conforman la arquitectura del modelo con el cual se pretende evaluar si la información de pose contribuye en las predicciones. Mientras que en la Sección 4.3 se presentan cada uno de los experimentos que se realizaron para medir si hay alguna contribución al usar esta información. Y finalmente en la Sección 4.4 se dan las conclusiones a las que se llegaron con este capítulo.

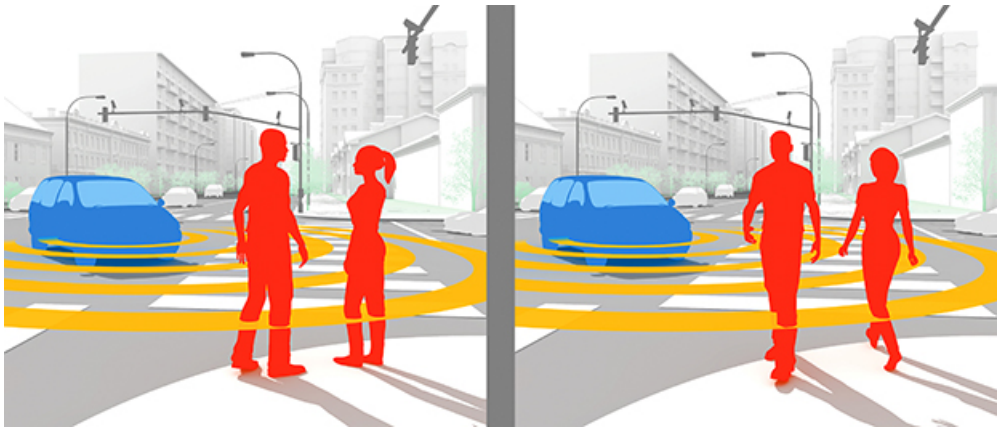


Figura 4.1: Una ilustración de cómo la apariencia de los peatones puede permitir inferir información sobre sus futuras acciones.

4.1. Preparación de los datos

En esta sección, se explicará primero cómo se preparan los datos, para entrenar la red. Para esto, lo primero que se hace es agrupar la información de posición (coordenadas absolutas) de cada frame. De esa manera, obtenemos una lista donde cada item de ésta contiene toda la información de cada frame $(frame, Id, x, y)$ con, respectivamente, el id $frame$ del frame, el id Id de la persona (al cual nos referimos por id-person), y las coordenadas x y y de la persona. Tomemos en cuenta que los frames están ordenados de menor a mayor. También procesamos la información de los keypoints (puntos clave detectados en la imagen), y esto lo almacenamos con la ayuda de un diccionario, donde la clave de cada elemento es la cadena "frame+_+Id" (+ es la concatenación de cadenas), y la información a guardar con esa clave es una matriz 18×3 de keypoints, como se explica en la Ecuación 2.24 de la Sección 2.4.

Posteriormente, tomamos todas las posibles sucesiones de tamaño obs_len + $pred\_len$ de la lista que tiene la información de cada frame. A la sucesión de frames resultante, le evaluamos la cantidad de personas únicas presentes en toda la trayectoria, es decir la intersección de los id-person en toda la sucesión. Después, se recorre cada uno de los id-person resultantes de la operación anterior y se obtiene la trayectoria correspondiente para ese peatón, quedándonos sólo con las coordenadas (x, y) de cada paso. Así obtenemos, para un agente i, una sucesión $\mathbf{x}^i_{1:T_{pred}}$ de

posiciones, la cual consta de la parte observada ($\mathbf{x}^i_{1:T_{obs}}$) y de la parte que se va a predecir ($\mathbf{x}^i_{T_{obs}+1:T_{pred}}$).

Con esta sucesión de coordenadas absolutas, construimos la sucesión de desplazamientos como:

$$\delta\mathbf{x}^i_1 = [0, 0], \tag{4.1}$$

$$\delta\mathbf{x}^i_{2:T_{pred}} = \mathbf{x}^i_{2:T_{pred}} - \mathbf{x}^i_{1:T_{pred}-1}, \tag{4.2}$$

considerando que en el primer instante de tiempo el peatón aún no se ha movido. Luego, se encuentra la información de keypoints correspondiente a $\mathbf{x}^i_{1:T_{pred}}$, ya que tenemos los id de los frames y el id-person de esta sucesión de trayectoria y así construimos $\mathbf{k}^i_{1:T_{pred}}$, la cual es la sucesión de keypoints. En cada instante de tiempo de esta sucesión, se tiene 18 tripletas (x, y, π), las cuales corresponden a la posición x, y de una parte del esqueleto humano del peatón en ese instante de tiempo, además de un índice π de confianza (probabilidad) sobre esa posición. De manera análoga a las posiciones en coordenadas absolutas, se construye la secuencia $\delta\mathbf{k}_{1:T_{pred}}$ de los *desplazamientos* de las coordenadas de los keypoints. Los desplazamientos de los keypoints se calculan encontrando la diferencia de cada posición de keypoint con su respectiva posición en el instante de tiempo anterior (similarmente a flujo óptico). Con respecto a las probabilidades, lo que hacemos es tomar el producto de las dos (anterior y actual), ya que la identificación de cada keypoint es independiente de su respectivo en otro instante de tiempo.

4.2. Descripción de la red neuronal

En muchos sistemas recientes basados en redes neuronales, como en Liang et al. [16] y Xu et al. [27], para las aplicaciones que nos interesan, los autores incluyen un mecanismo de atención, que consiste en aprender a seleccionar y resaltar elementos procesados del contexto (en el tiempo o en el espacio) para "enfocar" la atención de las últimas capas de una red (las que deciden) en la información mas relevante para ellas. Típicamente, esos mecanismos involucran proyectar múltiples características en un espacio de correlación (correlación entre la información del contexto y la información espacial actual de una trayectoria).

4.2.1. Información de entrada

Para la aplicación que nos interesa, el tipo de información en que extraemos características, es aquella de la cual podamos aprender comportamientos naturales de los peatones, para luego poder predecir las trayectorias futuras que tomen en cuenta esos patrones. A continuación, describimos algunas de las fuentes de información que utilizamos.

Una de las principales fuentes de información que utilizamos para extraer características son las secuencias de las posiciones observadas de cada peatón i, es decir, $\mathbf{x}^i_{1:T_{obs}}$. Para lograr mayor generalización, y motivados por los resultados del capítulo anterior, no utilizamos las coordenadas absolutas, sino los desplazamientos $\delta\mathbf{x}^i_{1:T_{obs}}$ introducidos en la ecuación 4.2. Por lo tanto, la secuencia de desplazamientos de las posiciones observadas hace parte de la información que recibe como entrada el modelo. A cada posición de la secuencia se le aplica una capa lineal de tipo *embedding*, la cual modifica las dimensiones de las coordenadas x, y de los desplazamientos, en cada instante de tiempo de la secuencia observada. Finalmente, la información secuencial codificada de esta manera se procesa en una red recurrente LSTM, a la cual le pedimos que nos regrese la secuencia de los estados ocultos.

Si recordamos el capítulo 2, las celdas LSTM retornan un vector $\mathbf{h}_t$ de dimensión fija d, al cual se le conoce como estado oculto, para el instante de tiempo t. Cuando la celda LSTM recibe como entrada una secuencia de longitud T_{obs} en la codificación de los desplazamientos de las posiciones históricas, entonces para cada instante de tiempo del histórico, se genera un estado oculto $\mathbf{h}_t$. Por lo tanto, obtenemos una secuencia $\{\mathbf{h}_1, .., \mathbf{h}_{T_{obs}}\}$ de T_{obs} estados ocultos, para una secuencia de entrada de la forma $\delta\mathbf{x}^i_{1:T_{obs}}$.

En la versión de nuestra red que describimos en este capítulo, otros datos que recibe la red es la *pose*, la cual es codificada con un conjunto de puntos $2D$ (en la imagen) que conforman el esqueleto de cada peatón i, en un instante de tiempo t, es decir los keypoints mencionados anteriormente.

Para esa información de pose, como lo hemos mencionado anteriormente, se tienen *sucesiones* de keypoints para cada tiempo observado, para cada peatón i. Nos referimos a esas sucesiones como $\mathbf{k}^i_{1:T_{obs}}$. De manera análoga a como se codifica la información de coordenadas, se evalúa desplazamientos $\delta\mathbf{k}^i_{1:T_{obs}}$ y a cada dato de esa

Figura 4.2: Keypoints: Cada circulo da un par de coordenadas x, y de uno de los puntos-llave. El conjunto de las coordenadas encodifica implícitamente la pose de la persona.

secuencia, se le aplica una capa lineal. La secuencia obtenida de este mapeo se procesa a través de un LSTM, con el cual obtenemos la secuencia de longitud T_{obs} de los estados ocultos que codifican la información de pose.

Al entrenar nuestro modelo, tomamos batches de secuencias observadas $\delta\mathbf{x}^i_{1:T_{obs}}$ (desplazamientos de las posiciones), y $\delta\mathbf{k}^i_{1:T_{obs}}$ (desplazamientos de las poses), para $i \in \{1, .., N\}$, como entrada a la red. Cabe mencionar que la información de cada fuente (posiciones/keypoints) es codificada por una LSTM distinta (pero de tamaño de estado oculto igual, a d). Para mayor claridad en la explicación del mecanismo de atención, supondremos en las líneas siguientes que el tamaño de batch es uno.

Las secuencias de los estados ocultos obtenidas de las secuencias de desplazamientos de trayectorias y de keypoints de la parte observada son denotadas como $\{\mathbf{h}_1, ..., \mathbf{h}_{T_{obs}}\}$, y $\{\mathbf{h}^k_1, ..., \mathbf{h}^k_{T_{obs}}\}$, respectivamente. Estas secuencias de estados ocultos son las características que conforman el contexto de la persona, y que entrenamos a proyectar en un espacio de correlación, para que la red pueda seleccionar y resaltar la característica que le de mayor información en un respectivo instante de tiempo.

4.2.2. Mecanismo de atención

De aquí en adelante, se explica como se realiza el mecanismo de atención en la etapa de *descodificación*, para nuestra aplicación, con las características ya extraídas. En este caso, estamos mencionando dos características, pero si se requiere agregar otro tipo de información que ayude al aprendizaje de la red, entonces lo agregaremos de manera análoga a las características antes mencionadas. La descodificación se detallará con más detalles en la sección siguiente; como se basa en una red recurrente, basta mencionar aquí que en cada momento de la descodificación, tenemos un vector de estado oculto asociado a la trayectoria descodificada. Es este estado oculto que se busca modificar en función del contexto, a través del mecanismo de atención.

Lo primero que se realiza es agrupar las características de contexto colectadas (secuencias de estados ocultos correspondiendo a las observaciones), y lo hacemos por medio de un tensor $\mathbf{Q}$ de tres dimensiones, donde la información está representada de la siguiente manera: En cada fila del tensor, se almacena información de una característica (coordenadas, keypoints, ...), mientras que la segunda dimensión denota cada uno de los estados ocultos extraídos para todos los instantes de tiempo observados de la característica en particular. Por lo anterior,

$$\mathbf{Q} \in \mathbb{R}^{M \times T_{obs} \times d}, \tag{4.3}$$

donde M denota el número de características usadas (coordenadas y secuencia de keypoints, etc). Por ejemplo, si sólo usamos coordenadas y keypoints, $M = 2$.

Para calcular el vector de atención en cada instante de tiempo t de la predicción, se hace lo siguiente:

- Se obtiene $\mathbf{h}_{t-1}^{d} \in \mathbb{R}^{d}$, el cual es el último estado oculto del decodificador en el instante de tiempo anterior $(t-1)$ (ver sección siguiente).
- Se calcula la matriz de correlación $\mathbf{S}^{t} \in \mathbf{R}^{M \times T_{obs}}$, donde cada entrada es $\mathbf{S}_{ij}^{t} = (\mathbf{h}_{t-1}^{d})^{T} \cdot \mathbf{Q}_{ij:}$, $i \in [1, M]$, $j \in [1, T_{obs}]$, por lo que $\mathbf{Q}_{ij:}$ son las filas de tamaño d, de las M submatrices de dimensión $T_{obs} \times d$.
- Se calcula $\mathbf{P}$, un vector donde su $i - ésima$ entrada es el máximo de $\mathbf{S}_{i:}$, con $i \in [1, M]$, por lo tanto $\mathbf{P} \in \mathbb{R}^{M}$.

- Se calculan las matrices de atención

$$\mathbf{A}^t = \text{softmax}(\mathbf{P}) \in \mathbb{R}^M,$$

y

$$\mathbf{B}^t = \begin{pmatrix} \text{softmax}(\mathbf{S_{1:}}) \\ \vdots \\ \text{softmax}(\mathbf{S_{M:}}) \end{pmatrix} \in \mathbb{R}^{M \times T_{obs}}.$$

La función softmax está dada por $\sigma : \mathbb{R}^n \to [0,1]^n$

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \text{ para } i = 1, ..., N.$$

El vector que resume las características ponderadas por los coeficientes de atención está dado por:

$$\mathbf{q}^t = \sum_{i}^{M} \mathbf{A}_i^t \Big(\sum_{j=1}^{T_{obs}} \mathbf{B}_{ij}^t \mathbf{Q}_{ij:} \Big) \in \mathbb{R}^d. \tag{4.4}$$

Vale la pena mencionar que para calcular $\mathbf{q}^t$ como lo vimos arriba, se usa un doble mecanismo de atención, ya que se tienen dos matrices de atención. La primera es $\mathbf{A}^t$, la cual es más global y da un peso a cada uno de las M características. Cada característica tiene un peso con respecto a la importancia que representa para la predicción que realizar en el tiempo t.

En cambio, la matriz $\mathbf{B}^t$, es más minuciosa, porque en ella no se da un peso global a la característica, sino que a cada vector que describe un instante de tiempo observado de la característica se le da un peso. Por lo tanto, $\mathbf{q}^t$ es un vector de suma pesada, donde se toma en cuenta la importancia de cada estado oculto de la secuencia de longitud T_{obs} de una característica ($\mathbf{B}_{ij}$), y posteriormente se pesa por la importancia global de dicha característica ($\mathbf{A}_i$).

La idea es resumir en esta variable vectorial $\mathbf{q}^t$ la información más pertinente en

el contexto que nos pueda servir en la última fase, la de la decodificación.

4.2.3. Decodificación y generación de la trayectoria

Para generar la trayectoria futura, utilizamos un decodificador LSTM, el cual es inicializado con el último estado oculto del codificador LSTM de los desplazamientos de la trayectoria observada del peatón, es decir con $\mathbf{h}_{T_{obs}}$.

En su funcionamiento recurrente, para cada $t > T_{obs}$, dada la predicción realizada sobre la posición de una persona en el instante de tiempo $t-1$ anterior, transformamos esas coordenadas como:

$$\mathbf{e}_{t-1} = \tanh\{\, W_e[\delta \mathbf{x}_{t-1}]\,\} + b_e, \tag{4.5}$$

donde $\delta \mathbf{x}_{t-1}$ es el último desplazamiento predicho de la trayectoria, mientras que W_e y b_e son parámetros que se aprenden.

El siguiente desplazamiento se calcula *recursivamente* a partir del estado del decodificador

$$\mathbf{h}_t^d = LSTM(\mathbf{h}_{t-1}^d, [\mathbf{e}_{t-1}, \mathbf{q}^t]) \tag{4.6}$$

donde se hace uso del vector de atención (contexto) descrito anteriormente. Para terminar, una capa lineal mapea los estados ocultos a desplazamientos mediante:

$$\delta \mathbf{x}_t = W_p \mathbf{h}_t^d, \tag{4.7}$$

para obtener las dos coordenadas que representan la abscisa y la ordenada de dicho desplazamiento.

4.2.4. Arquitectura del modelo

La Fig. 4.3 resume la arquitectura de la red, la cual está inspirada de [16]. La primera parte (las capas de embedding y LSTM) es el encoder, el cual se encarga de codificar la información de entrada como se explicó en la Subsección 4.2.1. En la sección 4.2.2, se ha detallado cómo funciona el mecanismo de atención, es decir, cómo se obtiene el vector de atención $\mathbf{q}^t$ en base al tensor de características $\mathbf{Q}$ y en el

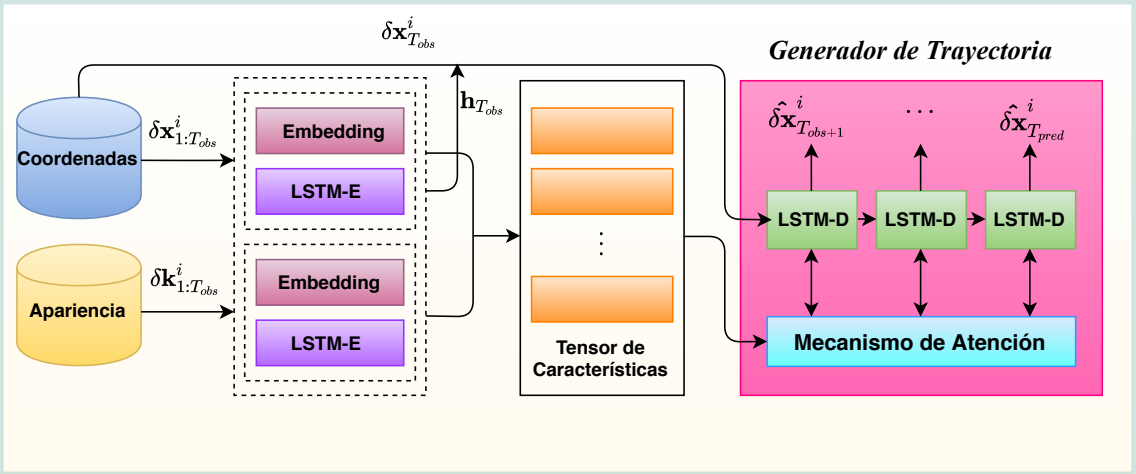


Figura 4.3: Arquitectura del modelo usando coordenadas espaciales y keypoints.

estado oculto actual del descodificador. Finalmente, en la parte derecha, se encuentra el decoder, el cual se encarga de generar las trayectorias futuras (desplazamientos futuros).

4.3. Evaluación

Para evaluar si la información de pose contribuye en la predicción de trayectorias futuras, lo hacemos con tres conjuntos de datos (PETS-S2L1 [8], TOWNCENTRE [3] y ActEV [2]), los cuales tienen una frecuencia de muestreo de 2.5 frames por segundo. Utilizamos estos datos y no los más clásicos (ETH, UCY...) porque necesitamos que las siluetas de los peatones y la caja englobante de cada uno de los peatones sean visibles, para calcularles sus keypoints con mayor precisión. Si los puntos de interés del esqueleto son mal calculados, entonces estaríamos metiendo más ruido al modelo. Por esa razón, integramos el indicador de confianza de cada punto clave del esqueleto para que el modelo aprenda a decidir cuando/cuanto tomarlo en cuenta, lo integramos como se explica en la Sección 4.1.

Las trayectorias de los primeros dos conjuntos están expresadas en marco mundo, mientras que los keypoints están expresados en píxeles. El último tiene tanto las trayectorias como los keypoints expresados en píxel en el marco de la imagen.

4.3.1. Evaluación de la contribución de la pose

Comparamos los resultados que nos proporciona el modelo que sólo tiene como información de entrada el histórico de la trayectoria (desplazamientos) con los que nos proporciona el modelo que además tiene la información de los keypoints.
Para este caso, los keypoints están expresados en el marco de la imagen y los datos de la trayectoria observada $\delta\mathbf{x}^i_{1:T_{obs}}$ en el marco mundo. Al hacer pruebas, se observó que es mejor trabajar con los keypoints definidos de manera relativa a las dimensiones de la caja que contiene la silueta del peatón en dicho instante de tiempo. Suponemos que la caja que contiene al peatón tiene dimensiones h como largo y w como ancho, por lo que ahora el $(0, 0)$ en coordenadas imagen, a partir del cual ubicamos a los keypoints, será la esquina superior izquierda de la caja. También cabe señalar que se analizó si agregar el valor de confianza de la detección de cada keypoints es de ayuda para el entrenamiento.

Entrenamiento

Para el entrenamiento de los modelos se hace con el enfoque **PRO**, descrito en la sección 2.6, es decir, dividiendo un solo dataset en sub-conjuntos de entrenamiento, validación y prueba.

Para el primer caso de nuestras pruebas, es decir la Tabla 4.1, no se toma en cuenta el valor de confianza de cada uno de los puntos claves detectados, por lo que sólo se procesa sus coordenadas:

$$\mathbf{p}_1 = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_{18} & y_{18} \end{pmatrix}. \tag{4.8}$$

Así construimos las sucesiones $\delta\mathbf{x}_{1:T_{pred}}$ y $\delta\mathbf{k}_{1:T_{obs}}$, como se mencionó en la Sección 4.1. Los parámetros que hemos usado para el entrenamiento son: 100 épocas, tasa de aprendizaje de 0.002, tamaño de batch de 20, tamaño del embedding de 64 (en todos los casos), tamaño del estado oculto, tanto en el encoder y en el decoder, de 128. Se optimizó el error cuadrático medio con el optimizador Adam.

Evaluación de la influencia de los puntos claves

En esta prueba como se menciono arriba, se usa la información de las posiciones históricas y además la información de pose (*puntos claves del esqueleto humano*), sin el valor de confianza de cada uno de ellos.

	Basado en LSTM
	Sin VC
PETS-S2L1	0.92/2.10
TOWN-CENTRE	0.65/1.21
AVG	0.79/1.66

Tabla 4.1: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE).

Evaluación de la influencia del valor de confianza

Para el estudio de la influencia de π, valor de confianza de la detección de los puntos claves en el entrenamiento del modelo, agregamos a cada punto clave detectado su respectivo valor de π, por lo que tenemos para cada posición histórica del peatón, sus respectivos keypoints con valores de confianza:

$$\mathbf{p} = \begin{pmatrix} x_1 & y_1 & \pi_1 \\ x_2 & y_2 & \pi_2 \\ \dots & \dots & \dots \\ x_{18} & y_{18} & \pi_{18} \end{pmatrix}. \tag{4.9}$$

Los cuales son procesados como se explica en la Sección 4.1.

Estudiamos la influencia del valor de confianza al observar la matriz $\mathbf{A}^t$, la cual es una de las matrices del mecanismo de atención, descrita en la sección 4.2.2. Esta matriz nos da, en sus columnas, el porcentaje de atención que se le da a cada uno de los M tipos de información de entrada que recibe el modelo (aquí $M = 2$, ya que sólo usamos información de posición, e información de pose) al momento de predecir el siguiente instante de tiempo, para cada instante predicho (en cada fila).

A continuación, presentamos ejemplos de cómo estos porcentajes se ajustan cuando se les proporciona el valor de confianza de la detección.

Las matrices 4.10 y 4.11 son ejemplos de los valores de confianza π que toman los desplazamientos de los keypoints para una sucesión observada con el conjunto de datos de PETS2009-S2L1 y TOWN-CENTRE respectivamente. Las columnas representan cada uno de los instantes de tiempo observados y las filas cada uno de los puntos claves. Se puede ver que los valores de confianza pueden variar significativamente.

$$\begin{pmatrix}
1 & 4{,}55952622e-02 & 4{,}56086174e-02 & 5{,}01915514e-02 & 7{,}41291195e-02 & 9{,}96925756e-02 & 7{,}47324526e-02 & 3{,}39787379e-02 \\
1 & 5{,}32714240e-02 & 5{,}25074415e-02 & 5{,}46706580e-02 & 6{,}41341358e-02 & 6{,}79317862e-02 & 5{,}98080233e-02 & 5{,}17166853e-02 \\
1 & 2{,}89355051e-02 & 4{,}31738310e-02 & 4{,}62042615e-02 & 4{,}03096601e-02 & 3{,}49319726e-02 & 4{,}24266011e-02 & 4{,}71748635e-02 \\
1 & 1{,}28818760e-02 & 2{,}62165889e-02 & 3{,}15888673e-02 & 2{,}27440894e-02 & 1{,}98357645e-02 & 2{,}74406429e-02 & 3{,}45632918e-02 \\
1 & 1{,}12575786e-02 & 1{,}82133056e-02 & 1{,}54536013e-02 & 1{,}45975212e-02 & 1{,}95067786e-02 & 2{,}07208134e-02 & 3{,}21896449e-02 \\
1 & 4{,}53496873e-02 & 4{,}55349050e-02 & 4{,}53059599e-02 & 5{,}59504591e-02 & 6{,}60447031e-02 & 5{,}47561683e-02 & 4{,}56312113e-02 \\
1 & 2{,}21053194e-02 & 1{,}60055850e-02 & 1{,}38565330e-02 & 2{,}81739235e-02 & 2{,}92649530e-02 & 2{,}88374070e-02 & 1{,}36317266e-02 \\
1 & 1{,}33948307e-02 & 1{,}49581861e-02 & 1{,}06572751e-02 & 1{,}59925502e-02 & 1{,}84283461e-02 & 1{,}63120478e-02 & 8{,}52792803e-03 \\
1 & 2{,}82669496e-02 & 3{,}41912769e-02 & 3{,}53765003e-02 & 3{,}54914330e-02 & 3{,}84166278e-02 & 3{,}77502106e-02 & 3{,}43288071e-02 \\
1 & 3{,}60746719e-02 & 4{,}04617712e-02 & 4{,}56164069e-02 & 4{,}45165187e-02 & 4{,}83452827e-02 & 5{,}55078574e-02 & 3{,}80511023e-02 \\
1 & 2{,}40960177e-02 & 4{,}10880931e-02 & 4{,}62461114e-02 & 4{,}33982462e-02 & 5{,}42354211e-02 & 6{,}96638301e-02 & 3{,}77358384e-02 \\
1 & 2{,}85575874e-02 & 3{,}36707644e-02 & 3{,}32221128e-02 & 3{,}49681564e-02 & 3{,}94164585e-02 & 3{,}77048030e-02 & 3{,}30448374e-02 \\
1 & 3{,}25910896e-02 & 4{,}33481857e-02 & 4{,}00492884e-02 & 3{,}85835804e-02 & 4{,}91855107e-02 & 4{,}96433936e-02 & 2{,}81821098e-02 \\
1 & 3{,}19477282e-02 & 4{,}42840643e-02 & 4{,}11039405e-02 & 3{,}48122641e-02 & 4{,}09436673e-02 & 5{,}33075929e-02 & 1{,}53345689e-02 \\
1 & 4{,}98202071e-02 & 4{,}75577377e-02 & 3{,}70770730e-02 & 3{,}99422981e-02 & 8{,}86661112e-02 & 6{,}56533390e-02 & 2{,}48306431e-02 \\
1 & 4{,}92299348e-02 & 4{,}45430689e-02 & 3{,}50535549e-02 & 3{,}26860547e-02 & 5{,}04684448e-02 & 4{,}13288958e-02 & 2{,}41777371e-03 \\
1 & 3{,}59189659e-02 & 3{,}34868580e-02 & 3{,}80162820e-02 & 4{,}58405986e-02 & 6{,}56828061e-02 & 8{,}17630440e-02 & 5{,}97389080e-02 \\
1 & 1{,}55207021e-02 & 1{,}48262745e-02 & 6{,}35598740e-03 & 5{,}08564338e-03 & 5{,}14391810e-03 & 7{,}79474853e-04 & 4{,}71236854e-05
\end{pmatrix} \tag{4.10}$$

Las matrices siguientes son ejemplos de matrices $\mathbf{A}^t$ obtenidas en las mismas condiciones de aprendizaje, donde la matriz de la izquierda no usa el valor de confianza (Sin VC), y la de la derecha sí hace uso de los valores de confianza (Con VC) de la matriz 4.10.

```
[[0.6788373  0.32116267]      [[0.7125421  0.28745788]
 [0.6661929  0.33380714]       [0.69458896 0.305411  ]
 [0.6538152  0.3461848 ]       [0.6757678  0.32423225]
 [0.6431601  0.3568399 ]       [0.65750265 0.34249732]
 [0.6358096  0.3641904 ]       [0.64406884 0.3559312 ]
 [0.62702644 0.37297356]       [0.6351363  0.36486366]
 [0.6198495  0.38015053]       [0.6286981  0.37130183]
 [0.6141743  0.38582572]       [0.6240554  0.3759446 ]
 [0.6096637  0.3903363 ]       [0.62070435 0.37929568]
 [0.60605127 0.3939487 ]       [0.6176496  0.38235036]
 [0.6031324  0.39686754]       [0.61458445 0.38541558]
 [0.6007512  0.39924875]]      [0.61213505 0.387865  ]]
```

Figura 4.4: Porcentaje de atención relativa entre posiciones y keypoints ($\mathbf{A}_i^t$ de la ecuación 4.4), en los datos de PETS-S2L1. A la izquierda, sin valores de confianza; a la derecha, con valores de confianza.

$$\begin{pmatrix}
1 & 2{,}72901535e-01 & 4{,}98338938e-01 & 3{,}93912375e-01 & 3{,}10995221e-01 & 1{,}11772664e-01 & 1{,}42599493e-01 & 5{,}47312200e-01 \\
1 & 2{,}50889331e-01 & 3{,}23021710e-01 & 2{,}65799910e-01 & 2{,}17720211e-01 & 1{,}82526425e-01 & 2{,}12445945e-01 & 3{,}79646391e-01 \\
1 & 2{,}15162516e-01 & 2{,}61533022e-01 & 2{,}05265313e-01 & 1{,}79366842e-01 & 1{,}50376424e-01 & 1{,}64109513e-01 & 2{,}83055365e-01 \\
1 & 9{,}07287449e-02 & 8{,}90217423e-02 & 9{,}28724408e-02 & 6{,}53296784e-02 & 3{,}81687470e-02 & 6{,}42279387e-02 & 1{,}10565804e-01 \\
1 & 4{,}98817302e-02 & 5{,}15431724e-02 & 6{,}15970418e-02 & 4{,}56062593e-02 & 1{,}52463028e-02 & 2{,}92683784e-02 & 2{,}68672574e-02 \\
1 & 2{,}29584321e-01 & 3{,}00747126e-01 & 2{,}61027366e-01 & 2{,}04487622e-01 & 1{,}79728299e-01 & 2{,}04907045e-01 & 3{,}18890005e-01 \\
1 & 1{,}01464830e-01 & 1{,}11789778e-01 & 1{,}30978480e-01 & 1{,}29165739e-01 & 9{,}32366997e-02 & 1{,}18665300e-01 & 1{,}08546488e-01 \\
1 & 5{,}64284772e-02 & 3{,}92092243e-02 & 4{,}67821062e-02 & 5{,}25806025e-02 & 4{,}00563702e-02 & 6{,}43952787e-02 & 3{,}00370362e-02 \\
1 & 9{,}42886472e-02 & 1{,}06432542e-01 & 1{,}06830359e-01 & 1{,}13655865e-01 & 1{,}10925116e-01 & 1{,}05418839e-01 & 1{,}14176214e-01 \\
1 & 5{,}70909902e-02 & 6{,}45662993e-02 & 6{,}62396252e-02 & 6{,}41923919e-02 & 5{,}15552498e-02 & 7{,}80484602e-02 & 5{,}33796772e-02 \\
1 & 4{,}87361513e-02 & 6{,}26685992e-02 & 5{,}48266731e-02 & 5{,}08130081e-02 & 2{,}71106381e-02 & 5{,}29192649e-02 & 5{,}87467067e-02 \\
1 & 8{,}60704407e-02 & 1{,}02291726e-01 & 1{,}04131319e-01 & 1{,}11571722e-01 & 1{,}09198324e-01 & 1{,}02208622e-01 & 1{,}13778986e-01 \\
1 & 3{,}68586108e-02 & 4{,}96043749e-02 & 6{,}54135793e-02 & 8{,}68610293e-02 & 7{,}14984089e-02 & 5{,}08817285e-02 & 5{,}32286204e-02 \\
1 & 2{,}32433155e-02 & 2{,}98524853e-02 & 5{,}46460859e-02 & 8{,}28872994e-02 & 3{,}32039446e-02 & 1{,}42197134e-02 & 2{,}62205228e-02 \\
1 & 2{,}40743950e-01 & 5{,}09029806e-01 & 3{,}97326887e-01 & 2{,}91404217e-01 & 9{,}20410603e-02 & 1{,}00922883e-01 & 3{,}06671113e-01 \\
1 & 2{,}52301097e-01 & 4{,}99880284e-01 & 3{,}89391154e-01 & 3{,}05841357e-01 & 1{,}15661532e-01 & 1{,}48523226e-01 & 5{,}57295382e-01 \\
1 & 1{,}39320418e-01 & 2{,}96573550e-01 & 2{,}28740543e-01 & 1{,}52044207e-01 & 2{,}84184329e-02 & 1{,}19862624e-03 & 1{,}65767517e-04 \\
1 & 9{,}48820412e-02 & 1{,}61340952e-01 & 1{,}40659854e-01 & 1{,}51018843e-01 & 1{,}04000300e-01 & 1{,}25889108e-01 & 3{,}53075862e-01
\end{pmatrix} \tag{4.11}$$

En la Fig. 4.5 se presentan las matrices $\mathbf{A}^t$ obtenidas para un ejemplo de TOWNCENTRE, donde de nuevo el dato de la izquierda es obtenida sin usar los valores de confianza de la detección de los keypoints y la de la derecha haciendo uso de los valores de confianza dados en 4.11.

```
[[0.60087144 0.3991286 ]      [[0.61319953 0.38680038]
 [0.6242607  0.37573925]       [0.60770553 0.39229447]
 [0.61795396 0.38204607]       [0.5981646  0.40183535]
 [0.61008054 0.38991946]       [0.5912661  0.40873393]
 [0.6041643  0.3958357 ]       [0.58494556 0.4150544 ]
 [0.6003701  0.39962992]       [0.5806479  0.41935208]
 [0.5981055  0.4018945 ]       [0.57766527 0.42233473]
 [0.59679866 0.40320137]       [0.5755869  0.42441314]
 [0.59604937 0.40395063]       [0.57421845 0.42578155]
 [0.5956111  0.40438896]       [0.57362014 0.42637983]
 [0.5953412  0.40465882]       [0.5731853  0.42681468]
 [0.5951603  0.40483966]]      [0.5728953  0.42710465]]
```

Figura 4.5: Porcentaje de atención relativa entre posiciones y keypoints ($\mathbf{A}_i^t$ de la ecuación 4.4), en los datos de TOWN-CENTRE. A la izquierda, sin valores de confianza; a la derecha, con valores de confianza.

Cabe mencionar que la primera columna de cada una de las matrices ($\mathbf{A}^t$) anteriores representa el porcentaje asignado a la información de posición y la otra columna es el porcentaje asignado a la información de pose.

Con los ejemplos de los valores de confianza y de las respectivas matrices $\mathbf{A}^t$ producidas, se puede inferir que los porcentajes que se le asigna a la información de pose parecen depender del valor de confianza de los keypoints, ya que en la Fig. 4.4 se puede apreciar que en la matriz de la derecha decrecen los porcentaje asignados a la información de pose, cuando se le da como información los valores de confianza al modelo. Esto puede ser porque dichos valores de confianza 4.10 son muy pequeños, es decir que no hubo una buena detección de los keypoints. Para el ejemplo de TOWN-CENTRE, los valores de confianza son un poco mejores que los de PETS2009-S2L1, y esto se refleja en la matriz de la derecha de la Fig. 4.5, donde el peso atribuido a la información de pose es mayor.

Para confirmar que estos valores sí son de ayuda, entrenamos el modelo que tiene como datos de entrada las informaciones de posición y de pose, y opcionalmente tomamos en cuenta el valor de confianza de la detección de los keypoints que son los que describen la pose del peatón. En la Tabla 4.2, se observan los resultados de la prueba. Para cada conjunto de datos (filas), y para cada configuración (sin o con los valores de confianza), reportamos el error ADE y el error FDE (ver sección 2.7).

Se puede concluir que usar el valor de confianza lleva a mejores resultados (errores inferiores), aunque por un margen muy chico (del orden de dos por ciento).

	Basados en LSTM	
	Sin VC	Con VC
PETS-S2L1	**0.92**/2.10	0.92/**2.08**
TOWN-CENTRE	0.65/1.21	**0.62**/**1.15**
AVG	0.79/1.66	**0.77**/**1.62**

Tabla 4.2: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con los datasets PETS2009-S2L1 y TOWN-CENTRE para las configuraciones Sin VC y Con VC.

De aquí en adelante, cuando se habla del modelo de pose, se contempla que ya tiene integrado los valores de confianza de la detección.

Evaluación cuantitativa

Lo que se quiere observar con los experimentos siguientes es si la información de pose contribuye en mejorar la predicción de trayectorias, comparado con la situación en que no usamos esa información. Para ello al momento de entrenar los modelos, esto se hace con el enfoque **PRO** explicado en la Sección 2.6.

	Basados en LSTM	
	Inf. Pos	Inf. Pos + Inf. Pose
PETS-S2L1	1.00/2.30	**0.92/2.08**
TOWN-CENTRE	0.66/1.24	**0.62/1.15**
PETS/TOWN-CENTRE	0.77/1.58	**0.75/1.51**
AVG	0.81/1.71	**0.76/1.58**

Tabla 4.3: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con los datasets PETS-S2L1 y TOWN-CENTRE para las configuraciones sin pose y con pose.

De los resultados expresados en la Tabla 4.3, se puede observar que, cuando al modelo se le agrega la información de pose (keypoints), los resultados sí mejoran en un siete por ciento, es decir que, al usar la información de pose, obtenemos errores menores.

	Basados en LSTM	
	Inf. Pos	Inf. Pos + Inf. Pose
ActEV	18.74/38.90	**18.52/38.36**

Tabla 4.4: Predicción de 12 posiciones futuras: Errores en píxeles (ADE/FDE), con el dataset ActEV para las configuraciones sin pose y con pose.

En la Tabla 4.4, para la base de datos ActEV, los datos están expresados ahora en el marco imagen. Se puede observar una mejora al usar la información de pose, pero mucho mas pequeña que la obtenida con los conjuntos que están en marco mundo.

Evaluación cualitativa

A continuación, en las Figs. 4.6 y 4.7, se muestran algunos resultados de las predicciones haciendo uso de información de trayectorias históricas y de pose. En negro, hacemos aparecer la información del ground-truth, mientras en rojo y azul, presentamos las predicciones sin y con información de pose, respectivamente.

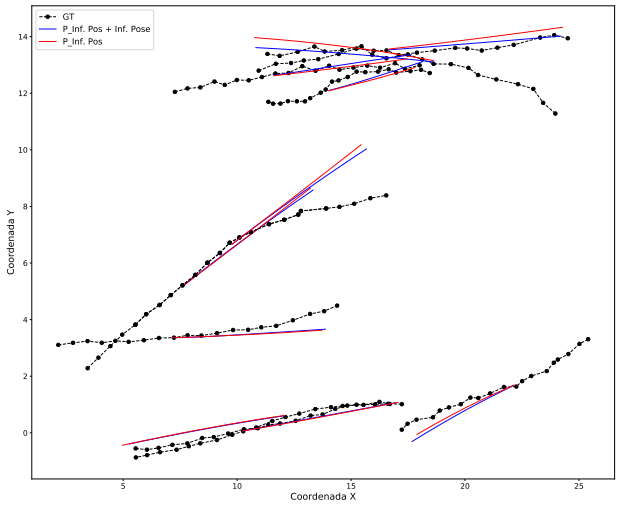


Figura 4.6: Comparación cualitativa de los modelos en el conjunto de TOWN-CENTRE.

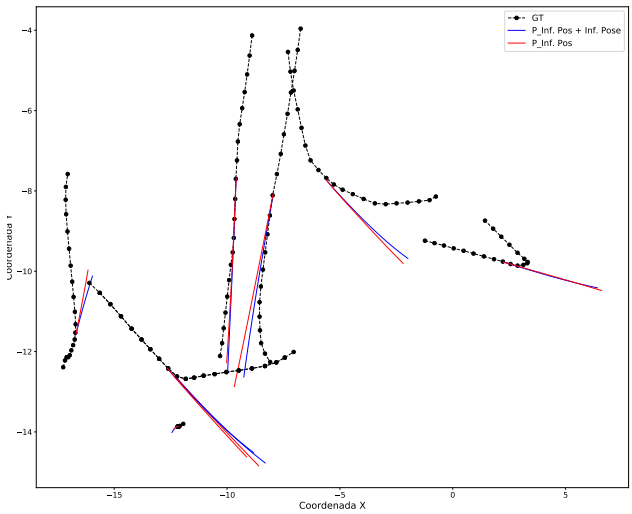


Figura 4.7: Comparación cualitativa de los modelos en el conjunto de PETS-S2L1.

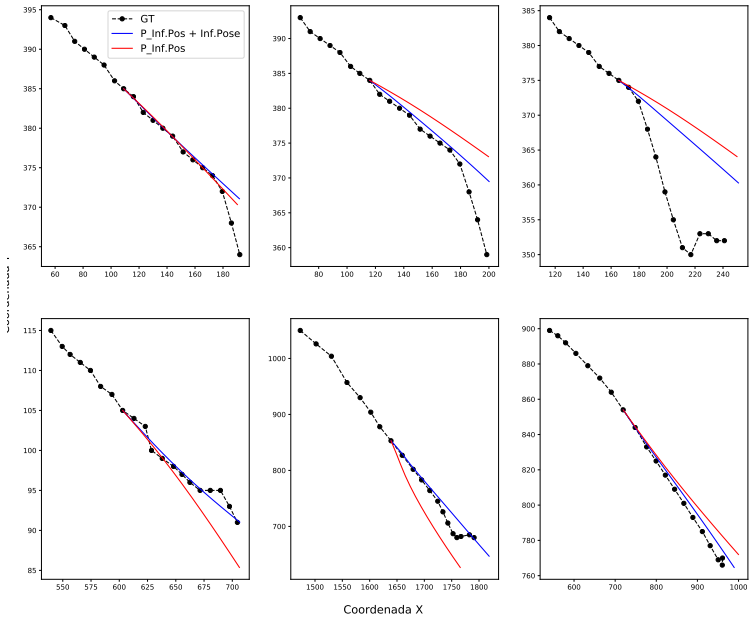


Figura 4.8: Comparación cualitativa de los modelos en el conjunto ActEV.

4.4. Conclusión

En este capítulo, se pudo comprobar que la información de pose sí ayuda al momento de hacer la predicción de trayectorias, pero cuanto sea su contribución depende también de la calidad de la adquisición de los puntos claves del esqueleto humano. Para los conjuntos de datos con los cuales se trabajó, se observó que al usar la información de puntos claves con su respectivo valor de confianza, la mejora es del 7 % aproximadamente en los errores ADE/FDE, en comparación con el modelo que sólo usa la información de las trayectorias históricas. También es importante comentar que cuando se realiza el experimento en el conjunto de datos PETS-S2L1, se puede observar que muchas de las trayectorias a predecir son curvas. En ese caso, el modelo que usa la información de pose tiende a curvar un poco más su resultado comparado con el que no usa información. No obstante, las predicciones siguen cometiendo error no despreciable. Cabe mencionar que este conjunto de datos es una grabación actuada y no natural como el de TOWN-CENTRE. Otro elemento es que la cantidad de datos de entrenamiento es relativamente baja, ya que para PETS-S2L1 el conjunto de entrenamiento consta 833 ejemplos y para el correspondiente

conjunto en TOWN-CENTRE se tiene 2,360 ejemplos. También, cualitativamente, se llega a la misma conclusión con la ayuda de las ilustraciones (Fig. 4.6, Fig. 4.7 y Fig. 4.8).

Capítulo 5

Contribución en la representación de la vecindad de los agentes

En los capítulos anteriores, nos hemos enfocado a predecir la trayectoria de agentes tomando en cuenta información individual a esos agentes, con diferentes índices (posición, keypoints) extraídos sobre los mismos agentes. No obstante, no cabe duda de que, en muchos casos, la realización de una trayectoria particular por un agente no obedece solamente a criterios individuales, sino también a elementos *contextuales* que incluyen, en particular, las trayectorias de los agentes vecinos. Un agente, por ejemplo, modificará su trayectoria para evitar colisiones con sus vecinos.

Por ello, en este capítulo, estudiaremos algunas formas de representar la influencia que tienen los demás agentes que se encuentran en el mismo entorno que el agente i y evaluaremos hasta qué punto este modelado permite mejorar la predicción de las trayectorias.

Por lo que la distribución del resto del capítulo es como se presenta a continuación. En la Sección 5.1 se da un pequeño panorama de como a sido modelado la interacción social en distintos trabajos del área, y se menciona que en nuestro caso usaremos una aproximación burda del *flujo óptico* para modelar las interacciones sociales del agente observador con su entorno. En la Sección 5.2 se define el flujo óptico tanto para imágenes como el que percibe un agente observador, y se describe como calcularlo. En seguida en la Sección 5.3 se presentan las dos propuestas del modelado de la interacción social haciendo uso del flujo óptico. Mientras que en la Sección 5.4, se

explica como agregar al modelado de la interacción social, la información de los obstáculos fijos. Posteriormente en la Sección 5.5 se describe cual es la preparación de los datos que serán usados para evaluar los modelos de interacción social. A continuación en la Sección 5.6, se ilustra y describe la arquitectura de la red, que se va a usar para evaluar los modelos de interacción social propuestos. En la Sección 5.7 se tiene cada uno de los experimentos realizados para medir si hay alguna contribución con los modelos propuestos, y finalmente en la Sección 5.8, se dan las conclusiones a las que se llegaron con los experimentos realizados a través de todo este capítulo.

5.1. Modelación de la Interacción Social

En distintos trabajos en el dominio de la navegación de agentes, se ha modelado la interacción social suponiendo que sólo los humanos en un cierto vecindario afectan el movimiento de los demás, lo que no es necesariamente cierto en escenarios reales con multitudes. Por lo tanto, proponemos dos diferentes maneras de modelar la influencia de los demás agentes. En una de las propuestas, conservamos la idea de vecindad, sólo que la refinamos para nuestros propósitos. En el otro enfoque, tomaremos en cuenta a todos los agentes que se encuentren en el campo de visión del agente observador, sin tomar en cuenta la noción de vecindad.

En este tipo de problema, como en el que estamos trabajando, se ve involucrado la presencia de múltiples movimientos independientes (peatones) y la percepción que esa genera sobre el agente considerado. Para modelar cada uno de los movimientos externos desde el punto de vista del agente de interés, y para estar lo más cercano del estímulo realmente procesado por el agente humano, proponemos utilizar técnicas de flujo óptico, que han sido utilizadas recientemente con cierto éxito para guiar la navegación de personajes virtuales [17]. La idea es reemplazar la modelación típica de la actuación de los otros agentes con posiciones relativas (lo que de cierta manera presupone un agente "omnisciente" sobre su entorno) [1, 10] por el estímulo perceptual (o una aproximación burda de éste) de los movimientos relativos entre agentes. Este estímulo es conocido en visión computacional como *flujo óptico*.

5.2. Flujo óptico

Se denomina flujo óptico al campo vectorial sobre la imagen que corresponde al movimiento ocurrido píxel a píxel entre dos o más cuadros de una secuencia de vídeo. Dicho de otra manera, da la velocidad aparente de los píxeles de la vídeo. El flujo óptico es el objeto matemático más general para describir la percepción del movimiento a través de un sensor de tipo *pin-hole* (como una cámara).

En nuestro caso, no calculamos el flujo óptico directamente de los cuadros (imágenes), ya que queremos modelar la percepción de los agentes, y no la del sensor percibiendo la escena; no obstante, contamos con las posiciones en marco mundo del agente i y con las de sus vecinos, en cada cuadro de la secuencia. Entonces, tenemos acceso a los movimientos relativos de los agentes en $3D$, quienes se traducen en la proyección de un movimiento en la imagen. Por ende, *simulamos* el flujo óptico percibido por cada agente a partir de los movimientos relativos de sus vecinos.

A continuación explicamos cómo calculamos el flujo óptico.

5.2.1. Calculo del flujo óptico

En esta subsección, se explica como calcular el flujo óptico (el movimiento aparente de los objetos en el campo visual). Un agente observador i percibe un punto $\mathbf{P}_j$, con velocidad *relativa* $\mathbf{v}_{ji}$. A las coordenadas en $3D$ (en un marco mundo) de un vecino j del agente i, las denotamos como $\mathbf{P}_j = (X_j, Y_j, Z_j)$. A sus coordenadas *relativas* (en un marco propio de la cámara del agente i), las denotamos como $\mathbf{P}_{ji} = (X_{ji}, Y_{ji}, Z_{ji})$.

Por otro lado, denotamos las *proyecciones* del agente j en la percepción visual del agente i como $\mathbf{p}_j = (x_j, y_j)$. El flujo óptico generado por el agente j en el campo visual de i lo denotamos como $\mathbf{u}_j = (u_j, v_j)$. El movimiento relativo del peatón j genera el flujo óptico del punto $\mathbf{P}_j$ en el objeto i de acuerdo a las siguientes ecuaciones:

$$\begin{cases} u_j = (v_{ji,x} - x_j v_{ji,z})/Z_{ji} \\ v_j = (v_{ji,y} - y_j v_{ji,z})/Z_{ji} \end{cases} . \tag{5.1}$$

Por la naturaleza de nuestros datos, vemos que se tiene un punto en un plano en el marco mundo, pero que no se tiene la altura del peatón. Entonces, aproximaremos

los peatones por cilindros de altura infinita, y calcularemos una aproximación al flujo óptico, es decir que razonaremos solamente en términos de un flujo 1D en la dirección horizontal de la percepción del agente i:

$$u_j = (v_{ji,x} - x_j v_{ji,z})/Z_{ji} \tag{5.2}$$

donde:

- $\mathbf{P}_{ji} = (X_{ji}, Y_{ji}, Z_{ji})$ es la posición del punto j del obstáculo i, en coordenadas mundo $3D$, *relativa* al peatón observador i (con un marco orientado por la orientación de i).
- $x_j = \dfrac{X_{ji}}{Z_{ji}}, y_j = \dfrac{Y_{ji}}{Z_{ji}}$ es la proyección perspectiva del punto $\mathbf{P}_j$ en el sistema visual de i, con una cámara "canónica", es decir con parámetros intrínsecos unitarios.
- $\mathbf{v}_{ji} = (v_{ji,x}, v_{ji,y}, v_{ji,z})$ es la velocidad del obstáculo j relativa al peatón observador i (con un marco orientado por la orientación de i).

Para simplificar la notación, y como en principio daremos siempre las explicaciones para un peatón i, se usará de aquí en adelante el subíndice j, únicamente, al referirnos al obstáculo. Ya que nosotros tenemos un único punto por cada obstáculo observado.

5.2.2. Simulación del flujo con datos de trayectorias

Los peatones que contemplemos en el *modelo de interacción social*, se les denominará vecinos del agente i, y se les calcula su contribución al flujo óptico observado por el agente i, como se describió anteriormente. El agente observador i tiene una secuencia de posiciones observadas $\mathbf{x}^i_{1:T_{obs}}$, y el peatón vecino j con $j \neq i$, también tiene una secuencia de posiciones observadas $\mathbf{x}^j_{1:T_{obs}}$ (ambos expresadas en el marco mundo).

Para evaluar la contribución del flujo óptico, en el instante de tiempo t, del peatón vecino j del agente observador i, se siguen los siguientes pasos:

- Se obtiene el vector dirección del peatón observador i, expresado en el marco mundo:

$$\mathbf{z}_t^i \triangleq (z_{x,t}^i, z_{y,t}^i) = \mathbf{x}_t^i - \mathbf{x}_{t-1}^i. \tag{5.3}$$

- Se calcula el ángulo del vector de dirección del peatón observador:

$$\theta = \arctan\left(\frac{z_{y,t}^i}{z_{x,t}^i}\right). \tag{5.4}$$

- Posteriormente, se calcula el vector desplazamiento del peatón vecino j relativo al agente observador i:

$$\Delta\mathbf{d} = \mathbf{x}_t^j - \mathbf{x}_t^i \tag{5.5}$$

$$\mathbf{P}_j = R_\theta \Delta\mathbf{d}, \tag{5.6}$$

  donde R_θ es la rotación 2D de ángulo θ, es decir:

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

- También, se estima el vector velocidad del peatón vecino j:

$$\mathbf{z}_t^j = \mathbf{x}_t^j - \mathbf{x}_{t-1}^j. \tag{5.7}$$

- Al igual, se calcula el vector de velocidad relativa entre el peatón vecino j y el agente observador i:

$$\Delta\mathbf{v} = \mathbf{z}_t^j - \mathbf{z}_t^i \tag{5.8}$$

$$\mathbf{v}_j = R_\theta \Delta\mathbf{v}. \tag{5.9}$$

Por último, se sustituye las expresiones de $\mathbf{P}_j$ y $\mathbf{v}_j$ en la siguiente ecuación, como se explicó en la subsección anterior:

$$u_j = (v_{jx} - \frac{X_j}{Z_j} v_{jz})/Z_j. \tag{5.10}$$

Veremos en la sección siguiente cómo se manejan las potenciales oclusiones entre

peatones vecinos.

5.3. Modelos de la interacción social: vecindario acotado y no acotado.

En esta sección, describimos dos variantes de nuestro modelo basado en flujo óptico.

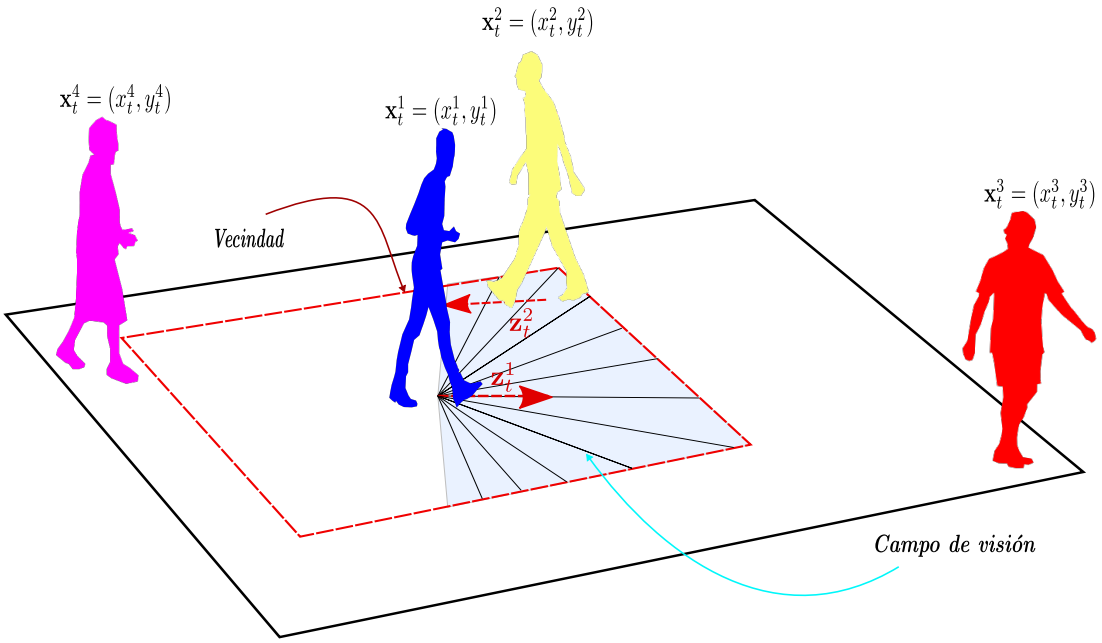


Figura 5.1: Modelo de vecindario con rango limitado.

Para la primera propuesta, al momento de modelar la interacción social, conservamos la noción de vecindad, pero la refinamos como sigue. Por ejemplo en [1], se toma una distancia alrededor del agente i en cuestión, tal que todos los peatones que se encuentren dentro de esta vecindad serán los peatones que influencian los movimientos de dicho agente, sin importar que el agente los vea o no. Nosotros también tomamos en cuenta una vecindad definida por una distancia máxima, pero sólo tomaremos a los peatones de la vecindad que sean *visibles* para el agente observador, y no a los que se encuentran fuera de su *campo de visión*. Este está definido como el área espacial (cono) que el ojo percibe mirando hacia delante (con respecto a su

vector dirección) sin necesidad de efectuar movimiento alguno. El ojo humano sano tiene la máxima resolución y nitidez en el área central de la imagen hacia donde miramos. Hacia la periferia del campo visual vemos, más bien, con poca nitidez, pero podemos percibir bien los movimientos, luces y siluetas.

Es por ello que para el agente observador i, en el tiempo t, con coordenadas $\mathbf{x}_t^i = (x_t^i, y_t^i)$, decimos que un peatón $j \neq i$, en el mismo instante de tiempo, es su *vecino*, si dicho peatón pertenece a la vecindad y al mismo tiempo se encuentra en el campo de visión (es decir dentro de un cono de visión) del peatón i, y es el más cercano al agente observador en dicho sector del cono, como lo ilustramos en la figura 5.1. Mantener el vecino más cercano en cada sector permite lidiar con los fenómenos de oclusión entre peatones. El campo de visión del agente i se construye discretizando en *n-bins* el ángulo de visión de γ grados, con respecto al vector normal del agente i. Por ende, un agente observador puede tener a lo más n vecinos percibidos. Si un peatón resulta ser *vecino* del peatón i, entonces se calcula su flujo óptico. Así, al finalizar, cada agente observador obtendrá un vector de tamaño fijo, con las contribuciones al flujo óptico de los vecinos. En la Fig. 5.5, se ilustra cómo se visualiza el flujo óptico 1D, para este caso (vecindad acotada).

Caso particular: vecindad no acotada

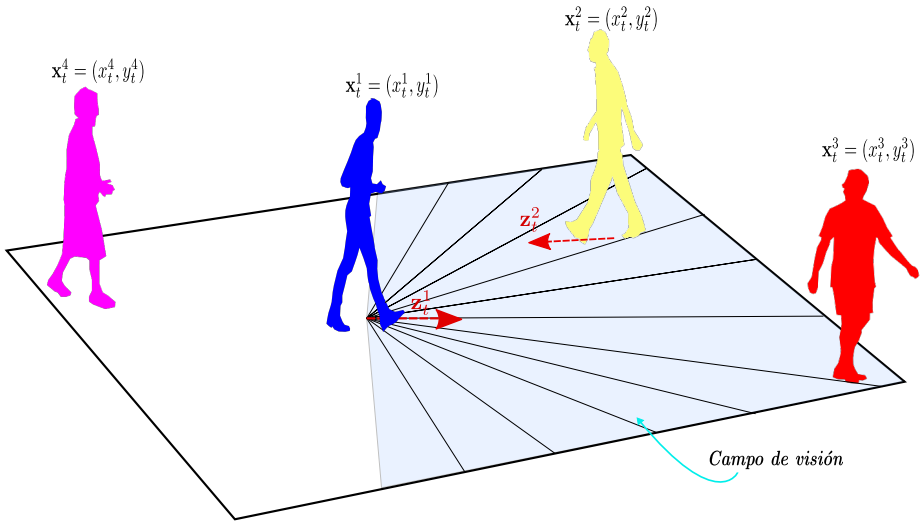


Figura 5.2: Modelo de interacción social sin vecindad.

En este segundo caso, ilustrado en la Fig. 5.2, pensamos que nuestra vecindad es no acotada, lo cual es lo mismo que no tener vecindad, así que sólo consideramos el campo de visión del agente observador i. De esa manera, un peatón j, con $j \neq i$, es un *vecino* de i, si pertenece a uno de los n sectores del campo de visión del agente observador y si es el más cercano a él en este sector del campo de visión. Si esto es así, entonces calculamos el flujo óptico del peatón j con respecto al observador i.

La Fig. 5.3 es un ejemplo de cómo se visualiza el flujo óptico en una dimensión. Este ejemplo es sin vecindad, y se observaron 8 instantes de tiempo, de los cuales se grafican los 6 intermedios. Las filas impares muestran la situación del agente observador y la de los peatones que se encuentran al rededor de él, mientras que las filas pares representa el vector de flujos ópticos asociado al agente observador. El agente observador, en un instante de tiempo t de la trayectoria observada, lo pintamos con un círculo azul, sus posibles vecinos con verde y los peatones que sí cumplieron con la definición de ser vecino (i.e. estar visibles desde el agente i) se pintan con un circulo rojo. Las gráficas de barras representan el valor del flujo óptico generado por el vecino en el campo de visión del agente observador, donde el eje x representa los ángulos (en radianes) en los que se pueden encontrar vecinos con respecto al vector normal del agente observador (vector en rojo), y se leen de izquierda a derecha las correspondencias entre los vecinos y su respectiva barra de flujo óptico, es decir que la primera barra corresponde al vecino que se encuentra más a la izquierda del agente observador. En este ejemplo se puede observar que los valores de los flujos ópticos de los dos vecinos que se encuentran casi en frente del agente observador son casi cero, esto es por que las velocidades de los vecinos son similares a la del observador.

5.4. Incorporación de mapas semánticos

En esta sección, se presentan brevemente los mapas que proponemos usar para completar la información de vecinos que hemos descrito anteriormente con la de obstáculos fijos, que un agente observador i puede considerar para adaptar su trayectoria. Lo interesante es que esta información se puede agregar de la misma manera a través de flujo óptico, considerando que edificios, mobiliario urbano, peatones son diferentes tipos de obstáculos, algunos fijos y otros móviles.

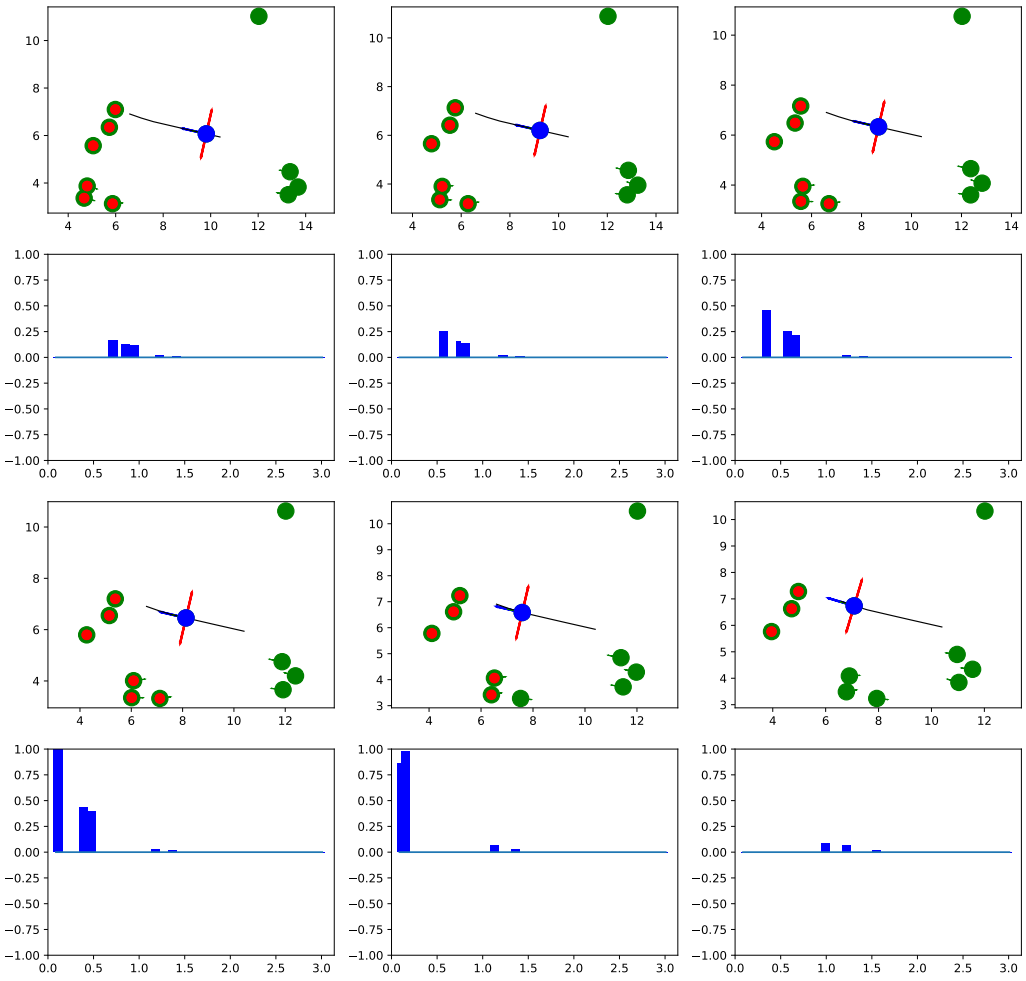


Figura 5.3: Flujo óptico: un ejemplo de la secuencia Zara01. Filas impares: situación del agente (en azul) y de los peatones que se encuentran alrededor de él (en verde). Los peatones en rojo son los que están visibles por el agente (vecinos del agente). Filas pares: visualización del flujo 1D. Para una explicación más amplia de lo que representa dicha ilustración ver Pag. 104.

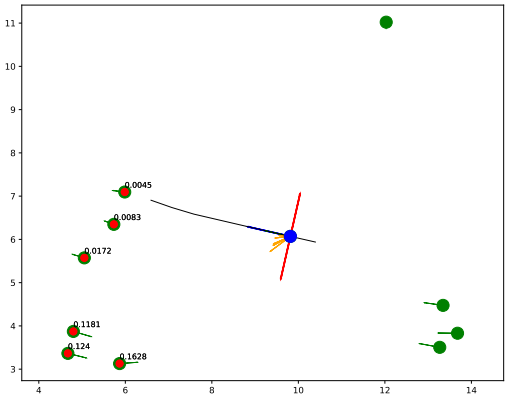


Figura 5.4: Esta ilustración agrega la información de las subgráficas (1,1) y (2,1) de la Fig. 5.3 con la posición del agente observador (en azul) y de los peatones a su alrededor (en verde). Los peatones en rojo son los que están visibles por el agente, y las cantidades ilustradas arriba de los puntos rojos son los valores del flujo óptico en 1D percibido por el agente observador (las barras azueles de la figura 5.3). Estos valores de flujo son también ilustrados con los vectores en naranja.

En los mapas semánticos ilustrados en la Fig. 5.6, las partes que están en blanco representan calles. En rojo son lugares donde sólo pueden transitar peatones (pasto, banquetas), en negro son obstáculos fijos (edificios, autos estacionados, etc.) y en amarillo se definen los bordes de los obstáculos fijos. Esos últimos son los obstáculos con los cuales se trabajará.

De manera análoga a la información de los vecinos, se procesa la información de los obstáculos fijos. La única diferencia es que la velocidad de estos obstáculos es cero.

5.5. Preparación de los datos

En este enfoque, como ya lo hemos comentado, para cada agente cuya trayectoria se quiere predecir, sí se toma en cuenta a los demás peatones que se encuentran en su entorno, en el mismo instante de tiempo t.

El entrenamiento del modelo se hace de la siguiente manera: se considera un *número máximo* de personas para todas las sucesiones de frames de una longitud determinada (T_{pred}). Esto se hace para tomar en cuenta a las personas que se en-

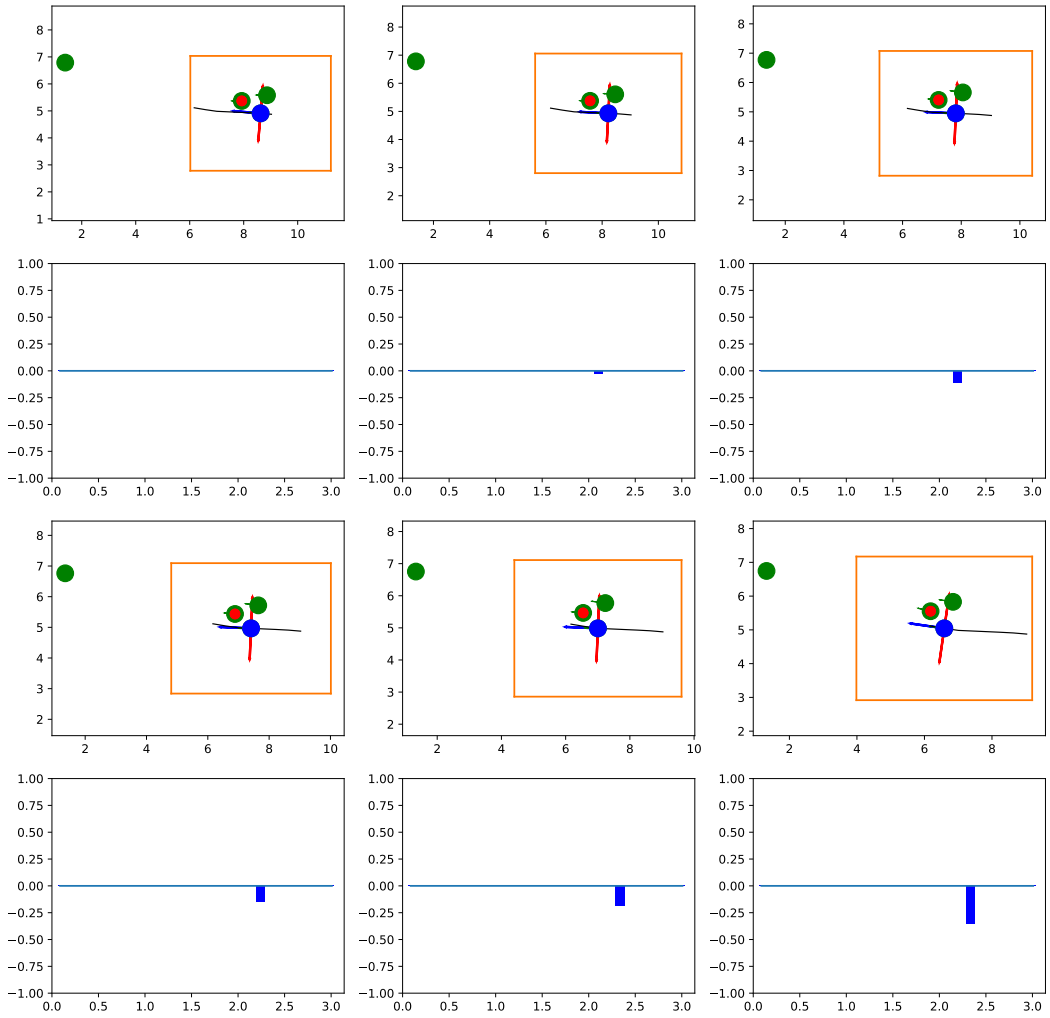
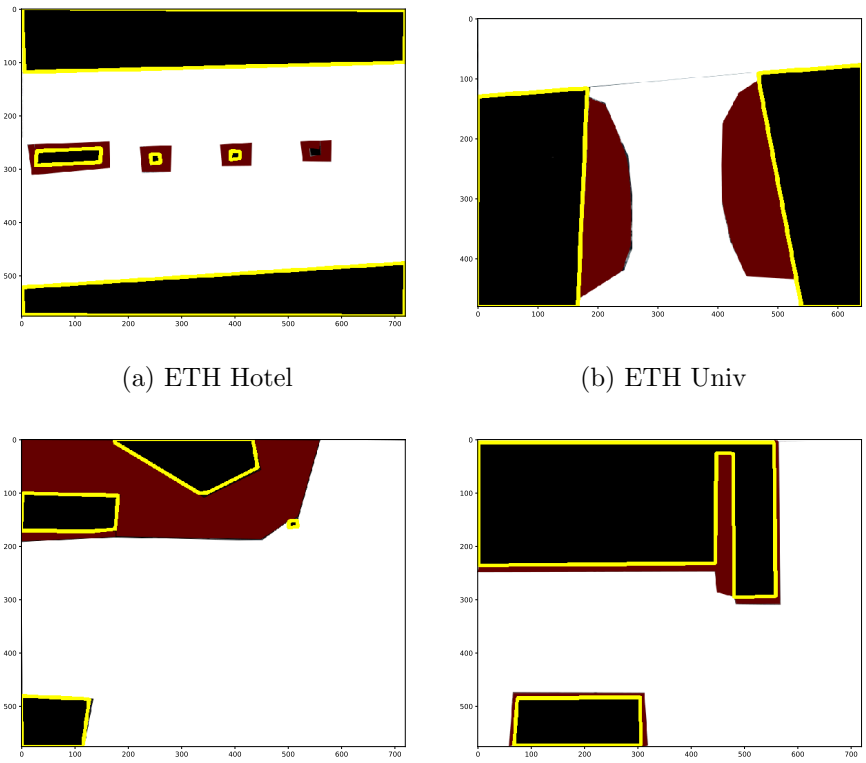


Figura 5.5: Flujo óptico con vecindario limitado: un ejemplo de la secuencia Zara01. Filas impares: situación del agente (en azul) y de sus vecinos (en verde). Los vecinos rojos son los que están visibles por el agente. Filas pares: visualización del flujo 1D.



(a) ETH Hotel

(b) ETH Univ

(c) UCY Univ

(d) Crowds Zara01

Figura 5.6: Mapas de obstáculos.

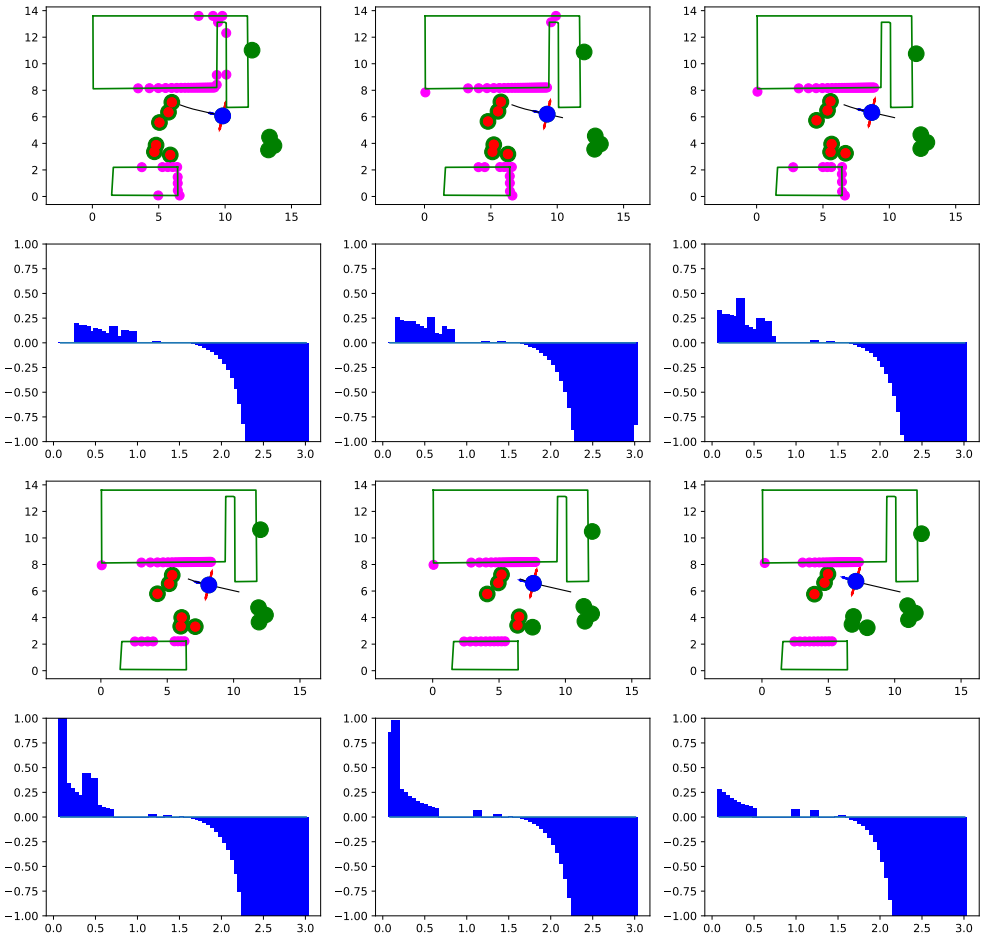


Figura 5.7: Flujo óptico: un ejemplo de la secuencia Zara01, incluyendo a obstáculos. Filas impares: situación del agente (en azul) y de sus vecinos (en verde). Los vecinos rojos son los que están visibles por el agente y en magenta los obstáculos fijos visibles. Filas pares: visualización del flujo 1D.

cuentran alrededor del peatón observador i durante la parte observada, por lo que se conserva la información de estos peatones j.

Para procesar los peatones j que se encuentran alrededor del peatón observador i, en el instante de tiempo t, definimos dos formas posibles:

- **MNP-INT**: En este caso usamos el número máximo de peatones que permanecen presentes en todo instante de tiempo en una secuencia de frames completa (es decir, con una duración igual a la longitud observada más la longitud predicha). Dicho de otra manera, se toma la intersección del conjunto de peatones que hay en una secuencia de frames de tamaño (T_{pred}).
- **MNP**: En este caso, usamos el número máximo de peatones únicos que puede haber en una secuencia completa de frames (T_{pred}). Es decir, tomamos la unión de las personas únicas de toda la secuencia.

A continuación se tiene la Tabla 5.1, donde se encuentra el *número máximo* de peatones en cada modalidad, para cada uno de los conjuntos de datos, y en la cuarta columna, se encuentran las dimensiones totales (en metros), que abarcar las trayectorias del conjunto de datos correspondiente.

Conjunto de datos	**MNP**	**MNP-INT**	**Ancho×Alto**
ETH Hotel	28	8	7.67×14.57
ETH Univ	42	5	22.11×16.38
Crowds Zara01	22	14	15.62×12.76
Crowds Zara02	26	14	15.92×14.22
UCY Univ	68	40	15.61×14.08
PETS2009 S2L1	9	8	25.06×27.03
TownCentre	38	16	36.33×19.64

Tabla 5.1: Valores máximos de los números de vecinos.

5.6. Arquitectura del modelo

De manera análoga al modelo presentado en el capítulo 4, a la red se le proporciona como entrada las sucesiones de desplazamientos (posiciones y keypoints). En

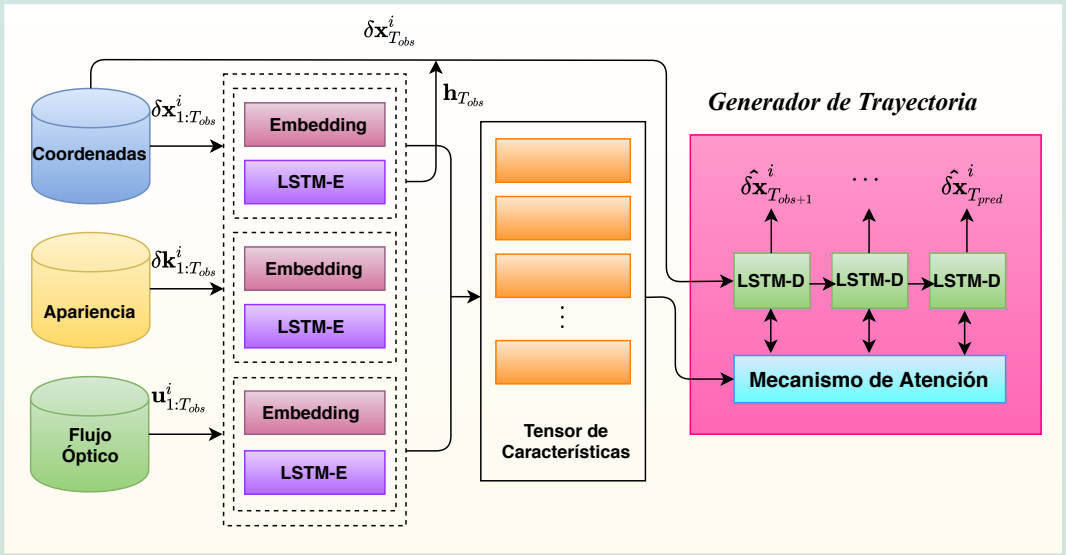


Figura 5.8: Arquitectura del modelo con interacción social.

este caso, también se le proporciona las sucesiones de vectores que representan la información del flujo óptico en 1D. Posteriormente, se codifica cada información por medio de una capa lineal y de una capa LSTM, y con la lista de estados ocultos obtenidos de la codificación de cada información, se construye el tensor de características que, en este caso, cuenta con 3 tipos de información. En la parte final del diagrama presentado en la Fig. 5.8, se encuentra el decodificador, que está señalado de color rosa, el cual es el encargado de estimar los siguientes desplazamientos (posiciones), después de los observados.

5.7. Evaluación

Para evaluar la propuesta de una nueva forma de codificar la información de los vecinos, hacemos un pre-procesamiento a los datos al quitar los ejemplos donde el observador no se mueve. Esto es por que, cuando no hay movimiento, no hay manera de calcular su vector dirección en cada instante de tiempo, ya que para eso se usa la información de sus posiciones históricas.

A continuación, en la Figs. 5.9 y 5.10, se presentan algunos ejemplos de la natu-

raleza de las trayectorias que se quieren aprender.

5.7.1. Experimento de vecindario con rango limitado

En esta subsección de experimentos nos referimos al modelo de interacción social ilustrado en la Fig. 5.1. El objetivo de este modelado es evaluar si realmente sólo los peatones cercanos influyen en la toma de decisiones del peatón observador como suponen en (*Social LSTM*) de Alahi et al. [1], o ver en qué casos es mejor sólo tomar en cuenta una vecindad (datos con una alta densidad).

Para la construcción de la *vecindad*, se encuentran los valores mínimos y máximos de las posiciones x y y para cada conjunto de datos. Así calculamos el largo y ancho (L y A respectivamente) del sector total donde se encuentran todos los peatones, ver cuarta columna de la Tabla 5.1. Dado el factor l, el cual representa en cuantas partes se va a dividir el largo y ancho total, $\frac{L}{l}$ es el largo y $\frac{A}{l}$ el ancho que tendrá la vecindad. Por ende, 0.5 es valor mínimo que puede tomar el factor l porque, en el peor de los casos, el peatón observador se encuentra en un borde del sector total y, como la vecindad está centrada en él, si tomamos $l = 1$, obtendríamos que el ancho total de la vecindad es L, pero como es centrada, en cada lado del peatón observador tendría L/2.

Para los experimentos de la Tabla 5.2, la división de los datos se hace con el enfoque de LOO, detallado en la Sección 2.6, y el número máximo de vecinos, con el enfoque MNP-INT (ver más arriba en la Tabla 5.1). Mientras tanto, los resultados de la Tabla 5.3 se realizaron con los enfoques PRO, el cual es explicado en la Sección 2.6 y MNP-INT.

En base a las dimensiones del sector total que ocupan las trayectorias de cada uno de los conjuntos (observar en la Tabla 5.1), se busca el valor que debe tomar el factor l, para que la vecindad tenga en promedio 4.5 metros de ancho y largo. Hay que mencionar que las vecindades son centradas en el peatón observador y no son necesariamente cuadradas. Por lo tanto, los valores que toma l para cada conjunto de datos, en el orden en el cual se presentan en la Tabla 5.1 son: 2, 4, 3, 3, 3, 5, 5.

De la Tabla 5.2, se puede concluir que, en este caso donde se entrena y valida con 4 conjuntos de datos y se prueba con el sobrante, la forma de modelar la interacción social por medio del flujo óptico, le perjudica aproximadamente por 1.7 por ciento

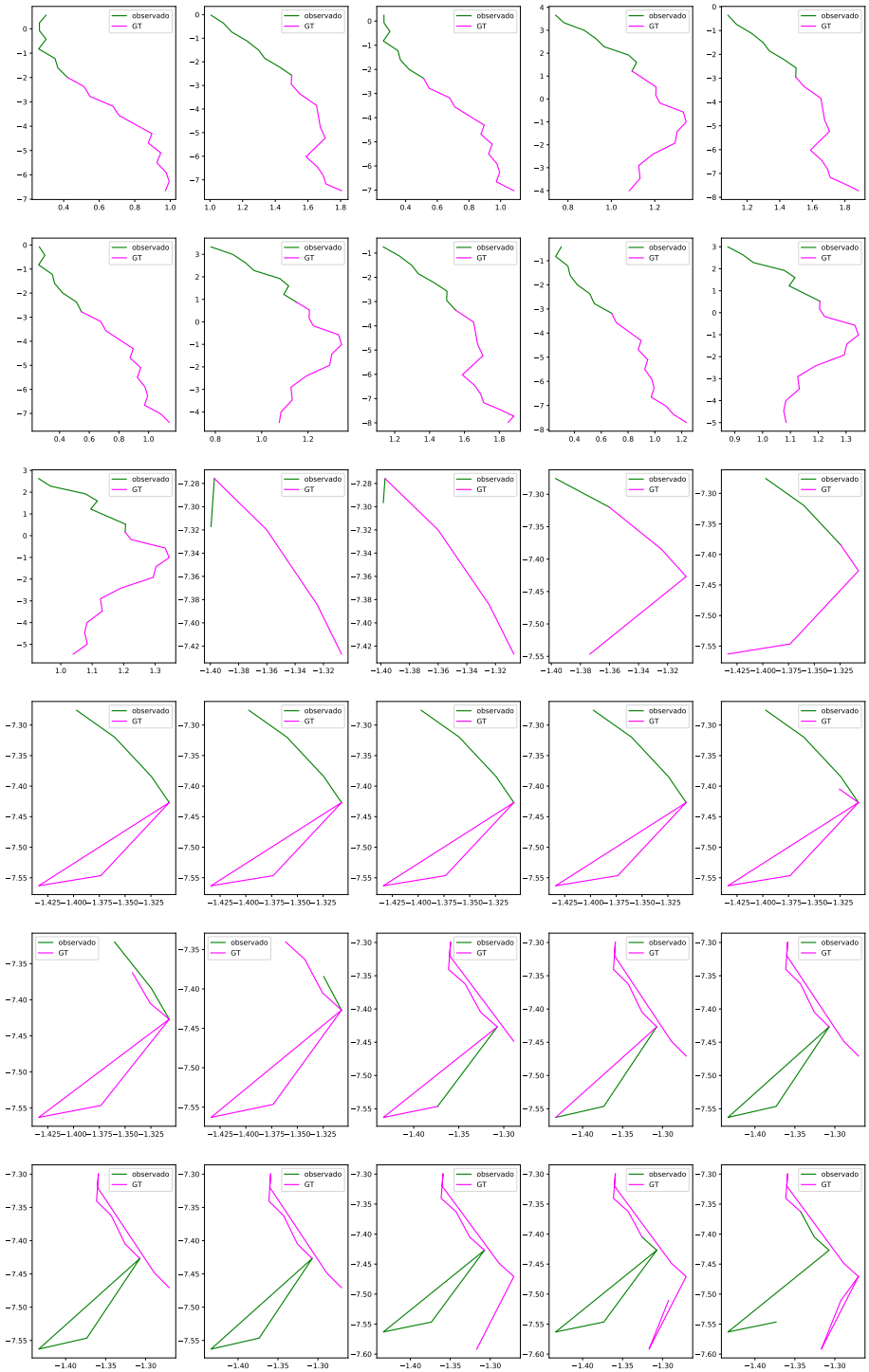


Figura 5.9: Ejemplos de algunas trayectorias en marco mundo, para diferentes peatones del conjunto de datos ETH Hotel, donde lo sombreado en verde son los 8 pasos que se observan y los de magenta son los 12 pasos futuros que se quieren que el modelo aprenda.

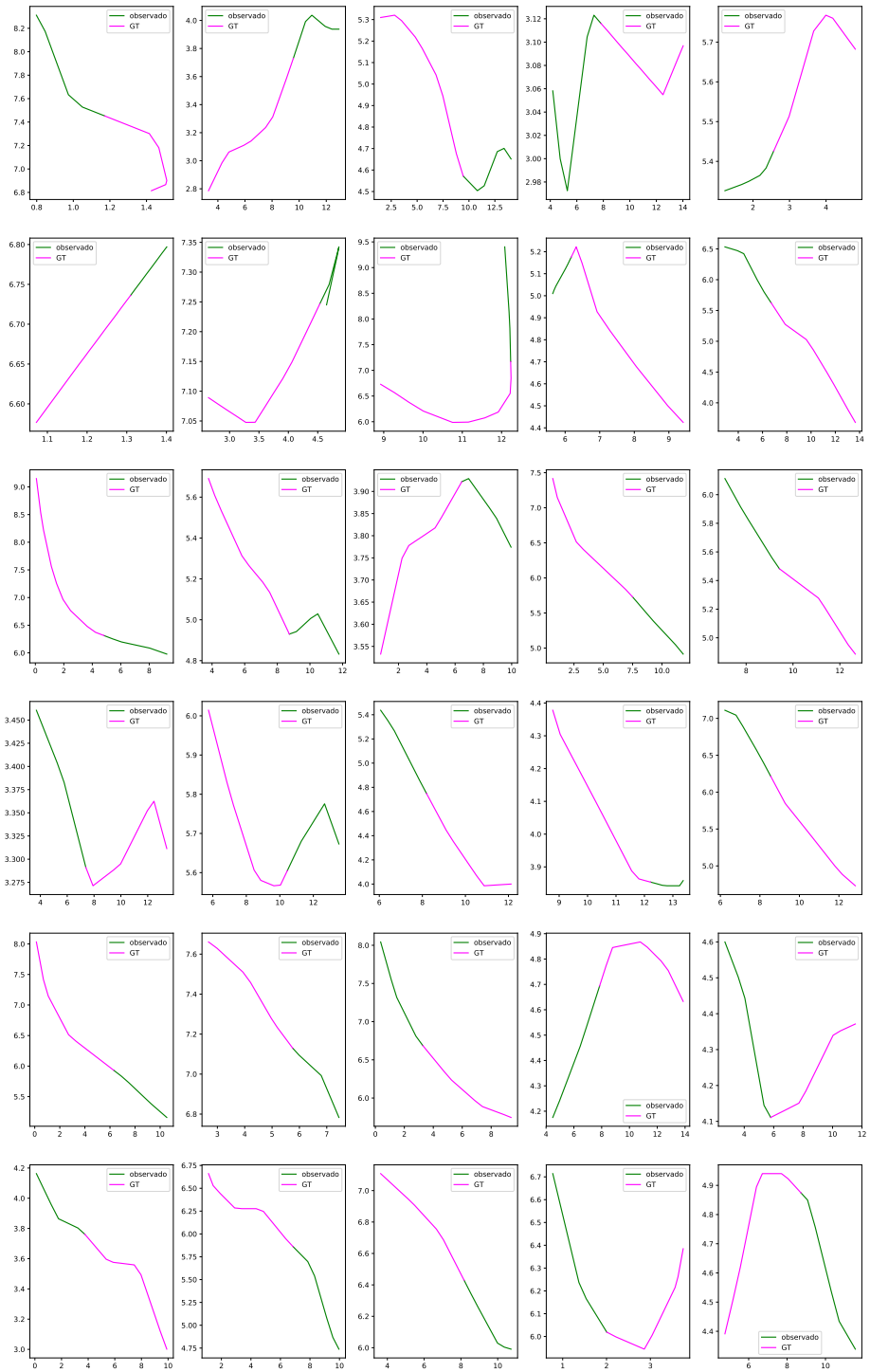


Figura 5.10: Ejemplos de trayectorias en marco mundo, de algunos peatones del conjunto de datos Zara01, donde lo sombreado en verde son los 8 pasos que se observan y los de magenta son los siguientes 12 pasos que se quieren aprender.

	Basados en LSTM	
	Inf.Pos	Inf.Pos+Soc.FO-V
ETH	**1.09**/2.29	1.09/**2.24**
HOTEL	**0.51/0.99**	0.53/1.03
UCY	**0.62/1.34**	0.63/1.37
ZARA01	**0.41/0.90**	0.41/0.90
ZARA02	**0.32/0.71**	0.32/0.72
AVG	**0.59/1.25**	0.60/1.25

Tabla 5.2: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con el enfoque LOO en la división de los datos y MNP-INT para el número máximo de personas, para las configuraciones de sólo posiciones, y la que además tiene el flujo óptico de sus vecinos en una vecindad finita.

en el promedio del ADE, al momento de la predicción.

En las Tablas 5.3 y 5.4, los modelos se prueban con una parte de los conjuntos de datos con los cuales también se entrenan y validan (enfoque PRO). Para este enfoque, se observa que, en promedio, al usar el flujo óptico para modelar la interacción que hay entre el agente observador y sus vecinos, se encuentra que hay una leve mejora, y, además, cuando a este enfoque le añadimos la información de pose, se encuentra que en promedio hay otra mejora.

Evaluación cualitativa

En las gráficas 5.11, 5.15, 5.16, se ilustran los resultados de las predicciones de los modelos que tienen, a parte de la información de posición, la información de flujo óptico. En este caso, hacemos la comparación entre el flujo óptico que está restringido a una vecindad y el que no usa vecindad. Estos modelos son evaluados en distintos conjuntos de datos.

5.7.2. Experimento sin vecindario

Para estos experimentos, se usa el modelo de interacción social ilustrado en la Fig. 5.2. En este caso hacemos una comparación entre el modelo que sólo fue entre-

	Basados en LSTM		
	Inf.Pos	Inf.Pos+Soc.FO-V	Inf.Pos+Inf.Pose+Soc.FO-V
PETS-S2L1	1.00/2.30	0.99/2.26	**0.91/2.05**
TOWN-CENTRE	0.66/1.24	0.67/1.26	**0.64/1.20**
PETS/TOWN-CENTRE	0.77/1.58	0.77/1.54	**0.75/1.52**
AVG	0.81/1.71	0.81/1.69	**0.77/1.59**

Tabla 5.3: Predicción de 12 posiciones futuras: Errores en metros(ADE/FDE), con el enfoque PRO en la división de los datos y en el número máximo de personas MNP-INT, para las configuraciones de sólo posiciones (segunda columna), posiciones y flujo óptico de sus vecinos en una vecindad finita (tercera columna), y por último con la que usa los tres tipos de información (cuarta columna).

	Basados en LSTM		
	Inf.Pos	Inf.Pos+Soc.FO-V	Inf.Pos+Inf.Pose+Soc.FO-V
PETS-S2L1	1.00/2.30	0.98/2.24	**0.91/2.04**
TOWN-CENTRE	0.66/1.24	0.67/1.26	**0.64/1.19**
PETS/TOWN-CENTRE	0.77/1.58	0.77/1.55	**0.74/1.51**
AVG	0.81/1.71	0.81/1.68	**0.76/1.58**

Tabla 5.4: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con el enfoque PRO en la división de los datos y MNP para el número máximo de personas, para las configuraciones de sólo posiciones (segunda columna), posiciones y flujo óptico de los vecinos en una vecindad finita (tercera columna) y por último posiciones, pose y flujo óptico de los vecinos (cuarta columna).

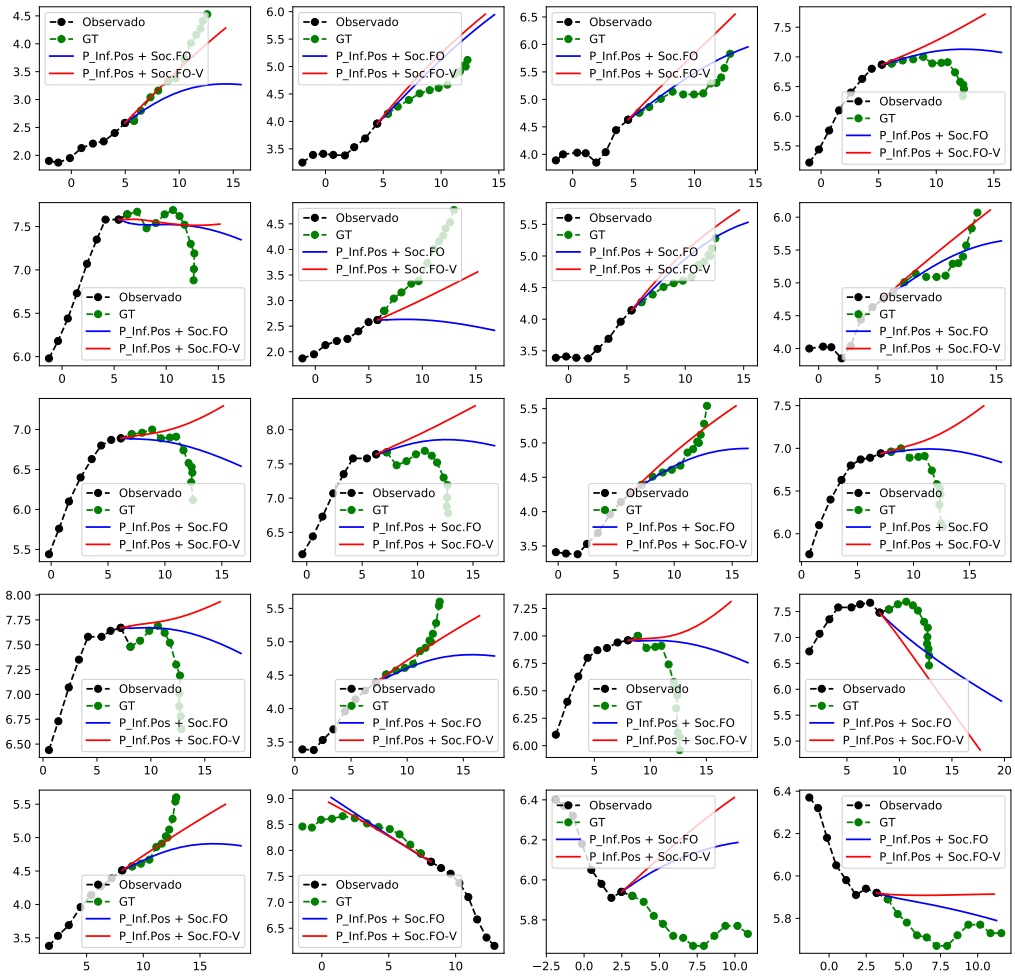


Figura 5.11: Ejemplos de predicciones en coordenadas mundo en ETH-Univ al entrenar el modelo con el enfoque LOO. Los 8 pasos observados aparecen con puntos negros, los 12 pasos verdaderos a predecir con puntos verdes, la predicción del modelo con configuración de posiciones y el flujo óptico de los vecinos en un vecindario finito en rojo, mientras que la predicción de la configuración con posiciones y flujo óptico de los vecinos sin tener un vecindario aparece en azul.

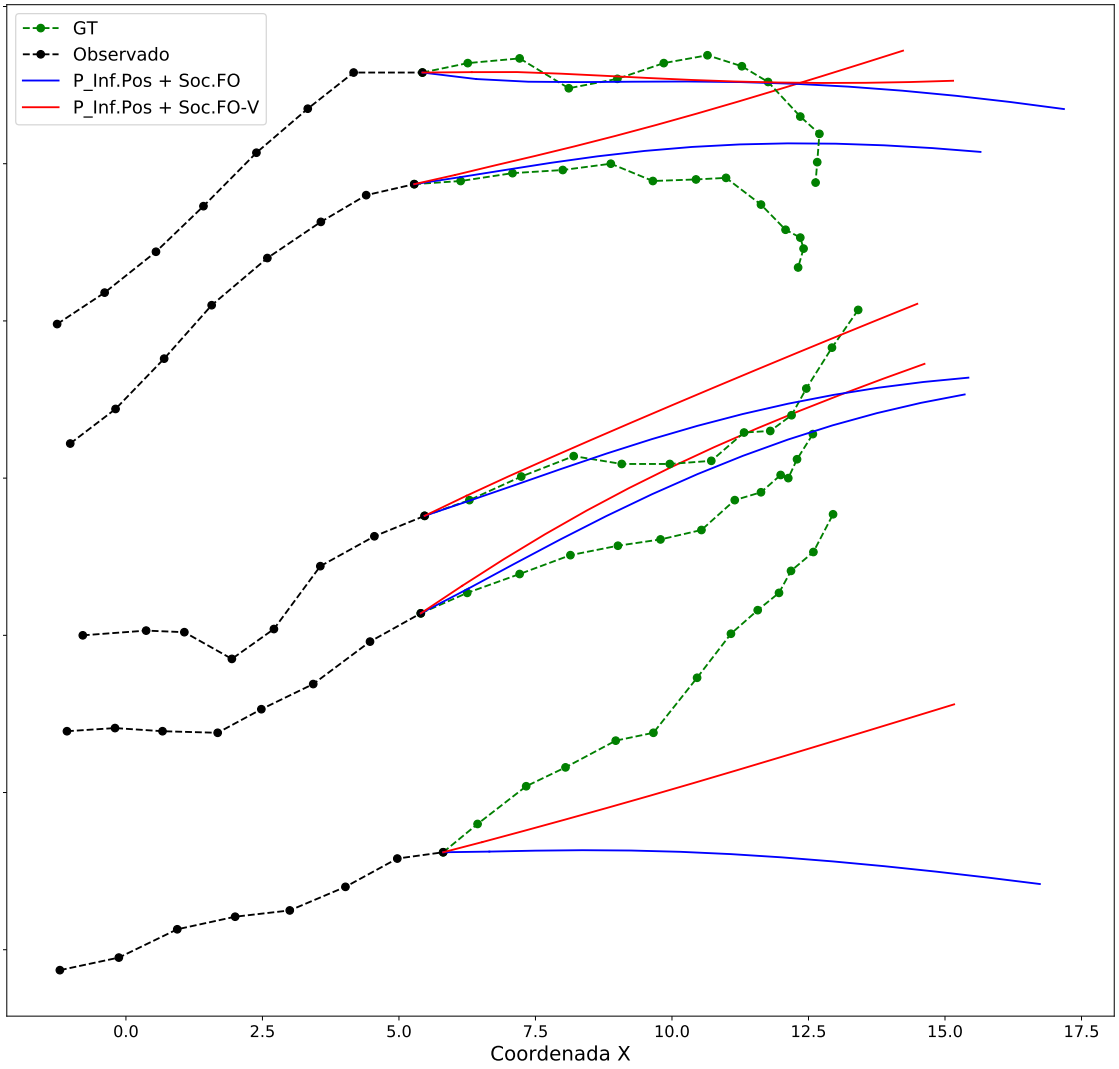
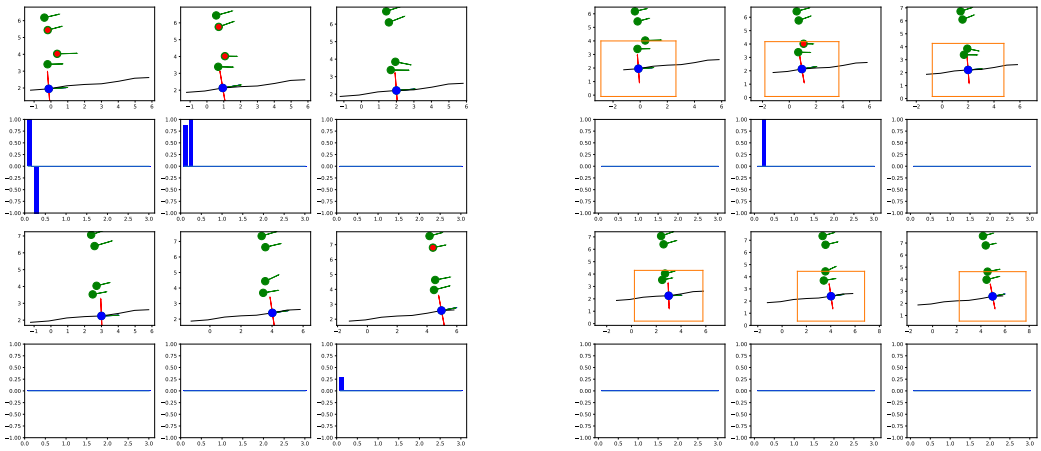


Figura 5.12: Predicciones en coordenadas mundo de 5 peatones del conjunto de datos ETH-Univ, que se encuentran en el mismo entorno. Con puntos negros se ilustra los 8 pasos observados de cada uno de los peatones, con puntos verdes los 12 pasos verdaderos a aprender, mientras que la predicción del modelo que usa posiciones y flujo óptico de los vecinos sin tomar una vecindad está en azul. En rojo aparecen las predicciones del modelo que usa posiciones y el flujo óptico de los vecinos que pertenecen a una vecindad finita.



(a) (b)

Figura 5.13: Estos dos flujos ópticos son del peatón que se encuentra más abajo en la Fig. 5.12. La figura (a) es sin tener vecindad y la (b), es con una vecindad de tamaño 5.5×4.1 metros en ancho y alto respectivamente.

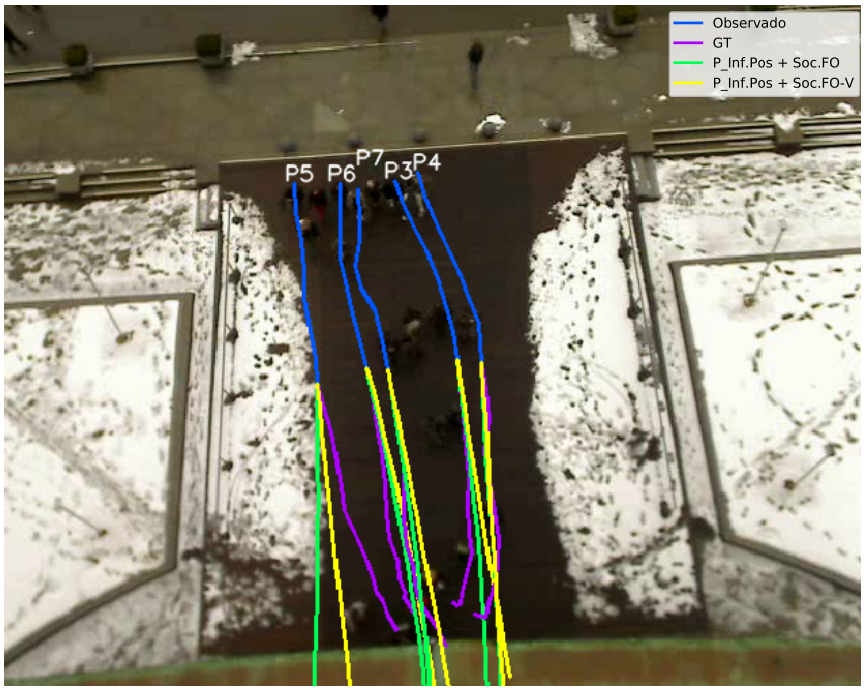


Figura 5.14: Representación de la Fig. 5.12, en coordenadas imagen.

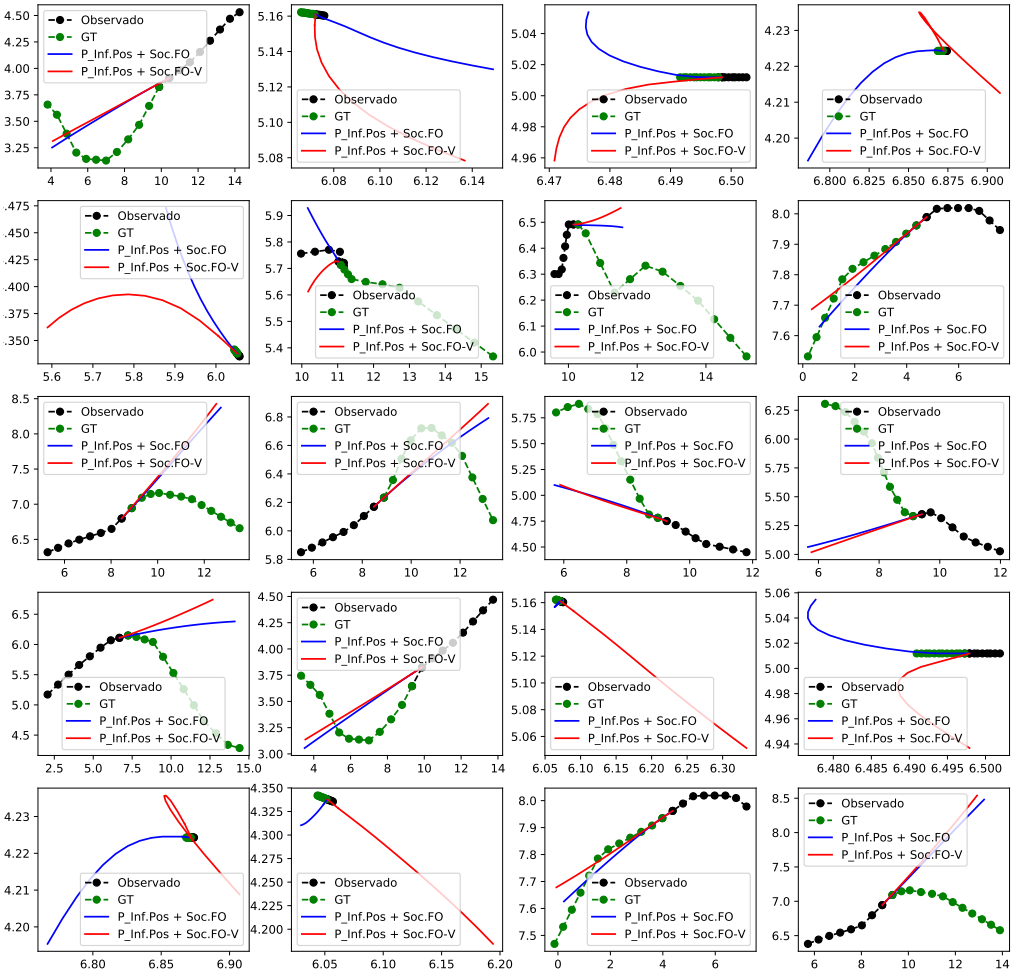


Figura 5.15: Ejemplos de predicciones en coordenadas mundo en Crowds-Zara02 al entrenar el modelo con el enfoque LOO. Los 8 pasos observados están con puntos negros, los 12 pasos verdaderos a predecir con puntos verdes, la predicción del modelo con configuración posiciones y flujo óptico de los vecinos en un vecindario finito aparece en rojo, mientras que la predicción de la configuración posiciones y flujo óptico de los vecinos sin tener un vecindario está en azul.

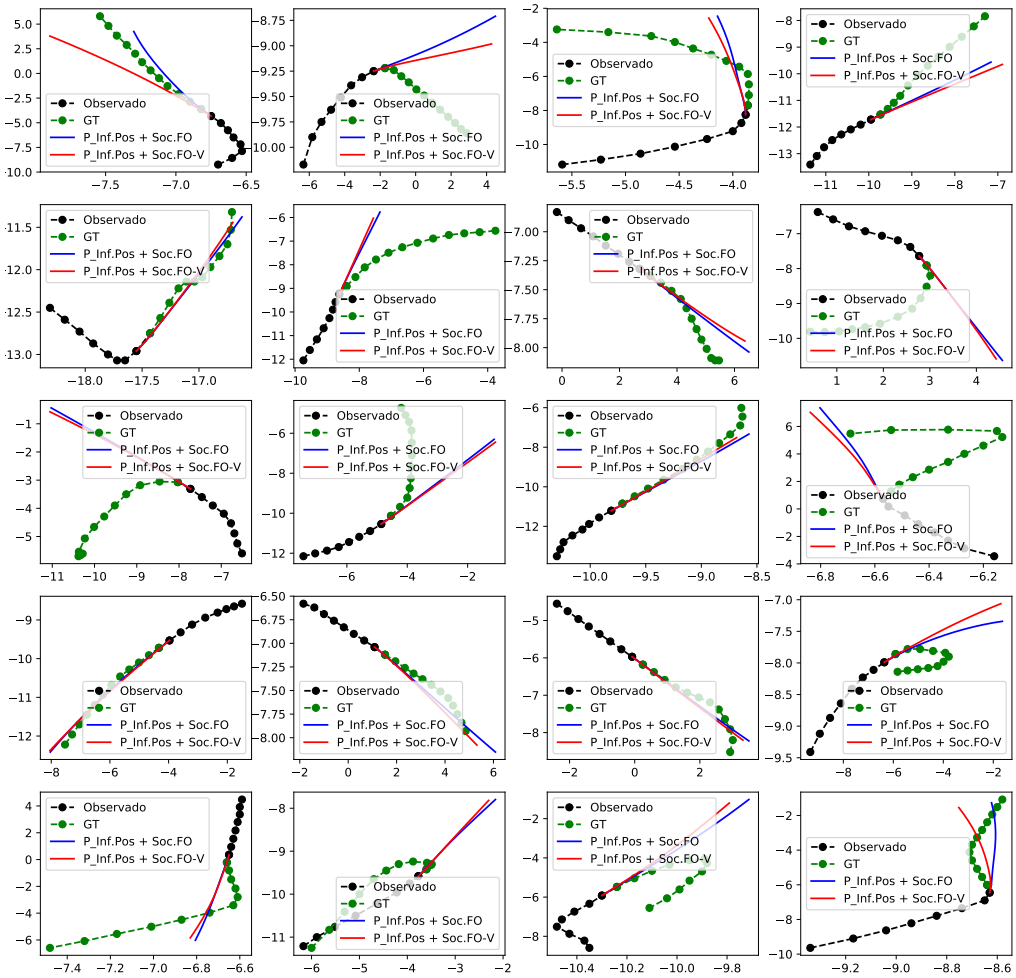


Figura 5.16: Ejemplos de predicciones en coordenadas mundo en PETS2009-S2L1 al entrenar el modelo con el enfoque PRO. Los 8 pasos observados aparecen con puntos negros, los 12 pasos verdaderos a predecir con puntos verdes, la predicción del modelo con configuración de posiciones y flujo óptico de los vecinos en un vecindario finito aparecen en rojo, mientras que la predicción de la configuración que tiene posiciones y flujo óptico de los vecinos sin tener una vecindad está en azul.

	Basados en LSTM	
	Inf.Pos	Inf.Pos+Soc.FO
ETH	**1.09/2.29**	1.10/2.33
HOTEL	0.51/0.99	**0.50/0.96**
UCY	**0.62/1.34**	0.64/1.40
ZARA01	**0.41**/0.90	0.41/**0.89**
ZARA02	**0.32/0.71**	0.32/0.71
AVG	**0.59/1.25**	0.59/1.26

Tabla 5.5: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con el enfoque LOO en la división de los datos y MNP-INT para el número máximo de personas, para las configuraciones de solo posiciones (segunda columna), posiciones y flujo óptico de los vecinos sin un vecindario (tercera columna).

nado con los datos de las coordenadas, el modelo que además tiene la información del contexto social del agente observador y para los conjuntos de datos que cuentan con una suficiente resolución también se evalúa el modelo que cuenta con los tres tipos de información (*posición, flujo óptico y pose*).

De la Tabla 5.5, se concluye que al agregarle el flujo óptico de los vecinos sin tomar en cuenta una vecindad a la información de posiciones, las predicciones se ven afectadas en menos del 1 por ciento en el promedio del FDE.

Los resultados de los experimentos plasmados en la Tabla 5.6 fueron realizados con los enfoques PRO y MNP-INT. Se puede apreciar que hay una mejora de un poco más del uno por ciento en el promedio del FDE. Mientras que al agregarle una tercera información que es la pose se encuentra que hay una mejora tanto en el promedio ADE, FDE del 5 y 7 por ciento respectivamente con respecto al modelo que sólo cuenta con la información de posición.

Los experimentos de la Tabla 5.7, fueron realizados con los enfoques PRO y MNP, y se puede apreciar que al añadir el flujo óptico a la información de entrada del modelo, se obtiene una mejora de poco más del cuatro por ciento en el promedio de ADE y poco más del uno por ciento en el promedio del FDE, mientras que cuando agregamos un tercer tipo de información (*pose*), hay una mejora del 6 y poco más

	Basados en LSTM		
	Inf.Pos	Inf.Pos+Soc.FO	Inf.Pos+Inf.Pose+Soc.FO
PETS-S2L1	1.00/2.30	0.98/2.24	**0.91/ 2.07**
TOWN-CENTRE	0.66/1.24	0.68/1.27	**0.65/1.21**
PETS/TOWN-CENTRE	0.77/1.58	0.78/1.57	**0.74/1.50**
AVG	0.81/1.71	0.81/1.69	**0.77/1.59**

Tabla 5.6: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con el enfoque PRO para la división de los datos y MNP-INT para el número máximo de personas, para las configuraciones solo posiciones (segunda columna), posiciones y flujo óptico de los vecinos sin un vecindario (tercera columna), y como última configuración posiciones, flujo óptico de los vecinos sin vecindario y pose (cuarta columna).

	Basados en LSTM		
	Inf.Pos	Inf.Pos+Soc.FO	Inf.Pos+Inf.Pose+Soc.FO
PETS-S2L1	1.00/2.30	0.96/2.19	**0.89/1.99**
TOWN-CENTRE	0.66/1.24	0.65/1.23	**0.64/1.21**
PETS/TOWN-CENTRE	0.77/1.58	**0.75/1.51**	0.75/1.51
AVG	0.81/1.71	0.79/1.64	**0.76/1.57**

Tabla 5.7: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), con el enfoque PRO en la división de los datos y MNP como el número máximo de personas, para las configuraciones de sólo posiciones (segunda columna), posiciones y flujo óptico de los vecinos sin vecindad (tercera columna), y como última configuración posiciones, flujo óptico de los vecinos y la pose (cuarta columna).

del 8 por ciento respectivamente en el promedio de ADE y FDE.

Evaluación cualitativa

En las imágenes (Figs. 5.17, 5.18, 5.19), se aprecian algunos ejemplos de predicciones en distintos conjuntos de datos, donde lo que hacemos es comparar los resultados de predecir sólo con la información de posiciones y la que tiene además el flujo óptico que generan los vecinos del agente observador, en el caso cuando no se tiene un vecindario.

5.7.3. Experimento sin vecindario y con mapa de obstáculos fijos.

Para estos experimentos, sólo trabajaremos con los conjuntos de datos de ETH/UCY, ya que son los que cuentan con un mapa semántico. Dichos mapas se ilustraron en la Fig. 5.6.

Los siguientes experimentos, donde se incluye el flujo óptico con obstáculos fijos, se realizó con los enfoques de LOO y MNP-INT.

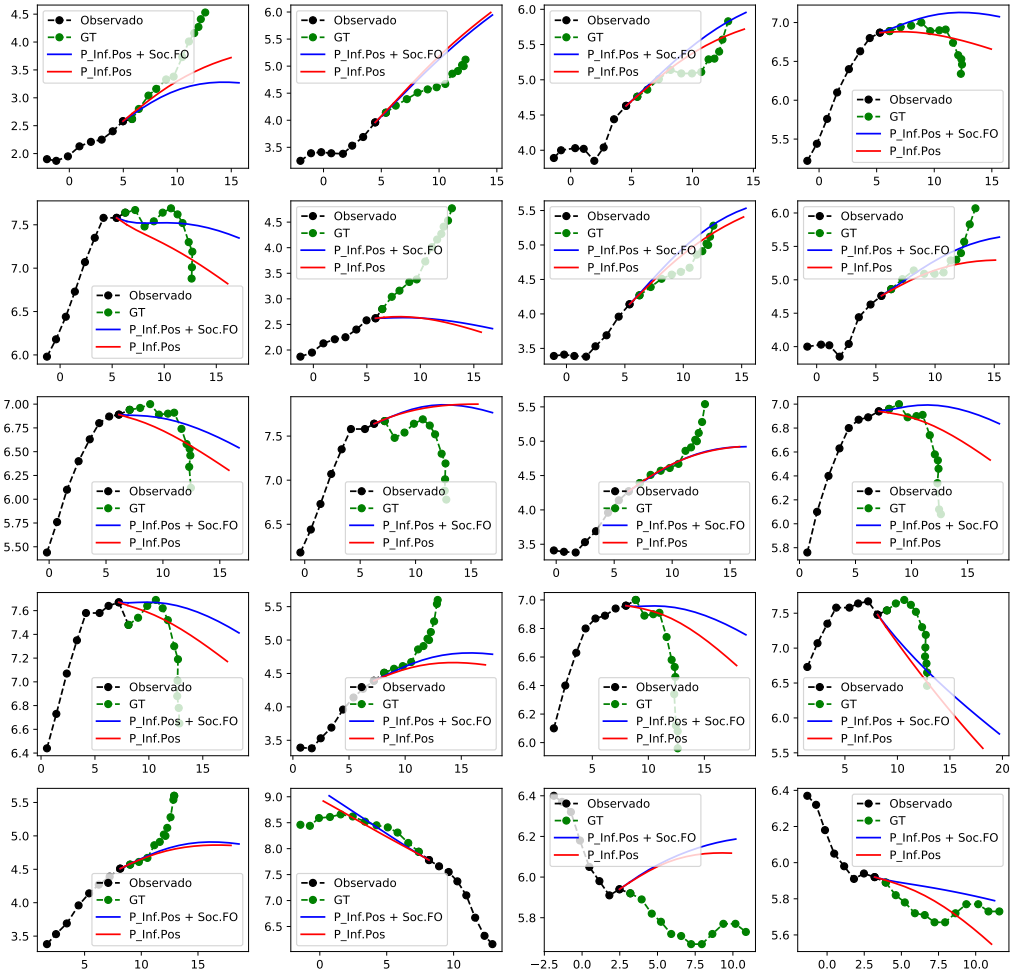


Figura 5.17: Ejemplos de predicciones en ETH-Univ al entrenar el modelo con el enfoque LOO. La secuencia observada se ilustra con puntos negros, la trayectoria verdadera a predecir con puntos verdes, mientras que la predicción del modelo con configuración de sólo posiciones está en rojo y la predicción del modelo que usa posiciones y el flujo óptico de los vecinos sin vecindario en azul.

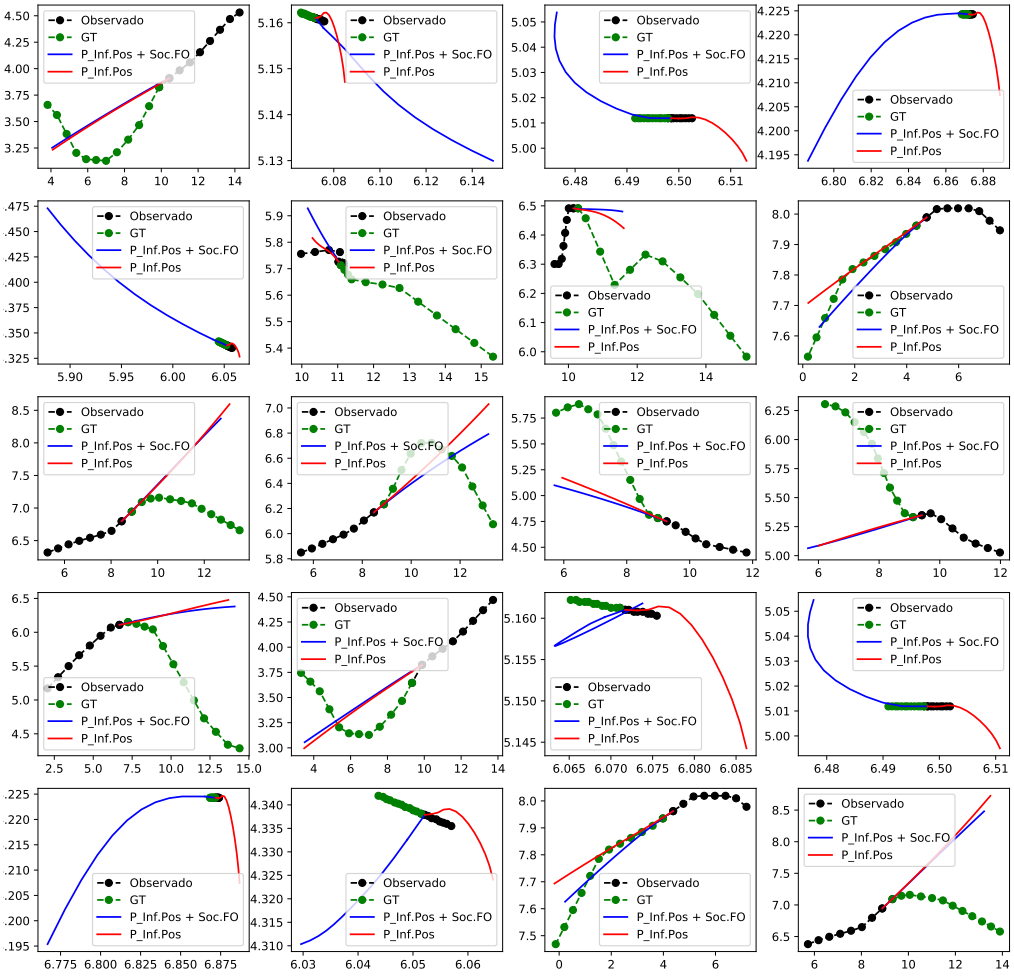


Figura 5.18: Ejemplos de predicciones en coordenadas mundo en Crowds-Zara02 al entrenar el modelo con el enfoque LOO. Las 8 posiciones observadas están en puntos negros, la trayectoria verdadera a predecir en puntos verdes, mientras que la predicción del modelo que usa sólo posiciones está en rojo y la predicción del modelo que además usa el flujo óptico de los vecinos sin usar una vecindad en azul.

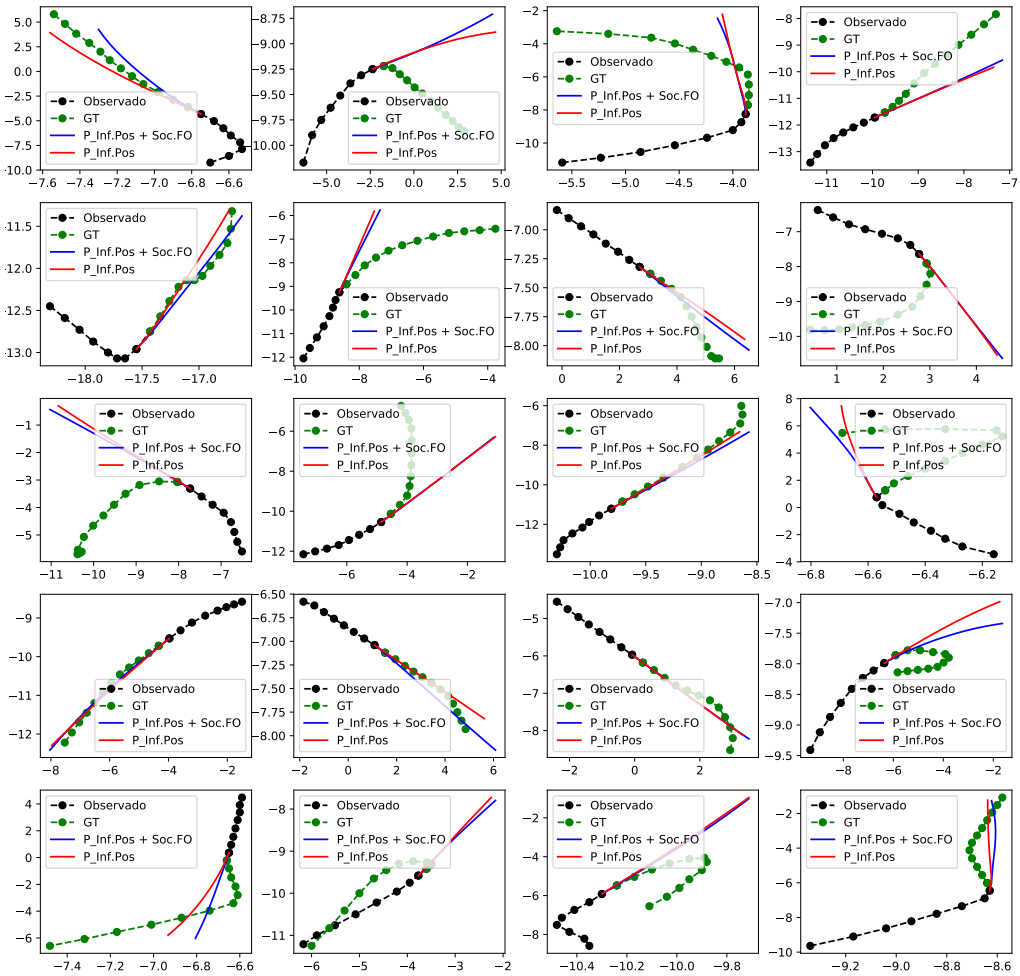


Figura 5.19: Ejemplos de predicciones en coordenadas mundo en PETS2009-S2L1 al entrenar el modelo con el enfoque PRO en la división de los datos. Los 8 pasos observados aparecen con puntos negros, los 12 pasos verdaderos que se quieren aprender a predecir con puntos verdes, mientras que la predicción del modelo que usa sólo la información de posiciones está en rojo y la predicción del modelo con configuración posiciones y flujo óptico de los vecinos sin estar restringidos a una vecindad en azul.

	Basados en LSTM	
	Inf.Pos	Inf.Pos+Soc.FO-OBS
ETH	**1.09/2.29**	1.12/2.35
HOTEL	**0.51/0.99**	0.55/1.07
UCY	**0.62/1.34**	0.64/1.39
ZARA01	**0.41**/0.90	0.41/**0.89**
ZARA02	**0.32/0.71**	0.33/0.74
AVG	**0.59/1.25**	0.61/1.29

Tabla 5.8: Predicción de 12 posiciones futuras: Errores en metros (ADE/FDE), donde el entrenamiento del modelo se realizo con el enfoque LOO para la división de los datos y MNP-INT para el número máximo de personas, para las configuraciones de solo posiciones (segunda columna), información de posiciones junto con el flujo óptico de los vecinos y obstáculos fijos sin tomar en cuenta una vecindad (tercera columna).

De la Tabla 5.8, donde cada uno de los modelos son entrenados y validado con 4 de los conjuntos de datos de ETH/UCY y probado con el restante, se puede apreciar de nuevo, que en está modalidad de entrenamiento al agregarle a la información de posición la de flujo óptico generada de los peatones vecinos y obstáculos fijos, la predicción se ve afectada aún más que cuando sólo agregábamos el flujo óptico de los vecinos.

Evaluación cualitativa

En las figuras 5.20, 5.21, lo que se ilustra es la diferencia de los resultados obtenidos al predecir con el modelo que usa la información de posiciones históricas junto con la de flujo óptico sin vecindad (en azul) contra el modelo que tiene además la información de obstáculos fijos por medio del flujo óptico (en rojo).

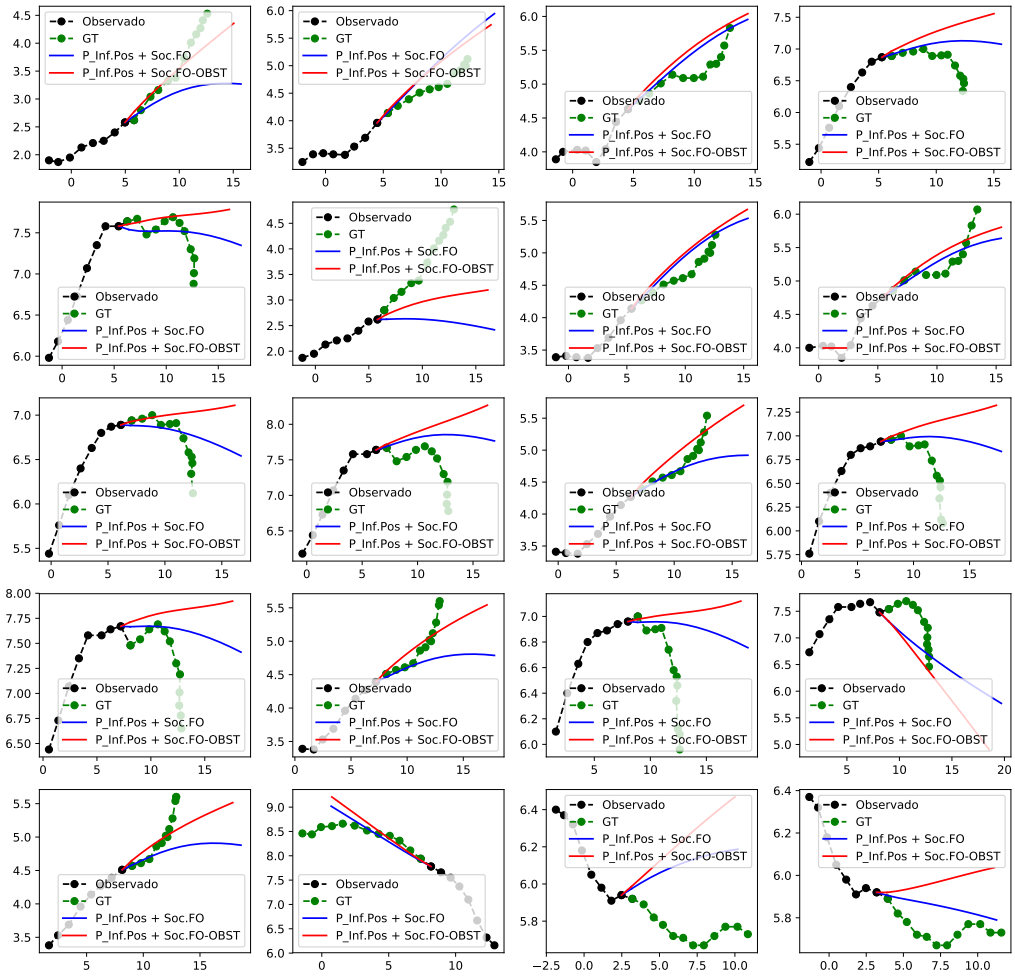


Figura 5.20: Ejemplos de predicciones en ETH-Univ al entrenar el modelo con el enfoque LOO para la división de los datos y MNP-INT para el número máximo de personas.

5.7.4. Comparación

En la Tabla 5.9, se aprecia los resultados de distintos modelos entrenados con el enfoque LOO en los conjuntos de datos ETH/UCY, presentados brevemente en la Tabla 2.1. Hay diferentes modelos, desde el más sencillo que no usa redes neuronales (*Interpolación*), así como los que modelan al peatón de forma individual, es decir,

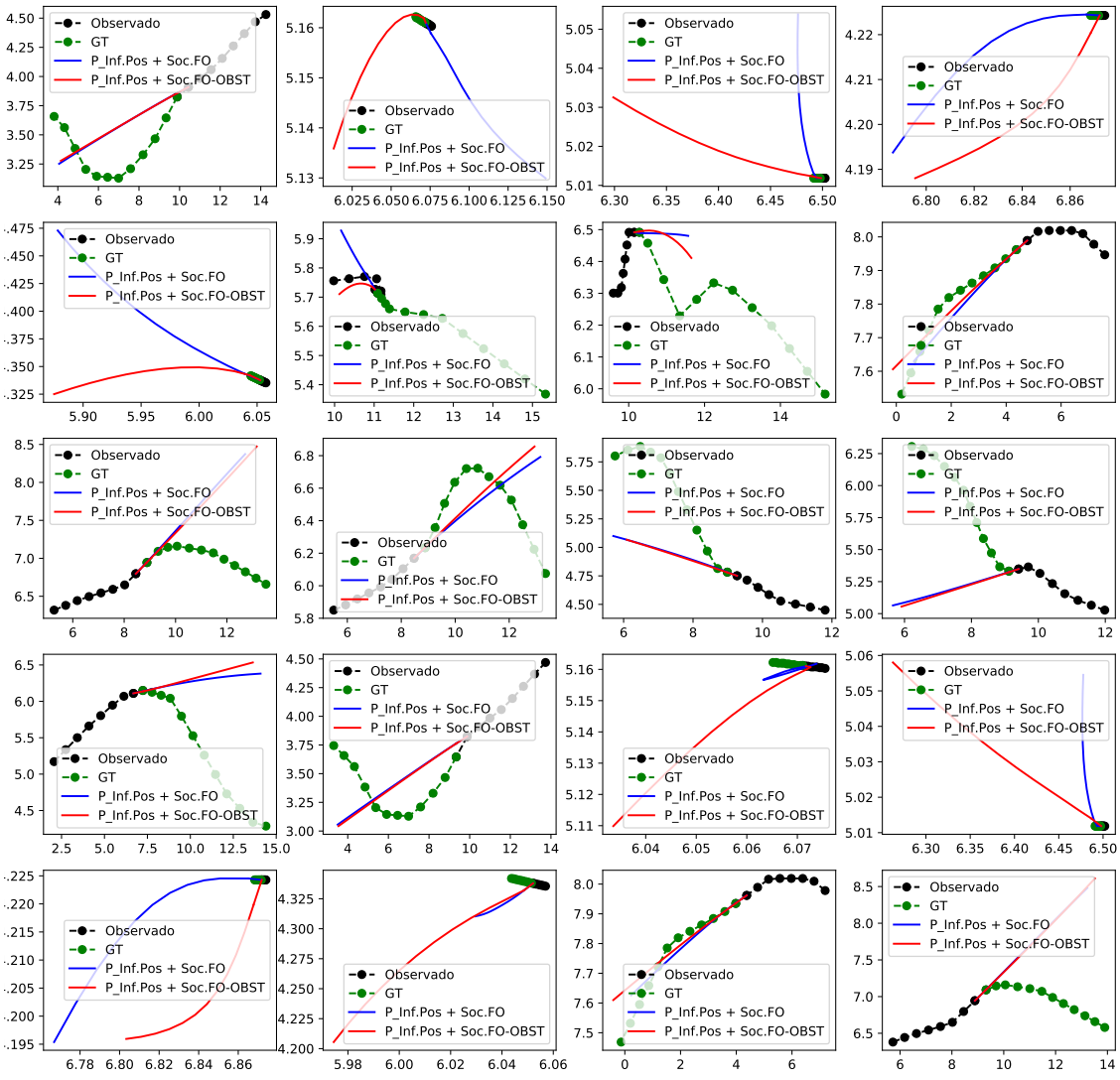


Figura 5.21: Ejemplos de predicciones en Crowds-Zara02 al entrenar el modelo con el enfoque LOO para la división de los datos y MNP-INT para el número máximo de personas.

	Lineal	Basados-LSTM				
	Individual	Individual		Social		
	Interpolación.	LSTM	S-GAN-ind	Social-LSTM	Soc.Att.	Inf.Pos+Soc.OF
ETH	1.33/2.94	1.09/2.41	1.13/**2.21**	1.09/2.35	**0.39**/3.74	1.10/2.33
HOTEL	0.39/**0.72**	0.86/1.91	1.01/2.18	0.79/1.76	**0.29**/2.64	0.50/0.96
UCY	0.82/1.59	0.61/1.31	0.60/**1.28**	0.67/1.40	**0.20**/0.52	0.64/1.40
ZARA01	0.62/1.21	0.41/**0.88**	0.42/0.91	0.47/1.00	**0.30**/2.13	0.41/0.89
ZARA02	0.77/1.48	0.52/1.11	0.52/1.11	0.56/1.17	0.33/3.92	**0.32**/**0.71**
AVG	0.79/1.59	0.70/1.52	0.74/1.54	0.72/1.54	**0.30**/2.59	0.59/**1.26**

Tabla 5.9: Comparación del error de predicción de nuestro método propuesto (Soc.OF) contra algunas baselines populares. Los valores de ADE y FDE son separados por un slash. Los errores están en metros.

sin tomar en cuenta a los peatones de su entorno, y también los hay que sí toman en cuenta la interacción social.

5.8. Conclusión

En este capítulo, se propuso el flujo óptico como una forma de modelar la interacción del agente observador con su entorno (*otros peatones, obstáculos fijos*). De acuerdo a los resultados obtenidos de los distintos experimentos, se puede concluir que cuando entrenamos nuestro modelo con el enfoque PRO, obtenemos una mejora al integrar el flujo óptico, ya que, por la forma con la cual estamos haciendo el entrenamiento, el modelo al momento de predecir ya conoce la naturaleza de los datos. Recordemos que PRO consiste en entrenar con 70 % y validar con 10 % de cada uno de los conjuntos de datos que tengamos y el 20 % restante de cada uno de los conjuntos de datos conformar el conjunto de prueba. En este caso, también se observó otra mejora de la predicción al momento de agregarle la información de *pose*.

En cambio, cuando se entrena con el enfoque LOO, es más difícil la tarea, y en este caso se observó que al incluir el flujo óptico, en promedio se tiene un pequeño perjuicio al momento de la predicción. No obstante, es importante comentar que el caso de las trayectorias en las cuales el peatón se mueve con una velocidad pequeña

se observa que al usar el flujo óptico, la dirección de la trayectoria predicha es mejor, aunque su longitud no lo es. Esto se puede apreciar en algunos ejemplos ilustrados en la Fig. 5.18.

Finalmente, al comparar la modalidad de usar una vecindad para el flujo óptico contra la que no tiene ninguna restricción, cuantitativamente no se aprecia una gran diferencia, pero esto es porque no se quiso hacer una reducción dramática del tamaño de la vecindad. Cualitativamente, se puede apreciar en las Figuras 5.11, 5.15, 5.16, que cuando no existe restricción en la vecindad, las predicciones asemejan mejor su dirección con las posiciones verdaderas. Por ende, se concluye que es mejor el modelado del flujo óptico sin vecindad, porque éste también contempla a las personas cercanas al peatón observador, y a los lejanos.

Capítulo 6

Conclusiones y Trabajo a Futuro

En resumen, esta tesis se centró en el trabajo de predicción de trayectorias futuras, que en los últimos años ha sido un tema de investigación que ha tomado gran relevancia en el campo de Visión Computacional. Poder predecir la posición futura de peatones, autos, etc. es crucial para los automóviles autónomos o robots no tripulados porque, a partir de las predicciones, pueden tomar mejores decisiones al momento de planear su ruta, y así poder coexistir en armonía con los otros usuarios. Por consiguiente, en esta tesis se estudió dos factores que pensamos que afectan a la predicción de trayectorias.

- Se analizó si la forma de representar los datos de entrada y salida de una red afecta el aprendizaje de ésta.
- Se estudió si la información de la pose de las personas ayuda en predecir las intenciones futuras de estas.
- Se propuso una nueva forma de modelar las interacciones sociales de los peatones con su entorno, por medio de un vector 1D de flujo óptico.

En la primera sección de este capítulo, se presentan las conclusiones generales a las cuales se llegaron después de realizar cada uno de los experimentos descritos en los capítulos anteriores. En la sección siguiente, se describe algunas ideas para trabajo a futuro.

6.1. Conclusiones

Como primer punto, se estudió varias formas de representar los datos de entrada y salida (*coordenadas absolutas, desplazamientos y variación al modelo de regresión lineal*). Con dichas representaciones, se hizo predicciones en diferentes conjuntos de datos y con diferentes frecuencias de muestreo. De esos experimentos, se concluyó que la forma de representar los datos sí es muy importante en la precisión de las trayectorias predichas. Además, trabajar con los datos totalmente en su forma absoluta conlleva a que el modelo tenga que aprender más parámetros y por ende a ser más tardado el aprendizaje. En particular, cuando el muestreo es más frecuente (7.5 fps) y la entrada a la red se da de forma absoluta y las salidas (*absolutas, desplazamientos y variación al modelo de regresión*), se pudo concluir que para este caso la mejor representación fue el modelo de variación al modelo de regresión lineal, y esto fue porque las trayectorias son trozos pequeños, en su mayoría cerca de rectas, por lo que el modelo de variación de regresión que fue elaborado con una regresión de grado uno, se ajusta muy bien en este caso. En general, el modelo que expresa tanto la entrada y salida con desplazamientos es el que da mejores resultados en base a las métricas ADE y FDE, sin importar la frecuencia de muestreo.

Posteriormente, en el Capítulo 4, se estudió en qué medida la información de la pose de las personas ayuda a describir sus acciones futuras. Para ello, la pose se representó por medio de 18 puntos claves del esqueleto, entre los cuales están: *cuello, nariz, pie derecho, pie izquierdo, rodilla derecha, rodilla izquierda,etc.*, y en base a los desplazamientos de las posiciones de estos, se evaluó que, al incluir dicha información como entrada al modelo, las predicciones mejoraban en promedio un 6 % en las métricas ADE y FDE. También se incluyó el valor de confianza de la identificación de cada uno de los puntos claves que estos tienen, y se tuvo una mejora del 2 % adicional, así que la mejora en cada uno de los conjuntos de datos depende de la calidad de la adquisición de los "keypoints".

Por último, se concluyó que al agregar un modelado de la interacción social no fue de gran ayuda, ya que se observan resultados mejores sólo al agregar información de pose. No obstante, puede ser de ayuda cuando no se tiene la información de pose y que el entrenamiento y las pruebas se hacen sobre datos similares.

6.2. Trabajo a Futuro

En el trabajo a futuro consideramos lo siguiente:

- Actualizar el tensor de características **Q** de la Ecuación 4.3, para los instantes de tiempo predichos (no sólo en el pasado).
- Estimar con sistemas inteligentes la dirección hacia la cual observan los peatones. Nos servirá cuando no se mueven en los pasos observados.
- Actualizar los mapas de obstáculos fijos, en cada instante de tiempo.
- Probar la propuesta del modelado de interacción social por medio del vector 1D de flujos ópticos, con diferentes ángulos de visión, y con diferentes tamaños del vector 1D.
- Extender el problema a distintos tipos de agentes (peatones,ciclistas. . .), donde las trayectorias de estos agentes heterogéneos puedan interactuar entre si.
- Inspirado en el trabajo de Zhong et al. [29], predecir trayectorias de peatones en el espacio 3D, tomando en cuenta las contribuciones de los demás agentes. Dicha contribución de los agentes en un instante de tiempo, sería modelado con la propuesta de modelado del flujo óptico, sólo que sería un vector 2D.

6.3. Publicaciones a raíz de esta tesis

Con esta tesis se obtuvo una publicación en la IEEE, con el título Evaluation of Output Representations in Neural Network-based Trajectory Predictions Systems [7], al participar en *The 27th International Conference on Systems, Signals and Image Processing.*

Bibliografía

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, y S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. En *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 961–971. 2016. doi:10.1109/CVPR.2016.110.

[2] George Awad, Asad A Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzad Godil, David Joy, Andrew Delgado, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, Joao Magalhaes, David Semedo, y Saverio Blasi. TRECVID 2018: Benchmarking Video Activity Detection, Video Captioning and Matching, Video Storytelling Linking and Video Search. En *Proceedings of TRECVID 2018*. Gaithersburg, MD, United States, 2018. URL https://hal.archives-ouvertes.fr/hal-01919873.

[3] Ben Benfold y Ian Reid. Guiding visual surveillance by tracking human attention. En *Proceedings of the British Machine Vision Conference*, págs. 14.1–14.11. BMVA Press, 2009. ISBN 1-901725-39-1. Doi:10.5244/C.23.14.

[4] Jason Brownlee. Machine Learning Mastery. https://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting/, 2017. [Online; accessed 03-May-2020].

[5] Diego Calvo. DIEGO CALVO Análisis de Datos. http://www.diegocalvo.es, 2018. [Online; accessed 16-Sepetember-2020].

[6] Z. Cao, T. Simon, S. Wei, y Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. En *2017 IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR), págs. 1302–1310. 2017. doi:10.1109/CVPR.2017.143.

[7] A. Ek-Hobak, A. Sanchez, y J. Hayet. Evaluation of output representations in neural network-based trajectory predictions systems. En *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, págs. 447–452. 2020. ISSN 2157-8702. doi:10.1109/IWSSIP48289.2020.9145309. URL https://ieeexplore.ieee.org/document/9145309.

[8] J. Ferryman y A. Shahrokni. Pets2009: Dataset and challenge. En *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, págs. 1–6. 2009. doi:10.1109/PETS-WINTER.2009.5399556.

[9] Francesco Giuliari, Irtiza Hasan, Marco Cristani, y Fabio Galasso. Transformer Networks for Trajectory Forecasting. *arXiv e-prints*, arXiv:2003.08111, 2020.

[10] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, y A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. En *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, págs. 2255–2264. 2018. doi:10.1109/CVPR.2018.00240.

[11] Dirk Helbing y Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995. ISSN 1095-3787. doi:10.1103/physreve.51.4282. URL http://dx.doi.org/10.1103/PhysRevE.51.4282.

[12] B. Ivanovic y M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. En *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, págs. 2375–2384. 2019. doi:10.1109/ICCV.2019.00246.

[13] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian D. Reid, Seyed Hamid Rezatofighi, y Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *CoRR*, abs/1907.03395, 2019. URL http://arxiv.org/abs/1907.03395.

[14] Namhoon Lee, Wongun Choi, Paul Vernaza, Chris Choy, Philip Torr, y Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with

interacting agents. En *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 2165–2174. 2017. doi:10.1109/CVPR.2017.233.

[15] Alon Lerner, Yiorgos Chrysanthou, y Dani Lischinski. Crowds by Example. *Computer Graphics Forum*, 2007. ISSN 1467-8659.

[16] J. Liang, L. Jiang, J. C. Niebles, A. Hauptmann, y L. Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. En *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, págs. 2960–2963. 2019. doi:10.1109/CVPRW.2019.00358.

[17] Axel López, François Chaumette, Eric Marchand, y Julien Pettré. Character navigation in dynamic environments based on optical flow. *Computer Graphics Forum*, 38(2):181–192, 2019. doi:10.1111/cgf.13629. URL https://hal.inria.fr/hal-02052554. Eurographics 2019 proceedings.

[18] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, y M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. En *CVPR 2011*, págs. 3153–3160. 2011. doi:10.1109/CVPR.2011.5995586.

[19] Christopher Olah. Colah's blog. http://colah.github.io/, 2008. [Online; accessed 01-March-2020].

[20] S. Pellegrini, A. Ess, K. Schindler, y L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. En *2009 IEEE 12th International Conference on Computer Vision*, págs. 261–268. 2009. doi:10.1109/ICCV.2009.5459260.

[21] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, y C. Cadena. A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments. En *2018 IEEE International Conference on Robotics and Automation (ICRA)*, págs. 5921–5928. 2018. doi:10.1109/ICRA.2018.8461157.

[22] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, y S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. En *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 1349–1358. 2019. doi:10.1109/CVPR.2019.00144.

[23] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, y Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. En Andrea Vedaldi, Horst Bischof, Thomas Brox, y Jan-Michael Frahm, eds., *Computer Vision – ECCV 2020*, págs. 683–700. Springer International Publishing, Cham, 2020. ISBN 978-3-030-58523-5. URL https://rdcu.be/cdZop.

[24] Karen Simonyan y Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. En Yoshua Bengio y Yann LeCun, eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL http://arxiv.org/abs/1409.1556.

[25] Pete Trautman, Jeremy Ma, Richard M. Murray, y Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015. doi:10.1177/0278364914557874. URL https://authors.library.caltech.edu/55783/1/335.full.pdf.

[26] A. Vemula, K. Muelling, y J. Oh. Social attention: Modeling attention in human crowds. En *2018 IEEE International Conference on Robotics and Automation (ICRA)*, págs. 4601–4607. 2018. doi:10.1109/ICRA.2018.8460504.

[27] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, y Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. En Francis Bach y David Blei, eds., *Proceedings of the 32nd International Conference on Machine Learning*, tomo 37 de *Proceedings of Machine Learning Research*, págs. 2048–2057. PMLR, Lille, France, 2015. URL http://proceedings.mlr.press/v37/xuc15.html.

[28] P. Zhang, W. Ouyang, P. Zhang, J. Xue, y N. Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. En *2019 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), págs. 12077–12086. 2019. doi:10.1109/CVPR.2019.01236.

[29] J. Zhong, H. Sun, W. Cao, y Z. He. Pedestrian motion trajectory prediction with stereo-based 3d deep pose estimation and trajectory learning. *IEEE Access*, 8:23480–23486, 2020. doi:10.1109/ACCESS.2020.2969994.