DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“SISTEMA INTELIGENTE PARA EL RECONOCIMIENTO DE PATRONES Y LA PREDICCIÓN DE CRÍMENES”

T E S I S

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

JOSÉ ULISES TEYECHEA PERALTA

Directora:

Dra. María Trinidad Serna Encinas

Co-directora:

Dra. Rosalía del Carmen Gutiérrez Urquídez

Hermosillo Sonora, México 20 de Agosto de 2021

SECCIÓN: DIV. EST. POS. E INV.
No. OFICIO: DEPI/168/21
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

12 de julio de 2021

C. JOSÉ ULISES TEYECHEA PERALTA,
P R E S E N T E.

Por este conducto, y en virtud de haber concluido la revisión del trabajo de tesis que lleva por nombre **"Sistema Inteligente para el Reconocimiento de Patrones y la Predicción de Crímenes"**; que presenta para el examen de grado de la MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN, y habiéndola encontrado satisfactoria, nos permitimos comunicarle que se autoriza la impresión del mismo a efecto de que proceda el trámite de obtención de grado.

Deseándole éxito en su vida profesional, quedo de usted.

A T E N T A M E N T E

M.C. MARÍA TRINIDAD SERNA ENCINAS
DIRECTOR

M.C. ROSALÍA DEL CARMEN GUTIÉRREZ URQUÍDEZ
CO-DIRECTOR

M.C. CÉSAR ENRIQUE ROSE GÓMEZ
SECRETARIO

S.E.P.

M.C. RAFAEL ARMANDO GALAZ BUSTAMANTE
VOCAL

INSTITUTO TECNOLÓGICO DE HERMOSILLO
DIVISIÓN DE ESTUDIOS DE POSGRADO

M.C.O. ROSA IRENE SÁNCHEZ FERMÍN
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

RISF/eme*

CARTA CESIÓN DE DERECHOS

En la ciudad de Hermosillo Sonora a el día 20 de Agosto del año 2021 el que suscribe C. José Ulises Teyechea Peralta , alumno de la maestría en Ciencias de la Computación adscrito a la División de Estudios de Posgrado e Investigación, manifiesta que es autor intelectual del presente trabajo de Tesis titulado “Sistema inteligente para el reconocimiento de patrones y la predicción de crímenes” bajo la dirección de Dra. María Trinidad Serna Encinas y ceden los derechos del mismo al Tecnologico Nacional de México/Instituto Tecnologico de Hermosillo, para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben de reproducir el contenido textual, graficas, tablas o datos contenidos sin el permiso expreso del autor y del director del trabajo. Este puede ser obtenido a la dirección de correo electrónico siguiente: uteyechea@gmail.com . Una vez otorgado el permiso se deberá expresar el agradecimiento correspondiente y citar la fuente del mismo.

A T E N T A M E N T E

José Ulises Teyechea Peralta

Agradecimientos

Gracias a mi esposa e hija por su paciencia y apoyo. Gracias a mi estimada directora de tesis por su apoyo, guía y amistad. Gracias a mis padres y familia que siempre han tenido una participación activa en mi vida.

Resumen

La criminalidad se caracteriza mediante las propiedades llamadas victimización repetida y victimización cuasi-repetida. La victimización cuasi-repetida se forma cuando personas o bienes con características similares se convierten en víctimas de actividad delictiva repetida. Este tipo de observaciones proporcionan a los departamentos de policía una idea general acerca de la existencia de patrones criminales. Para lograr la predicción del crimen el departamento de policía debe tener la capacidad de capitalizar el conocimiento acerca de la existencia de estos patrones identificados.

La modernización tecnológica en los departamentos de policía, ha permitido la recopilación de grandes conjuntos de datos acerca de la actividad delictiva. Al contar con grandes conjuntos de datos, de incidencias delictivas, es necesario implementar técnicas de análisis que permitan obtener información procesable de los patrones de comportamiento criminal.

La abundante cantidad de información digital y el acceso a poder de cómputo, ha abierto la posibilidad hacia una nueva forma de análisis, mediante la aplicación de métodos de aprendizaje automático; en contraste, el desarrollo de soluciones de ingeniería convencionales continúa presentando deficiencias, debido al proceso de modelado o bien a limitaciones inherentes en los algoritmos.

En el presente trabajo de tesis se busca desarrollar un sistema basado en Inteligencia Artificial que permita predecir la distribución espacio-temporal de crímenes. En la primera implementación del sistema inteligente para el reconocimiento de patrones y la predicción de crímenes se optó por el modelo "Long Short Term Memory" (LSTM, por sus siglas en inglés), dado que los datos criminales forman una serie de tiempo para cada tipo de crimen y por cada zona policiaca en la ciudad de Chicago.

El modelo LSTM tiene una ventaja significativa para encontrar correlaciones temporales en series de tiempo. Sin embargo, la naturaleza de los datos criminales, no es sólo temporal, sino espacio-temporal; por lo tanto, posteriormente a la aplicación del modelo LSTM, se procedió a buscar un nuevo modelo más adecuado para el reconocimiento de patrones espacio-temporales. Se encontró que el modelo "Convolutional Long Short Term Memory" (convLSTM, por sus siglas en inglés), es una opción natural para la identificación de patrones espacio-temporales.

Así que, a lo largo de la implementación del sistema se emplearon los dos modelos. De acuerdo a los resultados obtenidos se optó por emplear el modelo convLSTM para el reconocimiento y predicción de patrones criminales. Las predicciones generadas por el algoritmo predictivo se muestran al usuario a través de una aplicación web.

Los resultados experimentales obtenidos al implementar el modelo LSTM, muestran una correlación promedio de 0.599 entre la cantidad de crímenes registrados por el Departamento de Policía de Chicago y la cantidad de crímenes pronosticados por el modelo convLSTM, así mismo, el RMSE obtenido fue de 0.110. El valor de correlación positiva obtenido es significativo, sobretodo tomando en consideración la complejidad del comportamiento humano, en particular el comportamiento criminal futuro de un individuo.

Los resultados experimentales obtenidos durante las pruebas de funcionalidad del modelo convLSTM muestran una precisión promedio de 0.42 para el conjunto de prueba, la precisión medida en este caso corresponde a la diferencia promedio entre el valor exacto y el valor predicho. Sin embargo, como la evidencia experimental sugiere, se obtuvo una correlación cercana a 1 en la mayoría de las evaluaciones experimentales.

El modelo convLSTM es el caso general del modelo LSTM ya que no solo es eficaz durante la identificación de correlaciones temporales, sino también en la identificación de correlaciones espaciales entre los diversos tipos de crimen. Por lo tanto, el modelo convLSTM es el mejor modelo propuesto hasta el día de hoy para la identificación, predicción y clasificación de patrones en secuencias espacio-temporales.

Índice de figuras

Índice general

Capítulo 1

Introducción

El Índice de Paz Global (GPI, Global Peace Index, por sus siglas en inglés) se ha deteriorado en cuatro de los últimos cinco años previos al año 2020. México se encuentra en el lugar 137 de 153 países incluidos en el GPI. El valor total de pérdidas debidas a la violencia en México se estima que alcanzó el 10 % del PIB en el año 2019 [1]. Sin embargo, en 2020 México es más seguro con respecto al mismo periodo del año pasado. En el primer trimestre del año, el robo a negocios, casas y autos ha disminuido un 16 %, 8 % y 15 %, respectivamente, mientras que la violación y la violencia intrafamiliar muestran un aumento del 14 % y 16 % [2].

Un factor causal importante en este efecto transitorio de baja criminalidad en el año 2020, es la política de distanciamiento social, implementada como medida preventiva al contagio por COVID-19. La disminución en los delitos, como por ejemplo el robo residencial, pueden ser explicados como consecuencia del aumento en la custodia sobre el espacio personal y la propiedad. El aumento en los delitos como la violencia doméstica pueden ser explicados como consecuencia de períodos prolongados de contacto, entre posibles delincuentes y víctimas. Por lo tanto, la disminución del crimen en México no es un evento aislado [3].

En general, la sociedad considera que existe una relación causal entre la pobreza y la criminalidad. Sin embargo, la realidad no es tan simple, la existencia de una correlación entre la pobreza y la criminalidad es una condición necesaria pero no suficiente para establecer una relación causal. Experimentalmente aún no se ha confirmado que exista alguna correlación relevante entre algunos delitos e indicadores económicos [4]. Por lo tanto, la correlación entre indicadores económicos y la criminalidad aún está en discusión. Además, en el caso de algunos crimenes tales como el robo, el costo social asociado a la prevención del crimen (costo individual del individuo preso, recursos policiacos, etc) puede ser mayor al costo social si se considera solamente el valor de los bienes robados.

Por lo tanto, la solución a este problema es uno donde se debe buscar encontrar el equilibrio entre el costo social vs la prevención del crimen [5].

La buena gobernanza es una condición necesaria para la igualdad social. La anomia puede surgir cuando el estado de derecho no es aplicado por igual. Un estado de derecho débil en un país se convierte en un factor causal que da origen a la criminalidad. La lógica es que una política débil no produce un sentido de comunidad y solidaridad, lo que facilita que surja la anomia. En tales situaciones, las estructuras culturales fallan en su función normativa. Por lo tanto, la combinación de un estado de derecho débil y una sociedad capitalista, incentiva a que algunos miembros de la sociedad practiquen el crimen como fuente de ingreso económico [6].

En respuesta a esta problemática, diversos grupos de investigación han implementado diferentes técnicas de análisis para grandes conjuntos de datos de incidencias delictivas. Las máquinas de soporte vectorial (SVM, por sus siglas en inglés), han demostrado ser efectivas para la predicción en la reincidencia criminal de un individuo [7], y la predicción de "puntos calientes" donde la probabilidad de incidencia delictiva es significativa [8]. Algoritmos basados en lógica difusa han demostrado ser razonablemente eficientes para predecir el lugar más probable dónde ocurrirá un crimen [9]. Modelos basados en agentes han demostrado ser prometedores al simular una ciudad y sus ciudadanos, entre los cuales existen ladrones, y así, entender mejor los factores que motivan a este tipo de criminales [10]. Los métodos de análisis para series de tiempo convencionales, tales como agrupamiento por similitud, no han demostrado ser efectivos para encontrar tendencias en conjuntos de datos criminales multidimensionales [11]. Por último, los modelos de predicción de delitos seriales, utilizando la teoría de aprendizaje bayesiana, han demostrado ser efectivos para predecir el vecindario más probable donde el criminal cometerá el siguiente delito [12].

En este trabajo de tesis se desarrolla una propuesta de identificación y predicción de patrones en la distribución espacio-temporal, de la cantidad de crímenes por unidad de tiempo y por unidad de área geográfica para una ciudad. El modelo de inteligencia artificial empleado para la identificación y predicción de crímenes se llama "convLSTM" [13]. Las experimentaciones realizadas muestran la cantidad de crímenes esperados por día y por comunidad en la ciudad de Chicago, U.S.A. Los resultados obtenidos se presentan al usuario a través de una aplicación web. El conjunto de estas herramientas tecnológicas ayudan a combatir la delincuencia mediante la creación de una policía predictiva [14].

1.1. Antecedentes

La modernización tecnológica en los departamentos de policía ha permitido la recopilación de grandes conjuntos de datos acerca de la actividad delictiva. Al contar con una gran cantidad de conjuntos de datos de incidencias delictivas, se vuelve posible implementar técnicas de análisis que permitan obtener información accionable, acerca de los patrones de comportamiento criminal [15]. Experimentalmente, se observa que ciertos tipos de delitos tales como el robo y la violencia, forman secuencias de eventos altamente agrupados. Es decir, la distribución espacio-temporal de una secuencia de crímenes subsecuentes, a partir de la ocurrencia de un primer crimen, se concentra alrededor de la vecindad espacial y dentro de un intervalo de tiempo de aproximadamente algunos días, después de la ocurrencia de dicho primer crimen [16].

Así mismo, se observa el patrón criminal llamado victimización repetida. Este tipo de observaciones proporcionan a los departamentos de policía una técnica predictiva para la prevención del crimen, si y sólo si, la policía tiene la capacidad de analizar y obtener conocimiento a partir de estos patrones identificados [17].

Las fuerzas policiales en algunas ciudades ya están aplicando tecnologías diseñadas para predecir crímenes. Algunos ejemplos de estos sistemas son: PredPol [18], RTM [19] y Hunchlab [20].

Para analizar y obtener conocimiento a partir de un conjunto de datos, se ha optado por la implementación del aprendizaje automático. En este caso, el primer paso ya no es la adquisición de conocimientos de dominio, sino obtener suficientes datos asociados con el problema. Dependiendo de la naturaleza de los datos, se procede a seleccionar un modelo de aprendizaje automático apropiado. Finalmente, el conocimiento del dominio puede ser útil al ajustar el modelo.

Una red neuronal se basa en una colección de unidades o nodos conectados. Esta unidad de procesamiento o neurona artificial se llama perceptrón [21]. Una Red Neuronal Artificial (ANN, por sus siglas en inglés) busca modelar artificialmente la comunicación y la estructura de las neuronas en un cerebro biológico.

Las redes neuronales tipo Long Short Term Memory (LSTM, por sus siglás en inglés), son un tipo de RNN. LSTM cuenta con la capacidad de aprender las dependencias a largo plazo que resultan problemáticas para una RNN tradicional. El modelo LSTM está diseñado explícitamente para resolver el problema de dependencia a largo plazo. Su comportamiento predeterminado es aprender a recordar información durante largos períodos de tiempo.

Una limitante importante del modelo LSTM consiste en su incapacidad para modelar eficazmente correlaciones espaciales. El modelo convLSTM busca resolver este problema. En el modelo convLSTM se propone una extensión del modelo LSTM en la cual se tienen estructuras convolucionales en las transiciones de entrada a estado y de estado a estado. El ConvLSTM determina el estado futuro de una determinada celda en la cuadrícula, mediante las entradas y los estados pasados de sus vecinos locales.

En el aprendizaje profundo, es común ver mucha discusión sobre los tensores como la estructura de datos fundamental. Tensor incluso aparece en nombre de la biblioteca insignia de aprendizaje automático de Google: "TensorFlow". Los tensores son un tipo de estructura de datos que se utiliza en álgebra lineal y, al igual que los vectores y las matrices, se pueden realizar operaciones aritméticas con tensores.

1.2. Planteamiento del problema

El estado de Sonora, ocupa el tercer lugar a nivel nacional con la mayor cantidad de crímenes reportados por cada 100,000 habitantes. La ciudad de Hermosillo ocupa el primer lugar en la cantidad de crímenes reportados en todo el estado de Sonora, con aproximadamente 36 % del total de crímenes que ocurren en el estado, los tipos de crimenes considerados son: robo a comercio, robo a casa, homicidio y robo de auto [22].

Preguntas de investigación

1. ¿Qué variables predicen mejor la ocurrencia de crimen?
2. ¿Cuál es el esquema de base de datos a emplear?
3. ¿Qué algoritmo es eficiente y tiene la capacidad de realizar predicciones precisas?
4. ¿Cuál será el diseño de la interfaz del usuario que mostrará la predicción del crimen?

Pregunta de investigación principal: ¿Qué variables y modelo debe considerar un sistema inteligente que busca predecir la distribución espacio-temporal del crimen?

1.3. Objetivos

Para el desarrollo del sistema predictivo propuesto se han planteado los siguientes objetivos.

1.3.1. General

Implementar un sistema basado en Inteligencia Artificial que permita predecir la distribución espacio-temporal de crímenes.

1.3.2. Particulares

- Determinar experimentalmente una arquitectura viable del modelo de Inteligencia Artificial seleccionado.
- Aplicar un proceso ETL para la elaboración de la base de datos que contenga las variables seleccionadas como factores causales de los crímenes futuros.
- Desarrollar una interfaz de usuario que muestre la predicción del crimen por unidad de tiempo y por unidad geográfica.

1.4. Justificación de la tesis

El ambiente de educación y socialización más importante del individuo es la familia. En la familia se aprenden valores individuales y sociales, que después se proyectan en la comunidad. La exposición a ciertos eventos en diferentes momentos durante la vida de las personas pueden colocar a las personas en diferentes trayectorias, llevándolos por diferentes caminos. Los resultados experimentales muestran que el abuso o la negligencia aumenta el riesgo de mala conducta posterior, en particular tienden a generar individuos violentos. En general, las personas que experimentan algún tipo de abuso físico violento o abuso por negligencia durante la infancia, tienen una mayor probabilidad de cometer un crimen de adultos. Los niños que han sufrido algún tipo de abuso en general tienen una probabilidad del 26 % de ser arrestados siendo aún menores de edad, en contraste con un 17 % en niños que no sufrieron abuso; tienen una probabilidad del 29 % de ser arrestados siendo adultos, en contraste con un 21 % en niños que no sufrieron abuso y una probabilidad del 11 % de ser arrestados por algún delito violento, en contraste con un 8 % en niños que no sufrieron abuso [23].

El aumento en los índices de criminalidad en todo el país y la disminución de los recursos policiales, han provocado una creciente polémica sobre el nivel de inseguridad ante la sociedad. Debido al grave problema de inseguridad y al gran daño económico, las autoridades se enfrentan ante el gran reto de controlar y reducir la delincuencia mediante la utilización eficiente de los recursos limitados con los que cuenta. Lo anterior plantea el reto de realizar trabajos de investigación que propongan sistemas de software capaces de

reducir los crímenes y optimizar el uso de los recursos policiales en la ciudad. Utilizando algoritmos predictivos de inteligencia artificial se busca lograr la optimización de los recursos policiales disponibles.

1.5. Alcances y delimitaciones

El sistema no analiza perfiles criminales: el sistema propuesto sólo es capaz de analizar crímenes de los cuales existe un registro histórico que incluya por lo menos tres tipos de datos: tipo de crimen, lugar y tiempo del incidente delictivo; por lo tanto no es posible analizar ningún tipo de información sobre la víctima o sobre un presunto autor criminal.

Las predicciones no garantizan un arresto: el sistema propuesto procesa grandes conjuntos de datos de crímenes reportados, para encontrar el tiempo y lugar donde se detecte una incidencia delictiva futura. En la práctica, las áreas geográficas designadas de alto riesgo son susceptibles al decremento del futuros crímenes empleando estrategias de patrullaje y como resultado el criminal potencial es disuadido por lo cual no comete un crimen y no se lleva a cabo un arresto.

Las predicciones del sistema afectan su comportamiento futuro: se refiere a la capacidad de adaptación del sistema, puesto que si la predicción del crimen es acompañada por un patrullaje del área geográfica más efectivo, entonces puede ocurrir que se generan crímenes en nuevas zonas con pocos datos históricos, en el peor de los casos, y en el mejor de los casos se logren arrestos tales que minimicen el crimen futuro.

1.6. Metodología

Fase 1: se realiza un análisis a profundidad de las diferentes temáticas que inciden en el estado del arte. Con el objetivo de conocer a detalle la problemática y sustentar la propuesta de solución.

Fase 2: se realiza el análisis y diseño del sistema, es decir, se elaboran los diferentes diagramas que modelan el sistema y que incluye tanto el modelo de datos como la arquitectura propuesta.

Fase 3: se realiza la programación del sistema y las pruebas de funcionalidad.

Fase 4: se realiza el análisis de los resultados obtenidos.

1.7. Organización de la tesis

Capítulo 2: Estado del arte, presenta el estudio y análisis exhaustivo de los diferentes temáticos que inciden directamente con la problemática a resolver, tales como: el crimen, inteligencia artificial, aprendizaje profundo, predicción del crimen y los sistemas relacionados con la policía predictiva.

Capítulo 3: Análisis y diseño, describe el sistema a implementar, cuenta con diagramas de nivel y de datos, y la arquitectura del sistema.

Capítulo 4: Implementación, consiste en el desarrollo del sistema para la predicción de crímenes.

Capítulo 5: Análisis de resultados, cuantifica la calidad de los resultados obtenidos empleando métricas de error y precisión.

Capítulo 6: Conclusiones, resume el trabajo de investigación realizado, los resultados obtenidos y se realiza el planteamiento del trabajo a futuro.

Capítulo 2

Estado del arte

2.1. Introducción

Experimentalmente se observa que ciertos tipos de delitos tales como el robo y la violencia, forman secuencias de eventos altamente agrupados. Es decir, la distribución espacio-temporal de una secuencia de crímenes subsecuentes, a partir de la ocurrencia de un primer crimen, se concentra alrededor de la vecindad espacial y dentro de un intervalo de tiempo de aproximadamente algunos días, después de la ocurrencia de dicho primer crimen [16]. Así mismo, se observa el patrón criminal llamado victimización repetida. Este patrón se forma cuando la misma víctima de algún crimen, continúa siendo víctima de crímenes subsecuentes. También, se tiene la victimización cuasi-repetida, este patrón criminal se forma cuando personas o bienes con características similares se convierten en víctimas de la actividad delictiva. Este tipo de observaciones proporcionan a los departamentos de policía una técnica predictiva para la prevención del crimen, si y sólo si, la policía tiene la capacidad de capitalizar el conocimiento de estos patrones identificados [17]. La modernización tecnológica, en los departamentos de policía, ha permitido la recopilación de grandes conjuntos de datos acerca de la actividad delictiva a nivel local. Al contar con grandes conjuntos de datos, acerca de incidencias delictivas, es necesario implementar técnicas de análisis que permitan obtener información accionable acerca de los patrones de comportamiento criminal [15].

2.2. Historia de Inteligencia Artificial

A lo largo del siglo XX la imaginación popular con respecto a inteligencia artificial fue cultivada por autores tales como L. Frank Baum en su obra “El mago de Oz”, donde se introduce el hombre de hojalata “sin corazón” en 1900, en 1925 Thea von Harbou

describe a el robot humanoide María en su obra titulada “Metrópolis”, así mismo, Issac Asimov, quien en 1941 acuño la palabra “robot” en su historia corta “Liar!”, además definió las tres leyes de la robótica en su historia corta “Runaround” en 1942.

A partir de la segunda mitad del siglo XX, se había formado una generación de científicos, matemáticos y filósofos en donde la idea de robots inteligentes había sido asimilada culturalmente en sus mentes. Una de esas personas fue Alan Turing, quien sugirió que los humanos usan la información disponible y la razón para resolver problemas y tomar decisiones, entonces, ¿por qué las máquinas no pueden hacer lo mismo? Éste fue el marco lógico de su artículo de 1950, Computing Machinery and Intelligence, en el que discutió cómo construir máquinas inteligentes y como probar su inteligencia.

Sin embargo, antes de 1949, las computadoras aún no tenían la capacidad de almacenar comandos, es decir, memoria, sino sólo eran capaces de ejecutar comandos. Por lo tanto, durante la vida de Turing la tecnología aún no era suficiente para materializar sus ideas. En 1956, John McCarthy y Marvin Minsky organizaron el “Proyecto de investigación de verano de Dartmouth sobre inteligencia artificial (DSRPAI)”, durante esta conferencia John McCarthy acuñó el término inteligencia artificial y la definió como “la ciencia y la ingeniería de la fabricación de máquinas inteligentes”. Así mismo, Allen Newell, Cliff Shaw y Herbert Simon’s presentaron su programa “The Logic Theorist”, éste fue un programa diseñado para imitar las habilidades de resolución de problemas de un ser humano.

De 1957 a 1974, la inteligencia artificial floreció. Las computadoras podían almacenar más información y ser más rápidas, más baratas y más accesibles. El primer transistor de silicio fue desarrollado por G. Moore en 1959. El primer circuito integrado con cuatro transistores fue elaborado por Fairchild Semiconductors en 1960, este invento marcó el inicio de la Ley de Moore, donde el número de transistores en un circuito integrado denso se duplica cada dos años.

Las primeras demostraciones, como General Problem Solver de Newell y Simon y ELIZA de Joseph Weizenbaum, se mostraron prometedoras hacia los objetivos de resolución de problemas y la interpretación del lenguaje hablado, respectivamente.

Entre los años de 1974 y 1980, la innovacón en inteligencia artificial se detuvo, principalmente debido a tres factores, recordados por la historia como “El debacle de SUR”, en donde el equipo SUR había desarrollado un sistema que podía reconocer el inglés hablado, pero solo si las palabras se pronunciaban en un orden particular. “Reducción en el presupuesto asignado por DARPA” y “El reporte de Lighthill”, su informe criticaba el absoluto fracaso de la IA para lograr sus “grandiosos objetivos”. Concluyó que nada de lo que se hace en IA no se puede hacer en otras ciencias.

En la década de 1980, la investigación y desarrollo de la inteligencia artificial fue reactivada, principalmente debido a dos factores: una expansión del conjunto de herramientas algorítmicas y un aumento de fondos.

John Hopfield y David Rumelhart popularizaron las técnicas de "aprendizaje profundo" que permitían a las computadoras aprender a prueba y error a partir de la experiencia. Por otro lado, Edward Feigenbaum introdujo sistemas expertos que imitaban el proceso de toma de decisiones de un experto humano. El programa le preguntaba a un experto en un campo como responder en una situación dada, y una vez que esto se aprende, para prácticamente todas las situaciones, los no expertos pueden recibir asesoramiento de este programa.

Durante las décadas de 1990 y 2000, se habían logrado muchos de los objetivos históricos de la inteligencia artificial. En 1997, el campeón mundial de ajedrez y gran maestro Gary Kasparov fue derrotado por Deep Blue de IBM. Así mismo en 1997, se implementó en Windows el software de reconocimiento de voz, desarrollado por Dragon Systems.

Los éxitos de la actual "primavera de la IA" son los avances en la traducción de idiomas (en particular, Google Translate), el reconocimiento de imágenes (impulsado por la base de datos de entrenamiento ImageNet) comercializado por Google Image Search, y en sistemas de juego como Alpha Zero (campeón de ajedrez) y AlphaGo (campeón de Go), y Watson (campeón de Jeopardy). La mayoría de estos avances se produjeron en el periodo del 2010 al 2017.

2.3. Machine learning

2.3.1. Introducción

La abundante cantidad de información digital y el acceso a poder de cómputo ha abierto la posibilidad hacia una nueva forma de análisis, mediante la aplicación de métodos de aprendizaje automático; en contraste, el desarrollo de soluciones de ingeniería convencionales continúa presentando deficiencias debido al proceso de modelado o bien a limitaciones inherentes en los algoritmos previos [24].

Comparemos los dos enfoques comúnmente utilizados para la resolución de problemas. Por un lado, está la forma de ingeniería convencional. Como se muestra en la Figura 2.1, la resolución de problemas comienza adquiriendo conocimientos de dominio. Después de un estudio adicional, se desarrolla un modelo matemático. Dependiendo de la implementación, la codificación del algoritmo puede brindar una mayor eficiencia durante la experimentación y la validación del modelo.

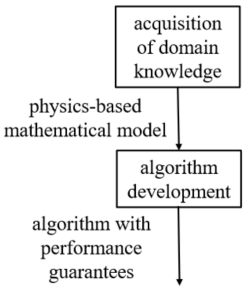


FIGURA 2.1: Flujo de diseño de ingeniería convencional.

Por otro lado, existe el enfoque de aprendizaje automático para la resolución de problemas. Como se muestra en la Figura 2.2. El primer paso ya no es la adquisición de conocimientos de dominio, sino obtener suficientes datos asociados con el problema. Dependiendo de la naturaleza de los datos, se debe seleccionar un modelo de aprendizaje automático apropiado. El conocimiento del dominio puede ser útil al ajustar el modelo.

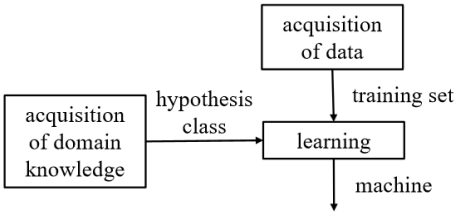


FIGURA 2.2: Metodología de aprendizaje automático que integra el conocimiento del dominio durante la selección del modelo.

2.3.2. Red Neuronal Artificial (ANN)

Una red neuronal es un conjunto de nodos. Generalmente los nodos se encuentran densamente conectados. Cada nodo contiene un algoritmo para procesar las entradas que recibe. El algoritmo dentro de cada una de estas unidad de procesamiento se conoce como "Perceptrón". El Perceptrón corresponde al modelo matemático asociado a una neurona biológica [21].

Una Red Neuronal Artificial (ANN, por sus siglas en inglés) busca modelar artificialmente la comunicación y la estructura de las neuronas en un cerebro biológico. En la Figura 2.3 se muestra el diagrama representativo para una red neuronal de dos capas, con una sola capa oculta.

En la Figura 2.3 cada nodo circular representa una neurona artificial y una flecha representa una conexión desde la salida de una neurona artificial a la entrada de otra. En el modelo de la ANN la señal de salida transmitida por las neuronas artificiales se obtiene mediante la aplicación de algún tipo de función de activación no lineal, luego de ser

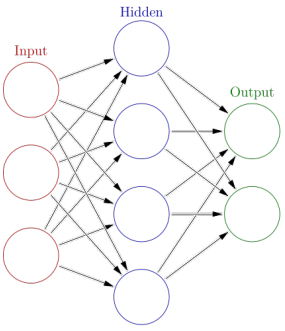


FIGURA 2.3: Diagrama de una ANN. Usado bajo los permisos de la licencia CC BY-SA 3.0.

aplicada la función de activación produce un número real. Generalmente esta función de activación es la función sigmoide σ dada por la Ecuación 2.1.

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{2.1}$$

Donde z es una suma ponderada de cada uno de los valores de entrada x.

$$z(\mathbf{x}) = \mathbf{W}\mathbf{x} + b \tag{2.2}$$

$\mathbf{W}$ y $\mathbf{x}$ son matrices renglón y columna, respectivamente. Ambas matrices poseen una cardinalidad igual a la longitud del vector que almacena los datos de entrada, con los cuales se desea entrenar a la red neuronal. El vector $\mathbf{W}$ almacena los pesos que posteriormente son ajustados durante el proceso de aprendizaje mediante propagación hacia atrás.

Una capa oculta se forma a partir de una colección de neuronas artificiales. La señal de entrada es procesada secuencialmente por cada una de las capas o "hidden layers", de izquierda a derecha. Si se tienen muchas capas en una ANN entonces se le describe empleando el adjetivo "deep". Si se tienen pocas capas en la ANN entonces se le describe empleando el adjetivo "shallow".

2.3.3. Red Neuronal Recurrente (RNN)

Una Red Neuronal Recurrente (RNN, por sus siglas en inglés) busca modelar el proceso de aprendizaje en donde se adquiere nuevo conocimiento, a partir de conocimiento previo del mismo tema o temas relacionados. Por ejemplo, considere la estrategia didáctica sobre el uso de analogías por parte del maestro en el aula de clases. Este modelo sugiere que el ser humano al procesar nueva información mantiene un estado interno que representa

información adquirida anteriormente, o bien, dicho estado interno corresponde a lo que llamamos memoria humana.

Una RNN tiene la capacidad de recordar conocimiento adquirido previamente mientras que aprende nueva información. Una RNN logra recordar mediante el establecimiento de una conexión cronológica horizontal entre las diferentes capas que la forman. De tal manera que la información previa persiste dentro del estado interno en cada una de las capas a lo largo de su ejecución. En la Figura 2.4 se muestra el diagrama de una RNN tradicional.

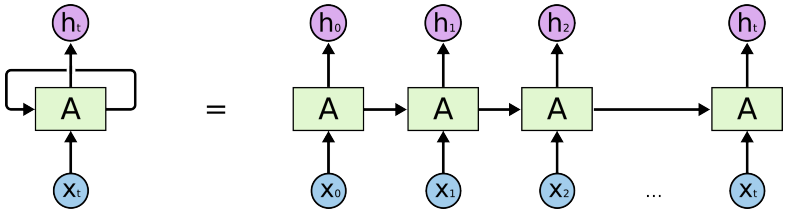


FIGURA 2.4: Diagrama de una RNN tradicional.

En la figura 2.4 los rectángulos verdes corresponden a capas o "hidden layers". Los círculos azules y morados corresponden a el carácter en alguna posición t dentro de la secuencia de entrada y en la secuencia de salida, respectivamente. En este diagrama se asume que tanto la longitud de la secuencia de entrada como la longitud de la secuencia de salida tienen el mismo tamaño. Debido a la forma en que las capas están conectadas en una RNN, éstas están íntimamente relacionadas con secuencias o series de tiempo, o bien, con cualquier tipo de secuencia donde los datos se registran en intervalos regulares, ya sea en el dominio del tiempo o alguna otra variable. Así, la arquitectura de la RNN está hecha *ad-hoc* para las series de tiempo.

En los últimos años, las RNN han obtenido un éxito increíble en aplicaciones tales como: reconocimiento de voz, modelado de lenguaje natural, traducción de textos, subtítulos de imágenes, generación de texto y música, entre otros.

Otra característica importante de una RNN es que a diferencia de las Redes Neuronales Convolucionales, las RNN permiten operar sobre secuencias de diferente tamaño en los vectores para la entrada y salida. Los diferentes tipos de RNN más comunes y una ANN tradicional se muestran en la Figura 2.5.

En la Figura 2.5 cada rectángulo es un vector y las flechas representan funciones, por ejemplo: multiplicación de matrices. Los vectores de entrada están en rojo, los vectores de salida están en azul y los vectores verdes mantienen el estado interno de cada una de las redes neuronales. Así mismo, se observan los siguientes tipos de redes neuronales, de izquierda a derecha:

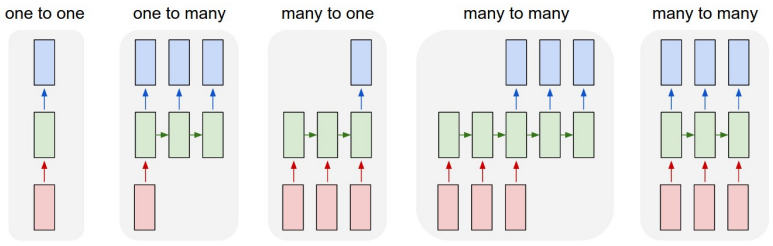


FIGURA 2.5: Tipos de RNNs.

1. ANN tradicional. Secuencia de entrada de tamaño fijo a secuencia de salida de tamaño fijo. Por ejemplo: clasificación de imágenes.

2. RNN uno a muchos. Secuencia de entrada de tamaño fijo a secuencia de salida de tamaño variable. Por ejemplo, subtítulos autogenerados donde se recibe una secuencia de audio y produce una secuencia de texto.

3. RNN muchos a uno. Secuencia de entrada de tamaño variable a secuencia de salida de tamaño fijo. Por ejemplo: análisis del sentimiento, donde una oración determinada se clasifica como que expresa un sentimiento positivo o negativo.

4. RNN muchos a muchos. Secuencia de entrada y salida de tamaños variables, donde la secuencia de salida genera nuevos datos. Por ejemplo: traducción automática, una RNN lee una oración en inglés y luego genera una oración en francés.

5. RNN muchos a muchos. Secuencia de entrada y salida sincronizada. Por ejemplo, clasificación de vídeo donde deseamos etiquetar cada fotograma del video.

En todos los casos descritos anteriormente no hay restricciones pre especificadas en longitudes de las secuencias específicas, porque cada capa en la RNN se aplica a cada elemento que forma parte de la secuencia de salida o entrada.

En ocasiones, sólo es necesario que la RNN recuerde información reciente para realizar la tarea actual. Por ejemplo, consideremos un modelo de lenguaje natural que intenta predecir la siguiente palabra basándose en las anteriores. Si estamos tratando de predecir la última palabra en la secuencia de entrada “las nubes están en el cielo”, no necesitamos ningún contexto adicional; es bastante obvio que la última palabra será “cielo”. De tal manera que las RNN pueden aprender a usar la información pasada reciente. En la Figura 2.6 se muestra el diagrama de la RNN correspondiente al problema mencionado anteriormente.

En la Figura 2.6 Los valores x_i codifican las primeras palabras en la oración. Los valores h_i son la salida de las funciones de activación en cada una de las capas y corresponden a palabras en el lenguaje humano. X_0 y X_1 representan dos palabras al inicio de la

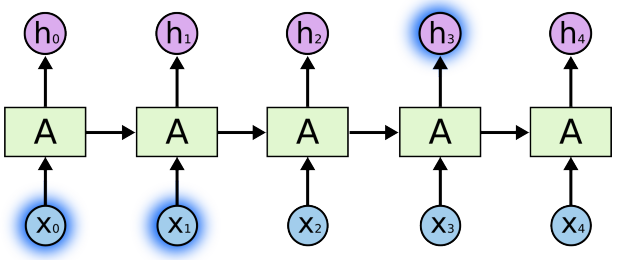


FIGURA 2.6: Red neuronal recurrente tipo muchos a muchos con dependencia a corto plazo.

secuencia a procesar, mientras que h_3 corresponde a una palabra de salida al procesar la oración formada por las palabras X_3, X_2 X_1 y X_0 y la información más relevante para calcular la palabra en la salida h_3 se encuentra en las palabras X_0 y X_1.

Sin embargo, si se requiere que la RNN recuerde información pasada no reciente puede ser que no sea posible entrenar la RNN a un nivel aceptable. Por ejemplo, consideremos, nuevamente, un modelo de lenguaje natural que intenta predecir la siguiente palabra basándose en las anteriores. Si estamos tratando de predecir la última palabra en la secuencia de entrada "Crecí en Francia ... hablo francés con fluidez". La información pasada reciente sugiere que la siguiente palabra es probablemente el nombre de un idioma, pero si se busca determinar con precisión a qué idioma se refiere, entonces se requiere acceso a la información pasada no reciente donde aparece la palabra "Francia".

Si la cantidad de palabras entre "Francia" y "hablo" es muy grande es posible que la RNN no pueda recordar que durante la información pasada no reciente se encontraba la palabra "Francia". Por lo tanto, es probable que no encuentre correctamente la palabra "francés" que corresponde al idioma esperado. En la Figura 2.7 se muestra el diagrama de la RNN correspondiente a este caso.

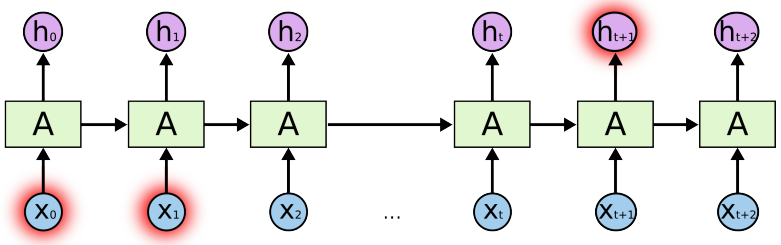


FIGURA 2.7: Red neuronal recurrente tipo muchos a muchos con dependencia a largo plazo.

En la Figura 2.7 los valores x_i codifican las primeras palabras en la oración, sin embargo estos valores se encuentran en información muy antigua, por lo tanto resulta que su impacto en los valores h_i es mucho menor. De acuerdo al diseño de las RNN se esperaría que fuera capaz de recordar dependencias a largo plazo, desafortunadamente, en la práctica,

las RNN no parecen ser capaces de recordar dichas dependencias de largo plazo, o bien, no parecen ser capaces de aprender que deben recordar dichas dependencias de largo plazo.

2.3.4. Long Short Term Memory (LSTM)

Las redes tipo Long Short Term Memory (LSTM, por sus siglás en inglés), son un tipo de RNN. LSTM cuenta con la capacidad de aprender las dependencias a largo plazo que resultan problemáticas para una RNN tradicional. El modelo LSTM fue introducido por Hochreiter y Schmidhuber en 1997 [25]. Hoy en día el modelo LSTM funciona muy bien en una gran variedad de problemas y son utilizadas ampliamente.

El modelo LSTM está diseñado explícitamente para resolver el problema de dependencia a largo plazo. Su comportamiento predeterminado es aprender a recordar información durante largos períodos de tiempo, o bien, olvidar las dependencias a largo plazo a favor de dependencias a corto plazo, cuando las dependencias a corto plazo resultan en el comportamiento deseado.

Todas las RNN tienen una estructura que consiste en una cadena de módulos repetidos cada uno de los cuales contiene una red neuronal. En la Figura 2.8 se muestra una RNN tradicional, donde la función de activación asociada a cada neurona artificial está dada por $\tanh(z)$, z se definió en la Ecuación 2.2

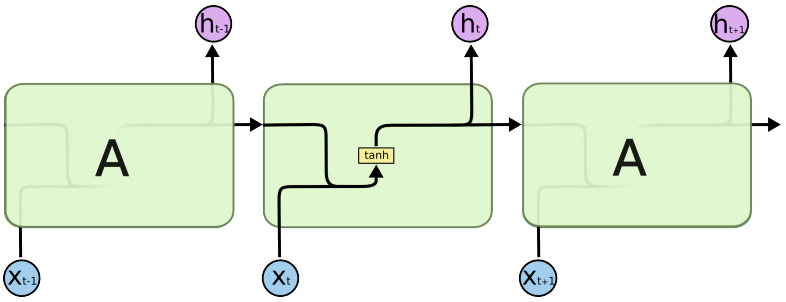


FIGURA 2.8: Diagrama de una RNN tradicional donde cada una de las redes neuronales que la forman tiene asociada una función de activación dada por *tanh*.

El modelo LSTM tienen una estructura similar a una RNN tradicional, la diferencia entre una RNN tradicional y la red LSTM consiste en el diseño de la estructura interna de cada una de las redes neuronales que la forman al modelo LSTM. En la Figura 2.9 se muestra el diseño de la estructura interna para un módulo de red neural.

En la Figura 2.9 los círculos rosas representan operaciones puntuales, como la suma de vectores, mientras que los cuadros amarillos son funciones de activación. Las líneas que se fusionan denotan concatenación, mientras que una bifurcación de líneas indica que su contenido se está copiando y las copias van a diferentes ubicaciones.

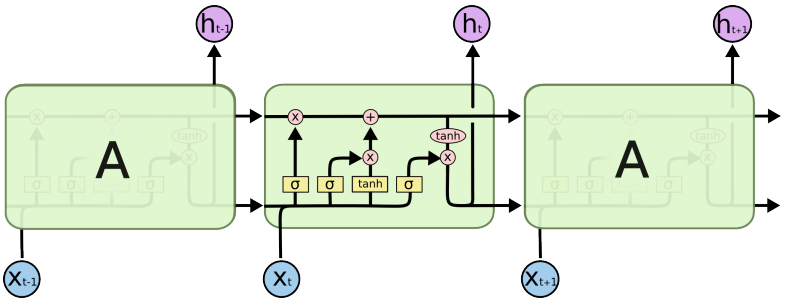


FIGURA 2.9: Diagrama del modelo LSTM.

La característica particular que permite al modelo LSTM recordar dependencias de largo plazo es debido a la línea horizontal superior que atraviesa el diagrama mostrado en la Figura 2.10. A esta línea horizontal superior se le asigna el nombre de "estado interno" y se representa mediante la variable C.

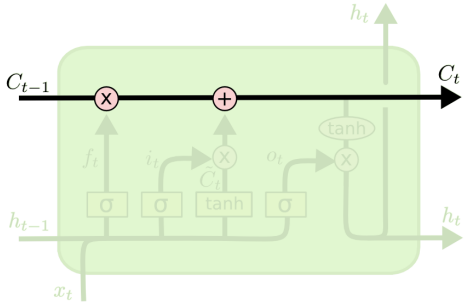


FIGURA 2.10: C_t representa el estado interno de la red neuronal para el elemento en la posición x_t en la secuencia de entrada.

El modelo LSTM realiza operaciones tales que es capaz de eliminar o agregar información al estado interno de la red neuronal, uno de estos mecanismos se presenta en la Figura 2.11. El mecanismo de eliminación o actualización es controlado mediante operaciones matemáticas que en su conjunto reciben el nombre de "compuertas". Las compuertas proporcionan el mecanismo a través del cual los módulos que aparecen antes pueden compartir su estado interno con módulos de redes neuronales posteriores, es decir, le permiten al LSTM recordar dependencias de largo plazo.

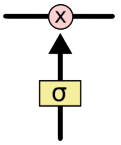
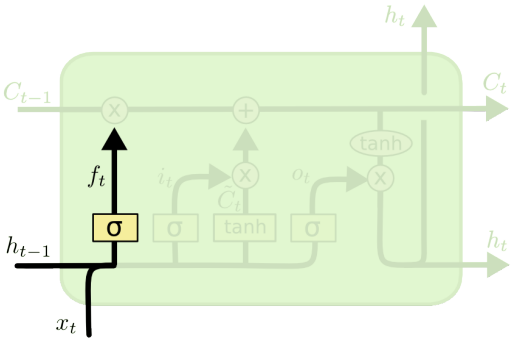


FIGURA 2.11: Función de activación sigmoide, que recibe como entrada un elemento x_t de la secuencia a procesar y su salida es multiplicada elemento a elemento con la información almacenada en C_{t-1}.

Las compuertas tipo sigmoide, mostradas en la Figura 2.11 generan números entre cero y uno, que describen cuánta información debe dejarse pasar. Un valor de cero significa "no dejar pasar nada", mientras que un valor de uno significa "dejar pasar todo". El

modelo LSTM tiene tres de estas puertas para proteger y controlar el estado de la unidad de procesamiento.

La compuerta "forget gate" regula la cantidad de información que debe ser eliminada del estado interno en cada módulo LSTM. Se aplica una función de activación tipo sigmoide sobre los vectores $\mathbf{h_{t-1}}$ y $\mathbf{x_t}$. Para estas compuertas un valor de "1" implica "recordar todo" mientras que un valor de "0" implica "olvidar todo". Por ejemplo, suponga una secuencia de entrada que consiste en dos oraciones, en la primera el sujeto es femenino y en la segunda el sujeto es masculino, al terminar de analizar la primera oración en dicha secuencia el modelo LSTM debe olvidar el género femenino para poder usar los pronombres correctos mientras analiza la segunda oración. En la Figura 2.12 se muestran las operaciones realizadas por la compuerta "forget gate".



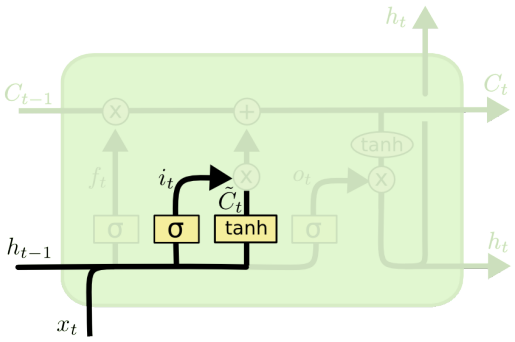
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

FIGURA 2.12: Diagrama de la compuerta "Forget gate".

La compuerta tipo "update gate" regula la cantidad de información nueva C_t que será añadida a la información actual C_{t-1} en el estado interno del módulo, para luego ser compartido con el siguiente módulo. Este proceso tiene dos etapas. Primero, una capa sigmoidea llamada "input gate" i_t decide qué valores actualizaremos. A continuación, una función de activación tipo *tanh* crea un vector de nuevos valores candidatos $\tilde{C}_t$, finalmente, se combinan ambas compuertas para generar $\tilde{C}_t$. Por ejemplo, suponga una secuencia de entrada que consiste en dos oraciones, en la primera el sujeto es femenino y en la segunda el sujeto es masculino, al terminar de analizar la primera oración en dicha secuencia, el modelo LSTM debe actualizar la información de su estado interno, de tal manera que recuerde el nuevo género masculino al momento de procesar la segunda oración. En la Figura 2.13 se muestran las operaciones realizadas por la compuerta "update gate".

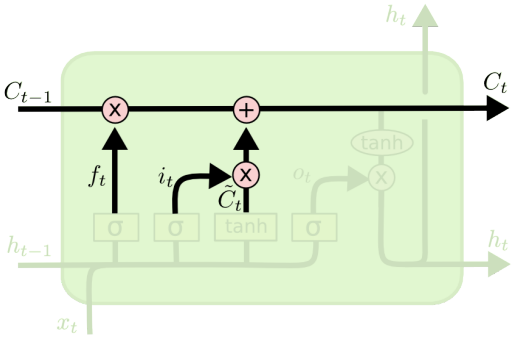
El nuevo estado interno C_t es compartirlo con el siguiente módulo LSTM. En la Figura 2.14 se observa cómo el modelo anterior en el modelo LSTM comparte su estado interno con el módulo posterior.

Finalmente, la compuerta "output gate" genera el valor correspondiente en la secuencia de salida. Una copia de este valor de salida es el que el usuario observa, mientras que otra



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

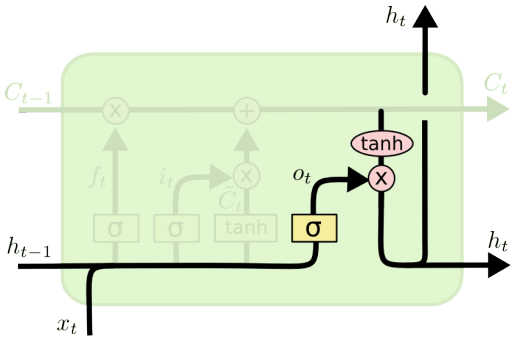
FIGURA 2.13: Diagrama de la compuerta "Update gate".



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

FIGURA 2.14: Diagrama sobre la actualización del estado interno C_t.

copia es enviada al siguiente módulo en la secuencia de módulos LSTM. En la Figura 2.15 se muestran el conjunto de operaciones realizadas por la compuerta de salida.



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

FIGURA 2.15: Diagrama sobre la generación de la variable de salida h_t.

2.3.5. Red Neuronal Convolucional LSTM (convLSTM)

La arquitectura de red neuronal LSTM es eficaz durante la identificación de correlación temporal en series de tiempo. Sin embargo, el modelo LSTM no es eficaz durante la identificación de correlación espacial. O bien, el modelo LSTM resulta inconveniente para el manejo de datos espacio-temporales. El modelo convLSTM busca resolver este problema. En el modelo convLSTM se propone una extensión del modelo LSTM en la

cual se tienen estructuras convolucionales en las transiciones de entrada a estado y de estado a estado.

Las ecuaciones del modelo ConvLSTM se muestran a continuación. El operador $*$ denota convolución y el operador $\circ$ denota el producto Hadamard.

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ \tanh(C_t)
\end{aligned}
$$

Una contribución importante del modelo convLSTM se refiere al hecho de que las entradas X_i, las salidas de cada módulo C_i, los estados ocultos H_i y las compuertas i_t, f_t y o_t son tensores de rango 3, donde las últimas dos dimensiones son dimensiones espaciales, por ejemplo, podemos visualizarlas como vectores de atributos distribuidos en una cuadrícula espacial.

En la Figura 4.5 se muestra un ejemplo de la arquitectura del modelo convLSTM, así como el proceso, por medio del cual el modelo procesa la entrada y genera una salida.

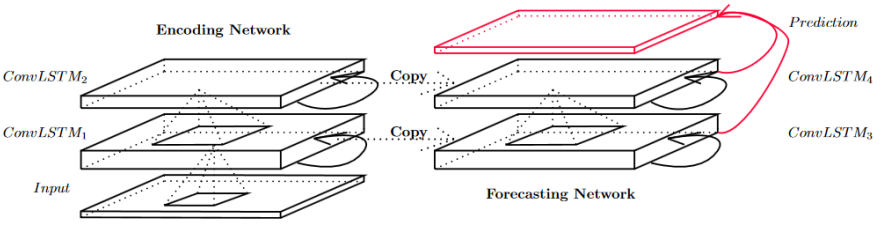


FIGURA 2.16: Proceso de codificación-pronóstico en un modelo ConvLSTM con cuatro capas.

El ConvLSTM determina el estado futuro de una determinada celda en la cuadrícula mediante las entradas y los estados pasados de sus vecinos locales. Esto se puede lograr fácilmente mediante el uso de un operador de convolución en las transiciones de estado a estado y de entrada a estado.

2.4. Tensores en Deep Learning

En el aprendizaje profundo es común ver mucha discusión sobre los tensores como la estructura fundamental de datos. Tensor incluso aparece en el nombre de la biblioteca de aprendizaje automático de Google: "TensorFlow". Los tensores son un tipo de estructura de datos que se utiliza en álgebra lineal y, al igual que los vectores y las matrices, se pueden realizar operaciones aritméticas con tensores.

Es posible considerar a un tensor como la generalización de una matriz. Por ejemplo: un tensor de rango cero es un escalar, un tensor de rango uno es un vector o una matriz columna o renglón, un tensor de rango dos es una matriz de tamaño $n \times m$, un tensor de rango tres es como si tuviéramos una matriz de tamaño $i \times j \times k$.

2.4.1. Tensores en Python

Al igual que los vectores y las matrices, los tensores se pueden representar en Python utilizando un arreglo n-dimensional de números, mediante el tipo de variable llamado "ndarray". En Python un tensor se puede definir empleando el constructor numpy.array() cuyo argumento es una lista de listas.

Por ejemplo: un tensor de rango tres con dimensiones $2 \times 3 \times 1$ en Python se define como sigue:

```
#In:
#create tensor
from numpy import array
T = array([ [[7], [11], [5]],
[[3], [2], [6]] ])
```

El tensor tiene las dimensiones asociadas a una secuencia de matrices. Este tensor de rango 3, la primera dimensión define el nivel de profundidad, la segunda dimensión define la fila y la tercera dimensión define la columna. En el contexto de un lenguaje de programación la forma de este tensor se obtiene como T.shape y es igual a (2,3,1).

2.4.2. Tensores en TensorFlow

Los tensores en Tensor Flow continúan la idea de tensores en python empleando una estructura similar a "numpy.array()" en Python. En Tensorflow todos los tensores son

inmutables como las tuplas de Python, es decir, no se puede actualizar el contenido de un tensor, solo crear uno nuevo.

Por ejemplo, podemos crear tensores de diferentes formas con valores iguales a cero, de la siguiente manera.

- Tensor, rango 0 (escalar): tf.constant(0), shape=().
- Tensor, rango 1 (vector): tf.zeros(3), shape=(3,).
- Tensor, rango 2 (matriz): tf.zeros([3,2]), shape=(3, 2), dtype=float16.

En la Figura 2.17 se muestra una representación visual de tensores de rango 0, 1, y 2.

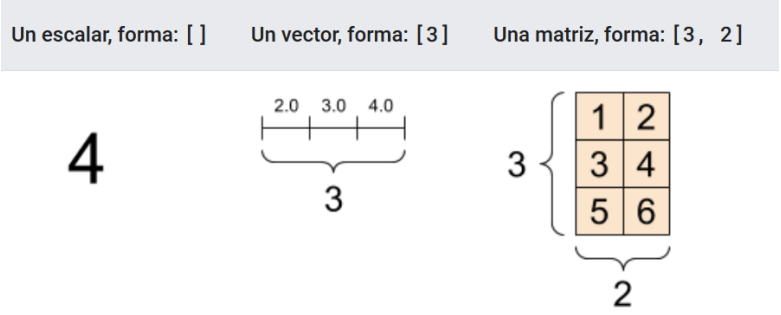


FIGURA 2.17: Ejemplos de tensores de diferentes tamaños y sus formas.

- Tensor, rango 3 (cubo): tf.zeros([3,2,5]), shape(3,2,5)

En la Figura 2.18 se muestran varias formas en las que podemos representar visualmente un tensor de rango 3.

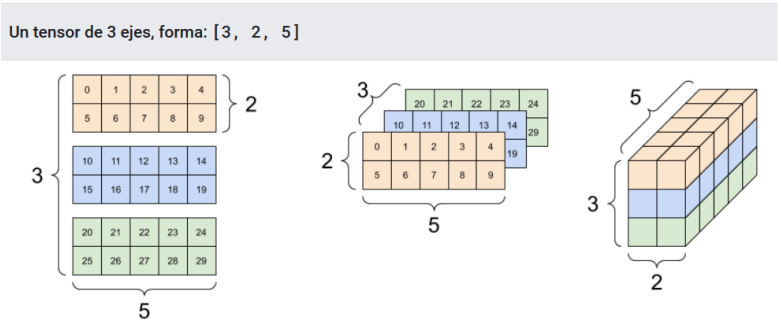


FIGURA 2.18: Visualización de un tensor de rango 3.

- Tensor, rango 4 (conjunto de cubos): tf.zeros([3, 2, 4, 5]). shape = (3, 2, 4, 5)

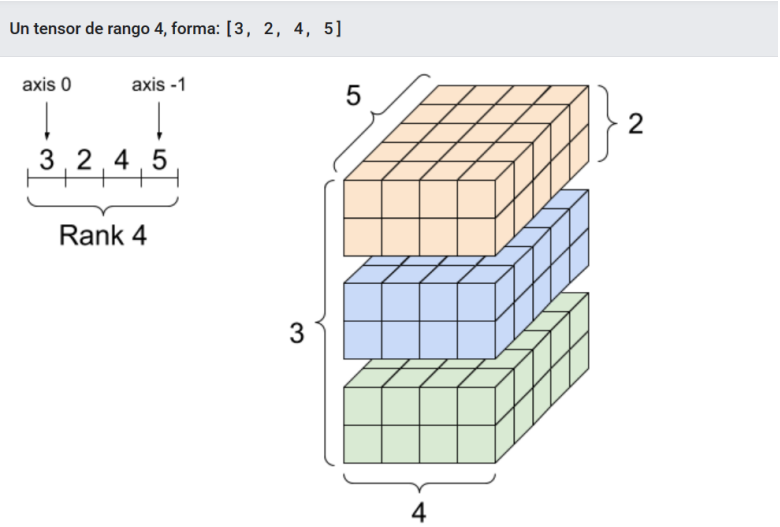


FIGURA 2.19: Visualización de un tensor de rango 4.

En la Figura 2.19 se muestra una visualización de este tipo de estructura de números.

Los elementos dentro de un tensor pueden ser números reales, enteros, complejos o caracteres. La clase base de Tensorflow, tf.Tensor, requiere que los tensores sean "rectangulares", es decir, a lo largo de cada eje, todos los elementos tienen el mismo tamaño. Sin embargo, existen tipos especializados de tensores que pueden manejar diferentes formas, tales como tensores irregulares y tensores escasos.

2.5. Trabajos relacionados

Las fuerzas policiales en varias ciudades ya están aplicando tecnologías diseñadas para predecir crímenes.

PredPol: En base a un historial de crímenes, señala las zonas donde puede ocurrir un crimen a determinada hora basándose en tres tipos de datos: tipo, lugar y tiempo. PredPol se basa en una década de investigación académica detallada sobre las causas de la formación de patrones de delincuencia. Esa investigación relaciona con éxito varios aspectos clave del comportamiento del delincuente, con una estructura matemática que se utiliza para predecir cómo evolucionarán los patrones delictivos día a día, momento a momento [18].

RTM: Diagnóstica condiciones ambientales que conducen a la delincuencia. Por ejemplo, RTM identifica qué características del entorno atraen la delincuencia. Este diagnóstico hace pronósticos precisos que la policía usa para desplegar recursos, prevenir delitos y reducir riesgos. Se concentra en los lugares, no en las personas. RTM agrega el por

qué al dónde. El análisis RTM reúne múltiples fuentes de datos al conectarlos a lugares geográficos. Agrega contexto a "big data" y pronostica nuevos patrones de riesgo para ciertas áreas. Con RTM, los funcionarios de la ciudad saben el "dónde y el porqué" del crimen, y qué hacer cuando llegan allí para abordarlo, sin los daños de la vigilancia excesiva. Se ha comprobado que RTM reduce las tasas de criminalidad y mejora las relaciones con la comunidad [19].

Hunch Lab: Es un sistema de gestión de patrulla proactivo basado en la web. Los modelos estadísticos avanzados pronostican cuándo y dónde es probable que surjan delitos. Pero no se trata sólo de anticipar el crimen, se trata de encontrar la mejor manera de responder. Las tácticas de vigilancia no solo deben ser efectivas, sino también reflejar las prioridades de la comunidad. Hunch Lab proporciona características que: (1) alinean las actividades de patrulla con las prioridades de la comunidad, (2) asigna recursos de manera inteligente para evitar la vigilancia excesiva y (3) determinan que tácticas funcionan y cuáles no. Hunch Lab utiliza datos históricos de crímenes como Predpol y RTM pero añaden un conjunto de factores extraídos del calendario como vacaciones, eventos deportivos, festivales, previsión meteorológica, entre otros [20].

Capítulo 3

Diseño del sistema

3.1. Introducción

Se realiza una descripción de la metodología a implementar, descripción de los diagramas de análisis del sistema, el diagrama de diseño y por último, el modelo de datos y la arquitectura del sistema.

3.2. Metodología

Para el desarrollo de esta investigación se plantearon tres fases, tal y como se muestra en la Figura 3.1.

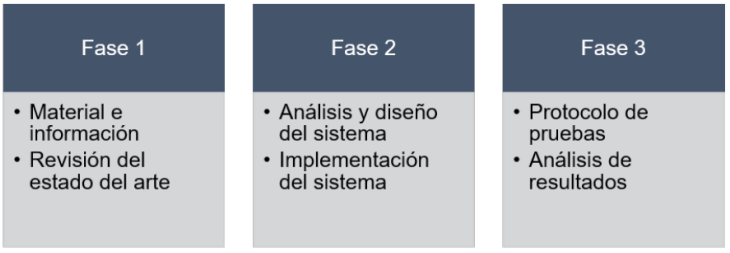


FIGURA 3.1: Metodología del proyecto

En la primera fase, se describe la problemática actual, con el material e información que se obtuvo a partir de la base de datos criminal de la ciudad de Chicago en E.U. y se realiza una revisión exhaustiva del estado del arte que se tiene al momento de plantear este proyecto. La segunda fase, corresponde al análisis y diseño del sistema, en este apartado se realiza la propuesta general sobre la que se trabajará para implementar un sistema que solvente la problemática principal. Así mismo, se incluye la implementación

del sistema. En la tercera fase, se realizan las pruebas de funcionamiento y análisis de resultados obtenidos en base al conjunto de datos utilizados y la ingeniería aplicada.

3.3. Desafíos técnicos en el desarrollo de una policía predictiva

Existen varios desafíos técnicos en el desarrollo de un modelo predictivo para determinar los tiempos y las zonas con mayor riesgo de que ocurran futuros delitos.

3.3.1. ¿Qué datos predicen mejor la probabilidad de crímenes futuros?

Algunas teorías y modelos han propuesto utilizar el clima, las tasas de desempleo, los precios de la vivienda, el nivel educativo, la disparidad de ingresos, la prevalencia del uso ilícito de drogas, los días de la semana, la colocación de "atractores de delitos", como estaciones de autobuses o estacionamientos de centros comerciales, entre otros.

Encontrar la información que se desea usar es la primera mitad del problema de datos; la otra mitad es obtener acceso continuo a una fuente de datos criminales oportuna, precisa y granular. Los pronósticos del tiempo, por ejemplo, se generan a diario, pero no siempre son lo suficientemente precisos o granulares. Los datos de desempleo pueden ser precisos, pero solo se generan una vez al mes teniendo alta granularidad. Los precios de la vivienda, aunque granulares, se generan sólo periódicamente y, por lo tanto, no son muy oportunos.

3.3.2. Encontrar un modelo que pueda hacer predicciones precisas

El modelo a seleccionar depende del tipo de datos con los que se dispone. Un modelo robusto debe ser capaz de predecir una variedad de delitos, desde delitos contra la propiedad hasta delitos violentos, así mismo debe tener un nivel constante de precisión para los tipos de delitos a predecir. El modelo debe funcionar bien en cualquier momento, ya sea entre semana o los fines de semana, días o noches e invierno o verano.

3.3.3. Métodos robustos para el registro de datos

Es necesario incorporar métodos resistentes para introducir nuevos datos criminales en el sistema, con indicadores de error y escenarios de recuperación para cuando las fuentes de datos estén alteradas o no estén disponibles.

- Se debe garantizar que la calidad de los datos se mantenga constante.
- Debe ejecutarse en un entorno donde la información esté disponible en todo momento, todos los días del año .
- El sistema debe ser escalable para evitar interrupciones en el servicio cuando muchos usuarios acceden al sistema.
- Se debe asegurar que los datos se cifren durante la transferencia y el almacenamiento.
- Se debe establecer niveles de acceso y transacciones basados en roles de acuerdo a cada tipo de usuario.
- Se debe mantener un registro de todas las consultas realizadas a la base de datos.

3.3.4. Consideraciones sobre el diseño del sistema

Se debe encontrar una manera de cómo presentar las predicciones generadas por el sistema, como recomendaciones de zonas en las cuales existe un alto y bajo riesgo de incidencia criminal. Por ejemplo:

- ¿Serán proporcionados mapas de calor o puntajes numéricos o clasificaciones?
- ¿Cubre toda la ciudad o sólo resalta zonas específicas de alto riesgo?
- ¿Hace predicciones a nivel de ciudad o estado o hace recomendaciones para zona de patrullaje de cada oficial?
- ¿Qué crímenes está pronosticando?
- ¿Pueden variar las predicciones según el día de la semana, la hora del día o la zona de patrullaje?

El acceso a la información relevante debe ser seguro pero simple, sin la necesidad de que el usuario final tenga que aprender a dominar una nueva tecnología.

3.4. Diagrama de nivel superior: nivel 1

Un diagrama de flujo de datos (DFD) traza el flujo de información para cualquier proceso o sistema. Los diagramas de flujo de datos pueden variar desde descripciones generales de procesos simples, incluso dibujadas a mano, hasta DFD detallados de varios niveles que

profundizan progresivamente en cuanto a cómo los datos son manipulados. Se pueden utilizar para analizar un sistema existente o modelar uno nuevo. Como todos los mejores diagramas y gráficos, un DFD a menudo puede "decir" visualmente cosas que serían difíciles de explicar con palabras, y funcionan para audiencias tanto técnicas como no técnicas.

Usando las reglas o pautas en un DFD de cualquier convención, los símbolos representan los cuatro componentes de los diagramas de flujo de datos.

1. Entidad externa: un sistema externo que envía o recibe datos, comunicándose con el sistema que se está diagramando. Son las fuentes y destinos de la información que entra o sale del sistema. Pueden ser una organización o persona externa, un sistema informático o un sistema empresarial. También se les conoce como terminadores, fuentes y sumideros o actores. Por lo general, se dibujan en los bordes del diagrama.

2. Proceso: cualquier proceso que cambia los datos y produce una salida. Puede realizar cálculos, ordenar datos según la lógica o dirigir el flujo de datos según las reglas comerciales. Se utiliza una etiqueta corta para describir el proceso, como "Enviar pago".

3. Almacén de datos: archivos o repositorios que contienen información para su uso posterior, como una tabla de base de datos o un formulario de membresía. Cada almacén de datos recibe una etiqueta simple, como "Pedidos".

4. Flujo de datos: la ruta que toman los datos entre las entidades, procesos y repositorios de datos externos. Representa la interfaz entre los otros componentes y se muestra con flechas, generalmente etiquetadas con un nombre de datos corto, como "Detalles de facturación".

En la Figura 3.2 se muestra el flujo de datos, los procesos que manipulan los datos y los agentes que interactúan con la información generada.

3.5. Modelo de datos

Un modelo de datos es un modelo abstracto que organiza elementos de datos y estandariza cómo se relacionan entre sí y con las propiedades de entidades del mundo real. Un modelo de base de datos es una especificación que describe cómo se estructura y utiliza una base de datos. En nuestro caso se describe el modelo de datos mediante un modelo relacional, el cual tiene como objetivo proporcionar un método declarativo para especificar datos y consultas: los usuarios declaran directamente qué información

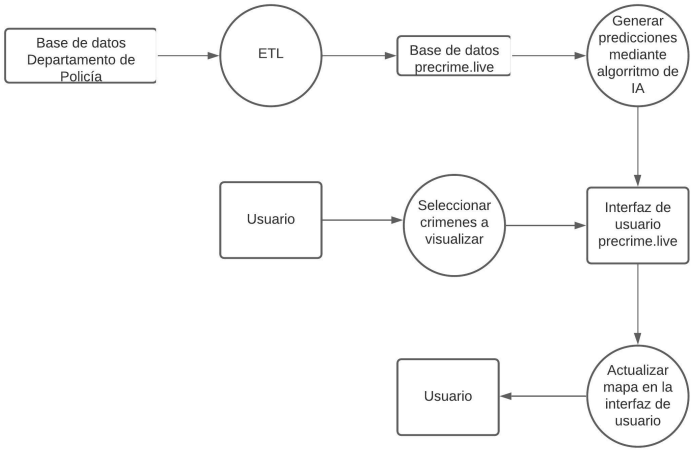


FIGURA 3.2: Diagrama de nivel superior: Nivel 1. Procesos del proyecto

contiene la base de datos y qué información quieren de ella, y dejan que el software del sistema de gestión de la base de datos se encargue de describir las estructuras de datos para almacenar la información. procedimientos de recuperación y datos para responder consultas.

En la Figura 3.3 se muestra el modelo relacional tipo Entidad-Relación empleado para describir el modelo de datos correspondiente a los datos proporcionados por el Departamento de Policía de Chicago.

Un almacén de datos es un sistema que se utiliza para la presentación de informes y el análisis de datos, y se considera un componente central de la inteligencia empresarial. Los almacenes de datos son repositorios centrales de datos integrados de una o más fuentes dispares. Almacenan datos actuales e históricos en un solo lugar que se utilizan para crear informes analíticos para los trabajadores de toda la empresa.

En la Figura 3.4 se muestra el almacén de datos empleado para la generación de reportes de acuerdo a los objetivos planteados durante el desarrollo de este trabajo de tesis.

3.6. Arquitectura del sistema

En la Figura 3.5 se muestra la arquitectura del sistema planteado, los diferentes procesos se describen brevemente a continuación. Se realiza un proceso ETL sobre la base de datos del Departamento de Policía de Chicago, el preprocesamiento de datos incluye el modelo de datos, en el contexto de programación, este modelo de datos corresponde a un cubo de datos donde cada uno de sus valores es una función de probabilidad $P = P(t, u, v)$. Donde las variables t, u y v corresponden a tiempo, ubicación y tipo de

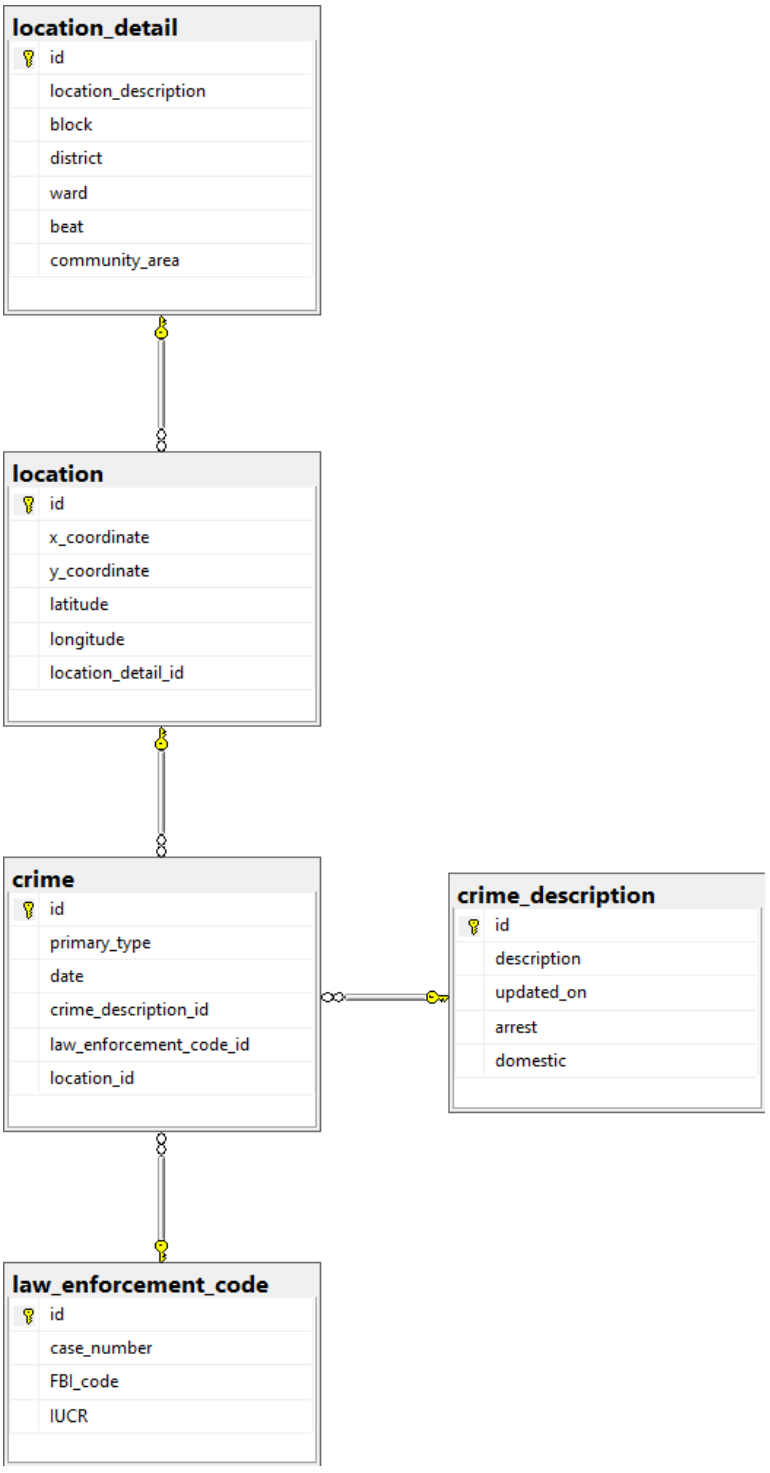


FIGURA 3.3: Diagrama Entidad-Relación de la base de datos del sistema precrime.live

crimen, respectivamente. A partir del cubo de datos es posible generar series de tiempo, el modelo predictivo consiste en un modelo de aprendizaje profundo multicapa, basado en una red neuronal convLSTM, este modelo genera predicciones cuya estructura de datos es, de nuevo, un cubo de datos, estos cubos de datos son alimentados a la base de datos que alimenta a la aplicación web ubicada en *precrime.live*, posteriormente el

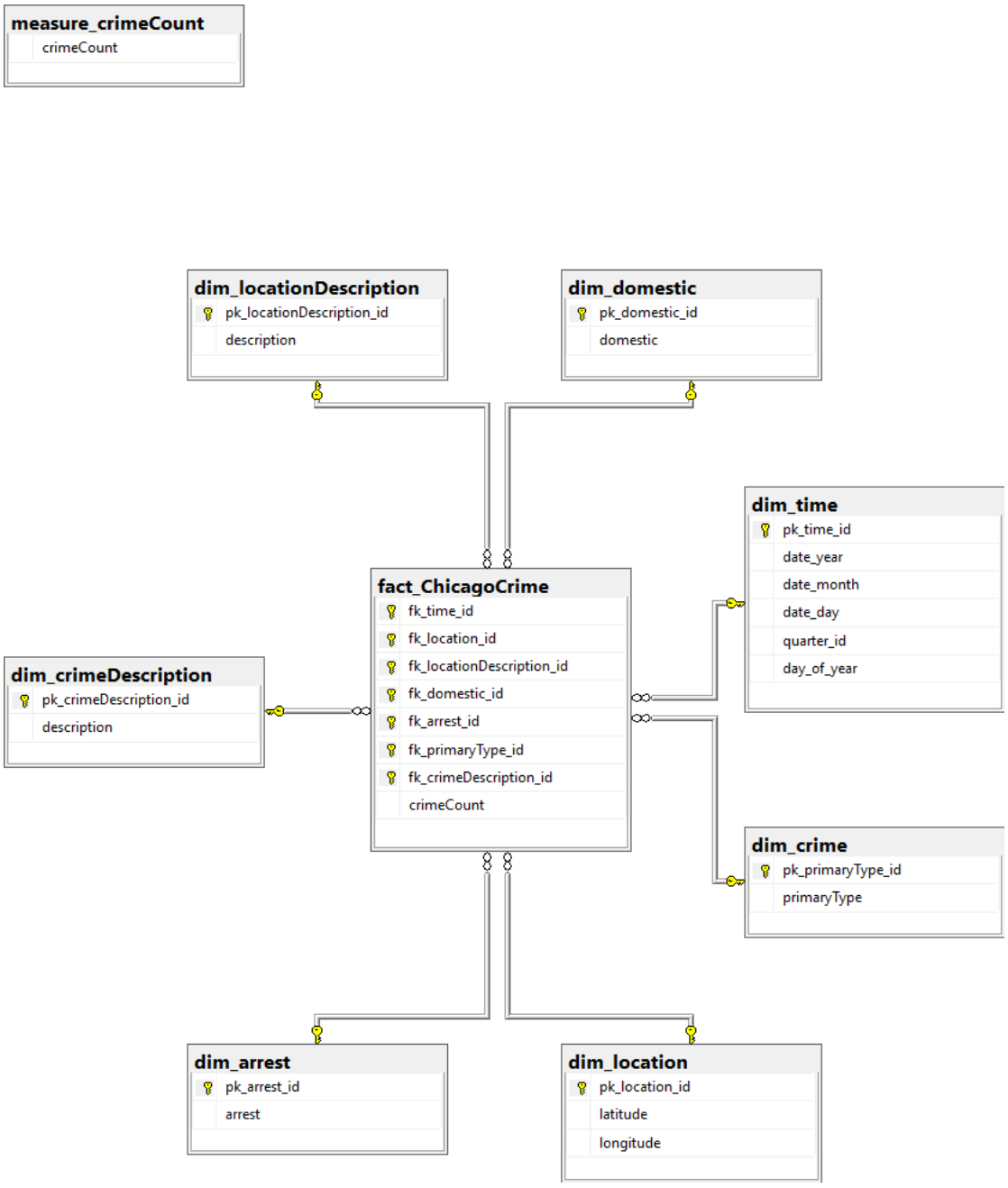


FIGURA 3.4: Almacen de datos del sistema precrime.live

usuario selecciona los tipos de crímenes a visualizar, y la aplicación web consulta la base de datos y actualiza el contenido mostrado al usuario, finalmente el usuario evalúa los resultados presentados y procede a la toma de decisiones.

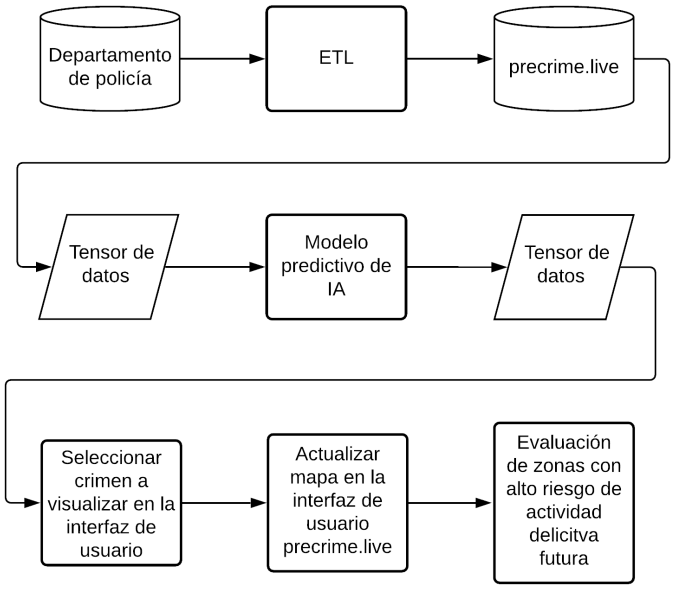


Figura 3.5: Arquitectura del sistema

Capítulo 4

Implementación del sistema

4.1. Introducción

El sistema inteligente para la predicción de crímenes recibe como entrada datos pre procesados mediante un proceso ETL, obtenidos del sistema "Citizen Law Enforcement Analysis and Reporting" (CLEAR, por sus siglas en inglés) del Departamento de Policía de Chicago (CPD, por sus siglas en inglés) [26]. En la primera implementación del sistema inteligente para el reconocimiento de patrones y la predicción de crímenes se optó por el modelo "Long Short Term Memory" (LSTM, por sus siglas en inglés), dado que los datos criminales forman una serie de tiempo para cada tipo de crimen por cada zona en la cual se dividió la ciudad de Chicago. Es claro que el modelo LSTM tiene una ventaja significativa para encontrar correlaciones temporales en series de tiempo [25]. Sin embargo, la naturaleza de los datos criminales, no es sólo temporal, sino espacio-temporal; por lo tanto, posteriormente a la aplicación del modelo LSTM, buscamos un modelo más adecuado para el reconocimiento de patrones espacio-temporales, al finalizar la búsqueda, se encontró que el modelo "Convolutional Long Short Term Memory" (convLSTM, por sus siglas en inglés), es una opción natural para la identificación de patrones espacio-temporales [13]. Así que, a lo largo de la implementación del sistema se emplearon los dos modelos, dados los resultados obtenidos se optó por emplear el modelo convLSTM para el reconocimiento y predicción de patrones criminales. Las predicciones generadas por el algoritmo predictivo se muestran al usuario a través de una aplicación web.

4.2. Modelo LSTM

Esta sección describe la metodología empleada para predecir la secuencia temporal de robos. Específicamente, se muestra la arquitectura propuesta para el sistema de predicción, compuesta por: las fuentes de datos, el módulo de Preprocesamiento de datos, las series de tiempo, el módulo del Modelo predictivo, y finalmente, la predicción de series de tiempo. La Figura 4.1 muestra el diseño propuesto de la arquitectura a utilizar para el sistema de predicción de crímenes propuesto.

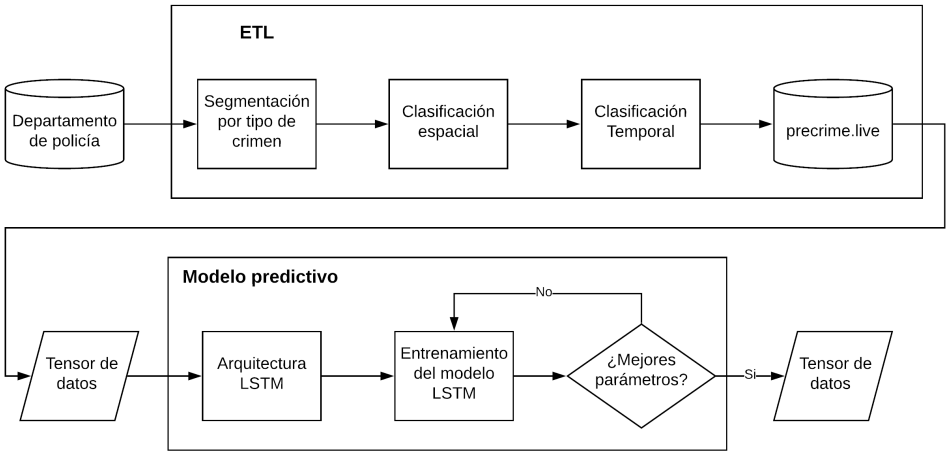


FIGURA 4.1: Arquitectura propuesta para el sistema de predicción de crímenes, utilizando el modelo LSTM. (Elaboración propia).

Se tomó como fuente de datos, el sistema "Citizen Law Enforcement Analysis and Reporting" (CLEAR, por sus siglas en inglés) del Departamento de Policía de Chicago (CPD, por sus siglas en inglés) [26]. Estos datos deberán sufrir un proceso que incluye la extracción, transformación y carga de datos (ETL, por sus siglas en inglés), para convertirlos en un conjunto de datos de calidad, que podrán ser utilizados como soporte a la toma de decisiones. Esta arquitectura está compuesta por dos grandes módulos: Preprocesamiento de datos y Modelo predictivo. A continuación se describe en detalle cada módulo.

Módulo ETL

La base de datos contiene veintidós atributos [26], a partir de los cuales se seleccionaron cuatro: longitud, latitud, tipo de crimen y fecha. Entre los veintidós atributos en la base de datos CLEAR hay valores nulos, sin embargo, no se encontraron registros nulos en los atributos seleccionados, por lo que no fue necesario implementar alguna técnica de imputación de datos; finalmente, se crea una nueva base de datos y se almacenan los datos obtenidos después de aplicar este proceso ETL.

Proceso Segmentación por tipo de crimen. Se selecciona un tipo de crimen para el cual se ha determinado que existe un patrón en su distribución espacio-temporal [16]. De acuerdo a Mohler, et al., el robo se ajusta a una distribución de Poisson, para el número de eventos delictivos que ocurren a una cierta distancia, con respecto a un primer evento dentro de una ventana de tiempo apropiada de acuerdo a los criterios establecidos por el departamento de policía.

Proceso Clasificación espacial. Este proceso mapea longitud y latitud a un número entero, se denota a este número como una zona. Para clasificar la ciudad en zonas, se aplicó el algoritmo k-means logrando 20 zonas, la Figura 4.2 muestra esta clasificación espacial, obtenida sobre la ciudad de Chicago, Illinois.

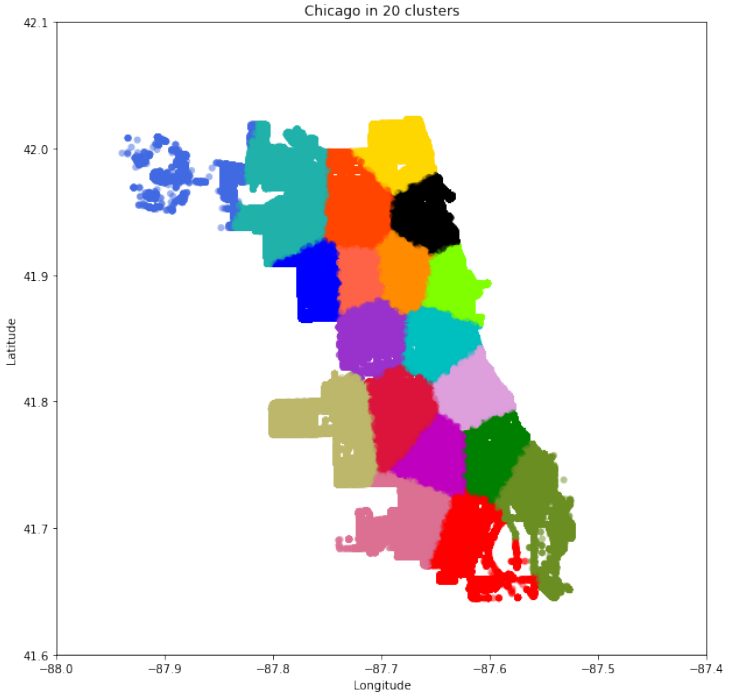


FIGURA 4.2: Clasificación espacial de la ciudad de Chicago en 20 zonas.

La selección del número de zonas depende del nivel de resolución, i. e., el área geográfica por zona que se desee cubrir al calcular la secuencia del número de robos futuros en dicha área geográfica.

Proceso Clasificación temporal. Consiste en determinar una ventana de tiempo, después se calcula la correlación que existe entre la ventana de tiempo seleccionada y el resto de todas las posibles ventanas de tiempo, dentro de la serie de tiempo para el crimen seleccionado [14]. Se define una correlación mínima que deben satisfacer las nuevas ventanas de tiempo encontradas.

Finalmente, se genera una secuencia de entrada para la RNN, que consiste en múltiples ventanas de tiempo altamente correlacionadas, cada ventana de tiempo es una serie de

tiempo. Estas series de tiempo son funciones que reciben como argumento una fecha y regresan una cierta cantidad de crímenes por unidad de tiempo; por cada zona se tiene una serie de tiempo. Se espera que la RNN aprenda a modelar el patrón en las ventanas de tiempo y así obtenga la capacidad de predecir eventos delictivos similares en cada una de las diferentes zonas.

Módulo Modelo predictivo

Arquitectura LSTM. El modelo RNN se basa en la arquitectura LSTM, que utiliza un vector de estado para mantener información sobre las relaciones temporales entre caracteres consecutivos. La salida final del LSTM se pasa como entrada a una capa densa completamente conectada. Se aplica la función de activación softmax sobre esta capa densa, para generar logits, que predicen la probabilidad logarítmica del siguiente carácter en la serie de tiempo a predecir; entre estos caracteres se escoge el más probable para generar la salida de la RNN

En la Figura 4.3 se muestra el diagrama de la arquitectura de la RNN tipo LSTM. Para definir el modelo RNN, se utilizó la API de Keras, específicamente, tf.keras.Sequential. Se tienen L capas que se utilizan para definir el modelo, donde L es mayor o igual que 3. Las tres capas principales son:

tf.keras.layers.Embedding: ésta es la capa de entrada, que consiste en una tabla de búsqueda entrenable que asigna los números de cada caracter posible en la serie de tiempo, a un vector con dimensiones embedding_dim.

tf.keras.layers.LSTM: es la LSTM con una cantidad de unidades rnn_units y función de activación sigmoide.

tf.keras.layers.Dense: es la capa de salida, con un número de salidas igual a vocab_size.

El vocabulario consiste en todos los caracteres únicos que aparecen en la serie de tiempo, y se define como el conjunto, $vocabulario = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$. La capa densa produce un vector de logits de longitud igual a la cantidad de caracteres en el vocabulario, por cada carácter en la secuencia de entrada. A partir de aquí, se mejora el modelo con una arquitectura LSTM apilada [27]. El número de capas LSTM dependerá de la mejora obtenida en la función de pérdida (Loss function) vs el tiempo de entrenamiento requerido.

Proceso Entrenamiento de la RNN. El entrenamiento consiste en darle al modelo la capacidad para responder a la pregunta ¿cuántos robos por unidad de tiempo se esperan en esta zona los siguientes días? Los parámetros de entrenamiento seleccionados son 3 capas LSTM con función de activación sigmoide, $rnn\_units = 2048$, $embedding\_dim =$

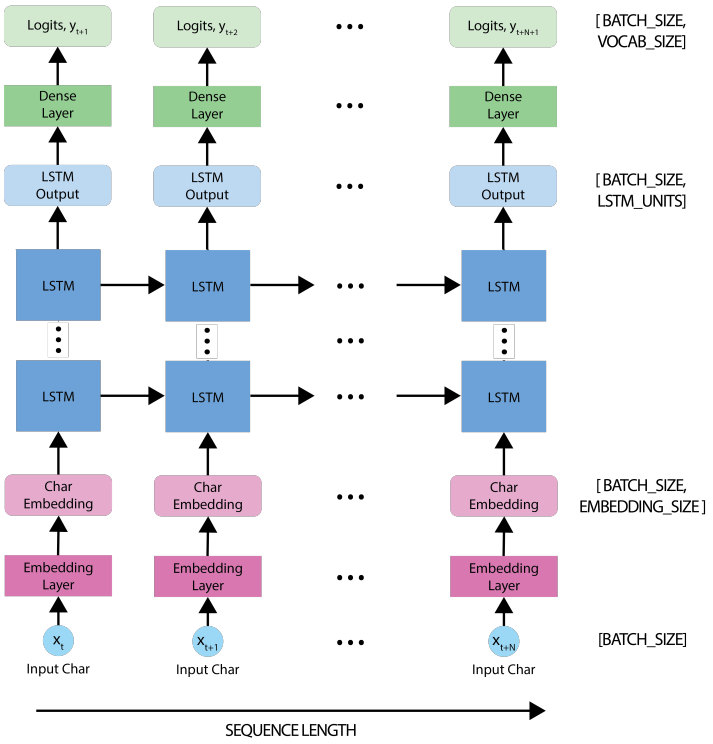


FIGURA 4.3: Diagrama de la RNN tipo LSTM apilada. Diagrama modificado para incluir el apilamiento de capas LSTM. Imagen original cortesía de TensorFlow usado con los permisos establecidos por Creative Commons Attribution 4.0 License.

256, $vocabulary\_size = 12$, $batch\_size = 128$ y $seq\_len = 256$. Esto produce un total de 86,035,468 de parámetros a entrenar.

De acuerdo al diseño del modelo predictivo, la predicción de cantidad de robos por unidad de tiempo es equivalente a una tarea de clasificación, ya que dado el estado previo en la RNN y el caracter en la secuencia de entrada, se busca predecir la clase o etiqueta del siguiente caracter.

Para entrenar el modelo en esta tarea de clasificación, se utiliza una función de pérdida llamada tf.keras.losses.sparse_categorical_crossentropy. En este caso, esta función de pérdida es apropiada, ya que se aplica en la última dimensión correspondiente a las predicciones, el resultado es calcular la pérdida entre los valores reales de los caracteres en las secuencias de entrada, con los caracteres en la secuencia de salida, el valor calculado por la función de error, indica la precisión de predecir correctamente la secuencia de caracteres de entrenamiento. La Figura 4.4 muestra el entrenamiento de la RNN.

En la Figura 4.4 se observa el valor calculado por la función de error de entropía cruzada y su evolución temporal con respecto al número de épocas de entrenamiento. Se observa un comportamiento asintótico para una pérdida, $Loss = 0{,}0$. Ésta es una imagen representativa del entrenamiento de la RNN para una cierta fecha, siendo las gráficas del

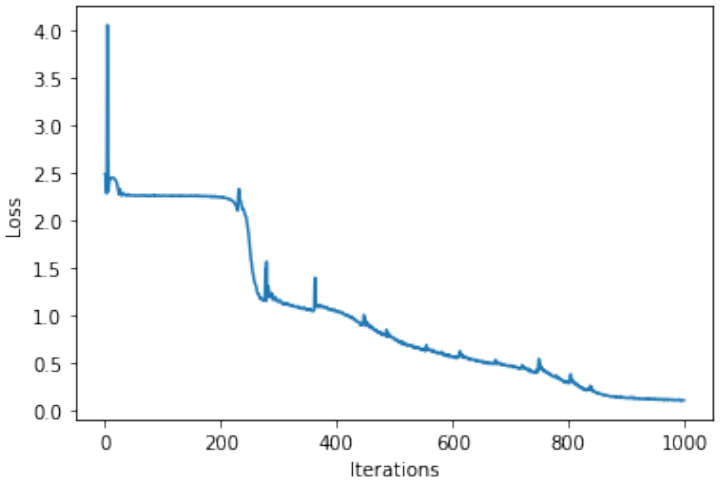


FIGURA 4.4: Progreso del entrenamiento de la RNN. Función de pérdida con respecto a la época de entrenamiento.

resto de las experimentaciones en el entrenamiento de la RNN similares en forma, a la observada en la Figura 4.4.

4.3. Modelo convLSTM

En esta sección se describe la metodología empleada para la predición de la distribución espacio-temporal del crimen. A diferencia del modelo LSTM, el modelo convLSTM permite identificar correlaciones espaciales entre varios crímenes.

Específicamente, se muestra la arquitectura propuesta para el sistema de predicción, compuesta por: las fuentes de datos, el módulo ETL, los tensores de datos, el módulo del Modelo predictivo, y finalmente, la predicción. La Figura 4.5 muestra el diseño propuesto de la arquitectura a utilizar para el sistema de predicción de crímenes propuesto.

Esta arquitectura está compuesta por dos grandes módulos: ETL y Modelo predictivo. A continuación se describe en detalle cada módulo.

Módulo ETL

La base de datos contiene veintidós atributos [26], a partir de los cuales se seleccionaron tres, a ser, zona (Beat), tipo de crimen (Primary Type) y fecha (Date). Observamos que la unidad de tiempo más pequeña es de "1 dia" los primeros años. Posteriormente, la unidad de tiempo mínima cambia a "1 min". Descartamos datos criminales previos al año 2005.

Proceso Segmentación por tipo de crimen. Este proceso permite seleccionar varios tipos de crimen, sin embargo, los tipos de crímenes para los cuales existen menos de

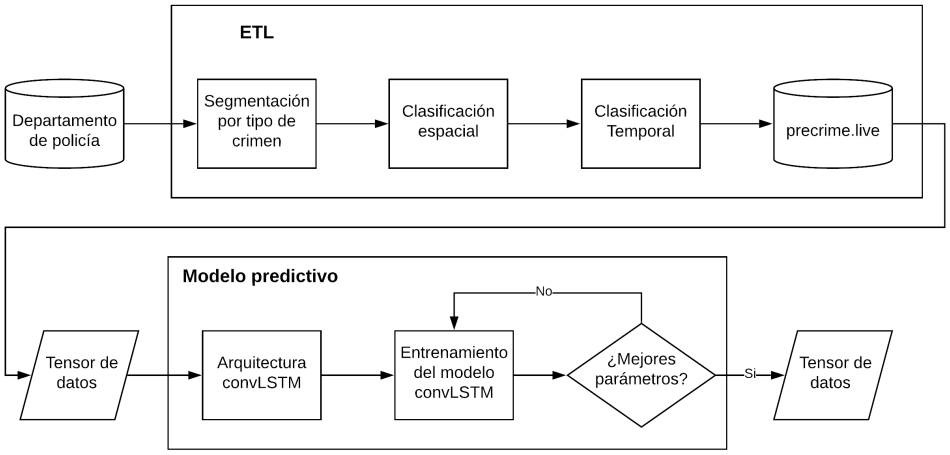


FIGURA 4.5: Arquitectura propuesta para el sistema de predicción de crímenes.

100,000 crímenes históricos fueron descartados para optimizar la ejecución durante el entrenamiento y no exceder el límite de memoria disponible. De acuerdo al criterio de selección establecido se obtuvieron 11 tipos de crimen.

Proceso Clasificación espacial. La ciudad de Chicago se divide geográficamente en áreas denotadas por polígonos irregulares llamados "comunidades". Cada crimen ocurre dentro de una de estas áreas geográficas. Por lo tanto, las coordenadas de longitud y latitud asociadas a un crimen son mapeadas a la comunidad correspondiente. Adicionalmente, es posible subdividir una comunidad en subáreas aumentando así la resolución espacial de las predicciones.

Proceso Clasificación temporal. La base de datos criminal cuenta con una unidad de tiempo de "1 min". Una unidad de tiempo muy pequeña implica reducir la cantidad de datos que pueden ser asignados en memoria. Por lo tanto, se determinó que una unidad de tiempo de "1 dia" es suficiente para evaluar la precisión del algoritmo en una primera aproximación a la solución del problema de la predicción del crimen.

Finalmente, se almacenan los datos que cumplen con las características mencionadas en los procesos de clasificación espacial, temporal y de segmentación por tipo de crimen en una nueva base de datos. A partir de esta nueva base de datos se construye un tensor de datos de entrada para el modelo convLSTM.

Módulo Modelo predictivo

Arquitectura convLSTM. Consiste en una secuencia de capas convLSTM. Los datos de entrada y salida son tensores de rango 3, formados por una secuencia temporal de matrices cuyos renglones denotan la zona y sus columnas denotan el tipo de crimen. El valor de cada elemento de estas matrices espaciales, corresponde a la cantidad de

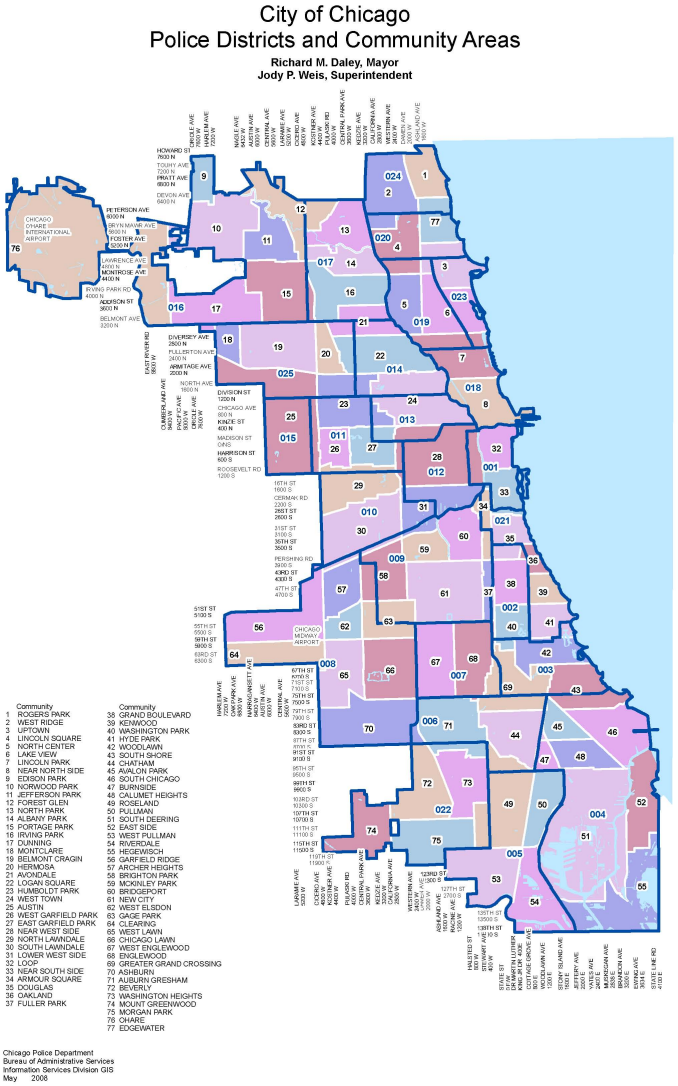


FIGURA 4.6: Clasificación espacial de la ciudad de Chicago en distritos y Comunidades.

crímenes que ocurrieron en las zonas para todos los tipos de crimen por unidad de tiempo.

Durante la programación del modelo convLSTM, se utilizó la API de Keras, específicamente, tf.keras.Sequential. Las cuatro capas principales son:

tf.keras.layers.conv3d: Esta capa crea un núcleo de convolución que se convoluciona con la entrada de la capa para producir un tensor de salidas.

tf.keras.layers.ConvLSTM2D: Es similar a una capa LSTM, pero las transformaciones de entrada y las transformaciones recurrentes son convolucionales.

tf.keras.layers.Dense: Es la capa densamente conectada usual de una red neuronal regular.

tf.keras.layers.BatchNormalization(): La normalización por lotes aplica una transformación que mantiene la media de la salida cercana a 0 y la desviación estándar cercana a 1.

```
Model: "sequential"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
batch_normalization (BatchNo (None, 5, 303, 11, 1) 4
_________________________________________________________________
conv_lst_m2d (ConvLSTM2D) (None, 5, 303, 11, 164) 974816
_________________________________________________________________
batch_normalization_1 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_2 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_1 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_2 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_3 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_2 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_3 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_4 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_3 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_4 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_5 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_4 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_5 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_6 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_5 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_6 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
```

```
_________________________________________________________________
batch_normalization_7 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_6 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_7 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_8 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_7 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_8 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_9 (Batch (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_8 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
conv_lst_m2d_9 (ConvLSTM2D) (None, 5, 303, 11, 164) 1937168
_________________________________________________________________
batch_normalization_10 (Batc (None, 5, 303, 11, 164) 656
_________________________________________________________________
re_lu_9 (ReLU) (None, 5, 303, 11, 164) 0
_________________________________________________________________
dense (Dense) (None, 5, 303, 11, 1) 165
_________________________________________________________________
batch_normalization_11 (Batc (None, 5, 303, 11, 1) 4
_________________________________________________________________
re_lu_10 (ReLU) (None, 5, 303, 11, 1) 0
=================================================================
Total params: 18,416,061
Trainable params: 18,412,777
Non-trainable params: 3,284
_________________________________________________________________
```

Proceso Entrenamiento del modelo convLSTM.

El entrenamiento consiste en darle al modelo la capacidad para responder a la pregunta ¿cuántos crímenes de cierto tipo por unidad de tiempo se esperan en esta zona los siguientes días?

De acuerdo al diseño del modelo predictivo, la predicción de cantidad de robos por unidad de tiempo es equivalente a una tarea de regresión. Para entrenar el modelo de regresión, se utiliza una función de pérdida llamada tf.keras.losses.mse. La Figura 4.7 muestra el entrenamiento del modelo convLSTM.

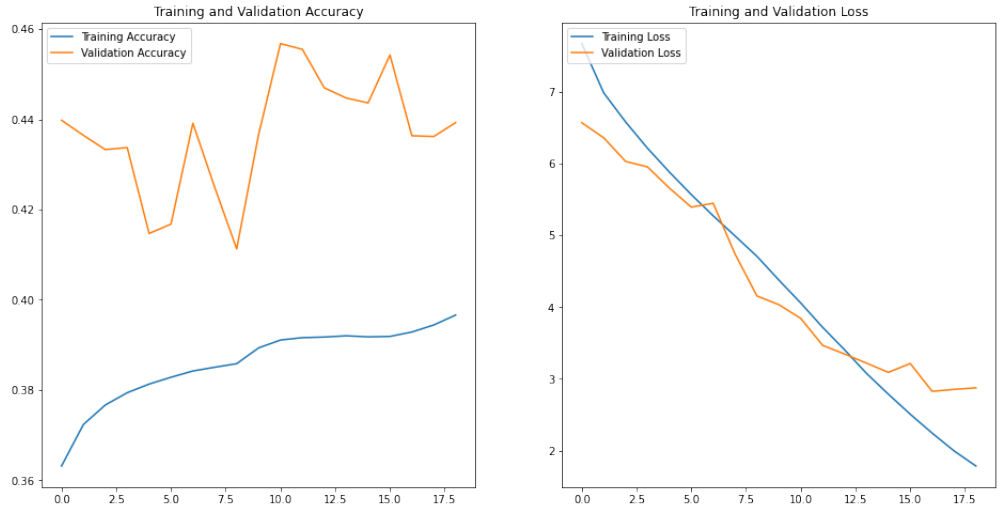


FIGURA 4.7: Progreso del entrenamiento del modelo convLSTM. Función de pérdida y precisión, con respecto a la época de entrenamiento.

A partir de la Figura 4.7 se observa que a lo largo del proceso de entrenamiento el modelo no presenta sobreajuste, ya que las curvas de error para el conjunto de entrenamiento y validación presentan un comportamiento similar. Así mismo se observa que la precisión se mantiene relativamente constante para el conjunto de validación a lo largo del entrenamiento. Lo cual implica que la arquitectura propuesta tiene un límite superior en su precisión al predecir crímenes. En el conjunto de prueba tenemos una pérdida promedio de 2.8299 y una precisión de 0.4224.

4.4. GUI: Aplicación web

En esta sección se calcularon las probabilidades de ocurrencia en cierta área geográfica y tiempo para distintos tipos de crímenes obtenidos de la base de datos criminal de la Ciudad de Chicago. Luego estas probabilidades se emplearon para generar una distribución de crímenes que fueron mapeados a la ciudad de Hermosillo.

El usuario interactúa con las predicciones generadas mediante una aplicación web. El diseño gráfico de la aplicación web fue elaborado empleando la librería “Dash”. Dash es un marco productivo de Python para crear aplicaciones de análisis web. Escrito sobre Flask, Plotly.js y React.js, Dash es ideal para crear aplicaciones de visualización de datos con interfaces de usuario altamente personalizadas en Python puro. Es especialmente adecuado para cualquiera que trabaje con datos en Python. A través de un par de patrones simples, Dash abstrae todas las tecnologías y protocolos necesarios para construir una aplicación interactiva basada en la web. Las aplicaciones de Dash se representan en el navegador web. Puede implementar sus aplicaciones en servidores y luego compartirlas

a través de URL. Dado que las aplicaciones de Dash se ven en el navegador web, Dash es intrínsecamente multiplataforma y está listo para usarse en dispositivos móviles [28].

El usuario ingresa a la url precrime.live, dada la naturaleza "open source" de los datos que empleamos durante el presente trabajo de tesis, no es necesario crear una cuenta o iniciar sesión. Una vez que el navegador termine de cargar el sitio se observa una página de inicio como la observada en la Figura 4.8.

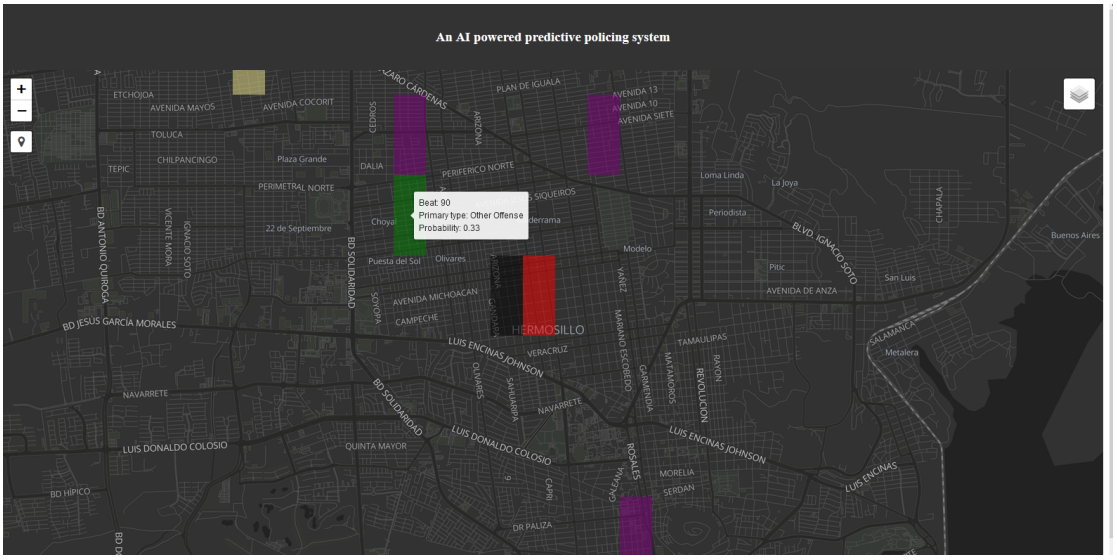


FIGURA 4.8: Vista general de la aplicación web para el sistema precrime.live

La interacción del usuario con la aplicación web es mediante el cursor, en la esquina superior derecha se cuenta con un menú desplegable que le permite al usuario seleccionar entre dos tipos de vistas diferentes, a ser modo oscuro y modo claro, el modo claro se presenta en la Figura 4.9, mientras que el modo oscuro se observa en la Figura 4.8. Así mismo, al colocar el cursor sobre alguna de las zonas, representadas mediante rectángulos de color, se despliega la información correpondiente al identificador de la zona, la probabilidad de ocurrencia del crimen y el tipo de crimen.

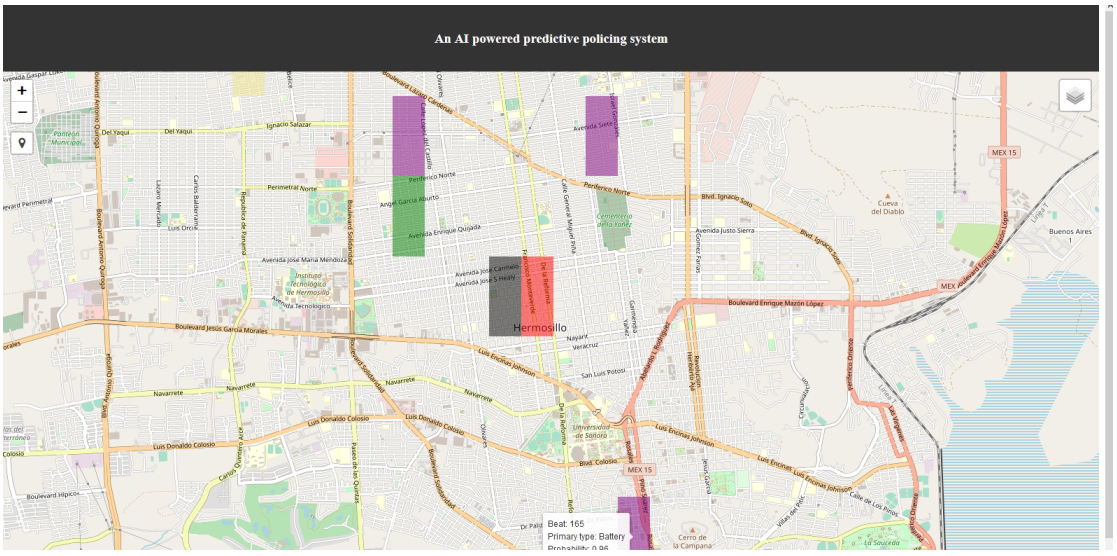


FIGURA 4.9: Vista general de la aplicación web para el sistema precrime.live en modo claro.

Así mismo, en este menú desplegable se pueden seleccionar diferentes tipos de crímenes a visualizar sobre el mapa, marcando o desmarcando las casillas de verificación se activan o desactivan los crímenes a visualizar. En la Figura 4.10 se observa este menú desplegable y los tipos de crímenes entre los cuales el usuario puede escoger visualizar.

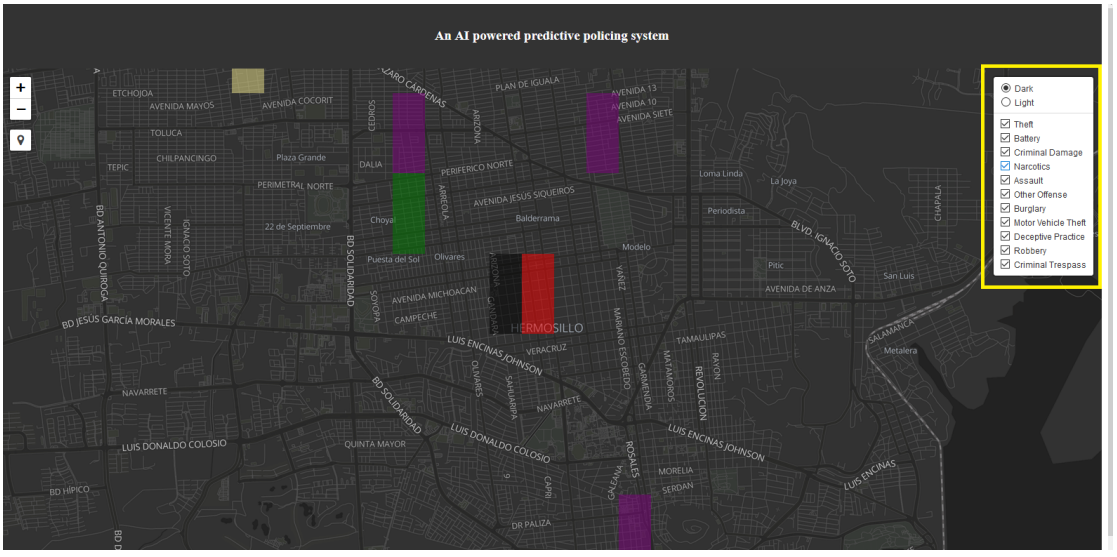


FIGURA 4.10: Vista general del menú para la selección del tipo de crimen a visualizar en el sistema precrime.live

El usuario puede presionar el botón izquierdo del ratón sobre alguna zona coloreada para generar estadísticas tales como el tipo de crimen, el total de crímenes del mismo tipo que han ocurrido en la zona, la probabilidad de ocurrencia del crimen, entre otros. En la Figura 4.11 se observa un ejemplo obtenido al hacer clic en una zona, esta información se muestra en la esquina superior izquierda.

FIGURA 4.11: Se muestra información de la zona coloreada correspondiente al hacer click sobre alguna de las zonas para la cual se desee obtener información detallada.

El usuario puede acercar o alejar el mapa para obtener el detalle acerca de nombres de calles, edificios de referencia, etc. Tal y como se espera al usar un mapa. En la Figura 4.12 se muestra un ejemplo de un acercamiento a una región en particular dentro del mapa.

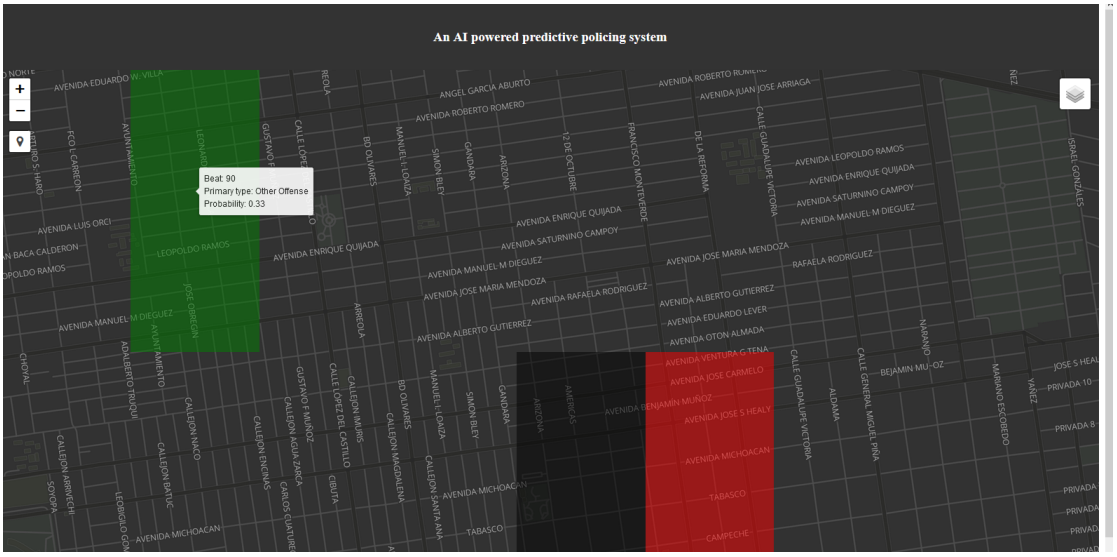


FIGURA 4.12: Efecto que resulta al presionar los botones de zoom en el mapa.

Capítulo 5

Análisis de resultados

Este capítulo incluye los resultados obtenidos mediante la aplicación de los modelos convLSTM y LSTM. Los resultados se presentan en el orden seguido durante la investigación y desarrollo del trabajo de tesis. Comenzando por el modelo LSTM, el cual fue sustituido por el modelo convLSTM, debido a que este último es capaz de identificar correlaciones espaciales así como temporales, adicionalmente permite procesar todos los tipos de crimen simultáneamente en una estructura de datos adecuada.

5.1. Modelo LSTM

Un valor muy pequeño en la función de pérdida puede indicar un sobreajuste de los datos; sin embargo, debido al diseño en la secuencia de entrenamiento para la RNN, al ser una secuencia de varias ventanas de tiempo altamente correlacionadas, un posible sobreajuste en los datos es probable después de tantas épocas de entrenamiento. Los saltos agudos en la función de error, observados a partir de iteraciones mayores a 1250 se deben a que en ese momento, el modelo recibió una secuencia de caracteres que no habían sido observados previamente, entre los cuales es posible haya habido un valor atípico.

El modelo propuesto en este trabajo de investigación predice el futuro en la actividad criminal, i. e., el número de robos por unidad de tiempo a partir de la última fecha donde se registró un crimen. El número de crímenes deberá ser normalizado y la unidad de tiempo es un día. La validación de los resultados se realiza por medio de un muestreo aleatorio para diferentes fechas, a partir de las cuales se busca predecir el comportamiento criminal. Se llevaron a cabo seis ejecuciones de la RNN y se calcularon dos medidas de desempeño: la correlación y la raíz del error cuadrático medio (RMSE, por sus siglas en inglés).

Los resultados experimentales, en esta muestra de estudio, muestran una correlación promedio de 0.599 y el RMSE obtenido fue de 0.110. El valor de correlación positiva obtenido es significativo, sobretodo tomando en consideración la complejidad del comportamiento humano, en particular el comportamiento criminal futuro de un individuo. El encontrar una correlación positiva con el modelo propuesto, muestra la existencia de un patrón en la mente criminal, que potencialmente puede ser explotado por los departamentos de policía.

En la Figura 5.1 se observa cómo durante el conjunto de fechas tales que Date es mayor que "Last date known to the system", el modelo propuesto en este trabajo de investigación está prediciendo el futuro. Las fechas tales que Date es menor que "Last date known to the system", el modelo busca replicar la cantidad de crímenes observados en los nueve días previos a la fecha "Last date known to the system" que ha registrado el CPD. De acuerdo a la correlación obtenida para la ventana de tiempo previa a la fecha "Last date known to the system", se implementa un algoritmo que selecciona la ventana de tiempo posterior a la fecha "Last date known to the system", esta última ventana de tiempo se encuentra en lo que el modelo percibe como el futuro.

En la gráfica superior izquierda, se muestran dos ventanas de tiempo de nueve días centradas alrededor del 23 de Marzo del 2017. La línea azul representa la cuenta total de robos normalizados y registrados por día por el CPD. La línea roja representa la cuenta total de robos normalizados predecidos por día por el modelo de RNN tipo LSTM apilado, propuesto en este artículo. Este modelo fue entrenado empleando los datos criminales registrados en la Zona 11 por el CPD, en un período de tiempo del 2001-01-01 al 2017-03-23. Una vez entrenado el modelo se hace la predicción de dos ventanas de tiempo, la primera corresponde al período de tiempo previo al 2017-03-23 y la segunda corresponde al período de tiempo posterior al 2017-03-23.

De acuerdo a la correlación obtenida para la ventana de tiempo previa al 23 de Marzo del 2017, se implementa un algoritmo que selecciona la ventana de tiempo posterior al 23 de Marzo del 2017, esta última ventana de tiempo se encuentra en lo que el modelo percibe como el futuro. Por lo tanto, la ventana de tiempo posterior al 23 de Marzo es la secuencia del total de crímenes normalizados por día que se encuentran en el futuro con respecto a la fecha de referencia 23 de Marzo de 2017, fecha en que el CPD había registrado el crimen más reciente.

Cuando ambas líneas se empalman, por ejemplo, en la gráfica superior derecha e inferior izquierda, significa que la RNN fue capaz de predecir con exactitud el número de crímenes registrados por el CPD. De esta manera, estos resultados permitirán coordinar la actividad de patrullaje hacia la zona, en particular para el día donde el sistema ha estimado habrá un pico en la actividad delictiva.

FIGURA 5.1: La línea azul muestra la secuencia de crímenes (CS, por sus siglas en inglés) conforme fue registrada por el Departamento de Policía de Chicago (CPD, por sus siglas en inglés), la línea roja muestra la CS predicha por el algoritmo propuesto en este trabajo de investigación.

5.2. Modelo convLSTM

En la Figura 5.2 se muestra una muestra de los resultados experimentales para la predicción del crimen, en este caso en particular se seleccionaron fechas y zonas aleatoriamente para los diferentes tipos de crimen. La curva azul representa el número de incidencias delictivas históricas, la curva naranja representa el número de incidencias delictivas pronosticadas y la curva verde representa el número de incidencias delictivas registradas por el CPD en el mismo periodo de tiempo.

Los resultados experimentales obtenido empleando el modelo convLSTM son similares a los obtenidos empleando el modelo LSTM, sin embargo una diferencia significativa es la capacidad que tiene el modelo convLSTM para procesar el tensor de datos completo

que incluye todos los crímenes y sus respectiva zonas, mientras que el modelo LSTM procesa la serie de tiempo contenida en una celda del cubo de datos.

Durante la evaluación del modelo en el conjunto de prueba obtuvimos una precisión promedio de 0.42, la precisión medida en este caso corresponde a la diferencia promedio entre el valor exacto y el valor predicho. Sin embargo, como la evidencia experimental sugiere, se obtuvo una correlación cercana a 1 en la mayoría de las evaluaciones experimentales.

FIGURA 5.2: Muestra de resultados experimentales del modelo convLSTM para una zona y fecha aleatoria de los diferentes tipos de crímenes analizados.

Capítulo 6

Conclusiones y Trabajo a futuro

Experimentalmente se demostró que el modelo propuesto, una red neuronal recurrente LSTM apilada, predice el futuro con una correlación promedio de 0.599, para una ventana de tiempo en el futuro de nueve días, a partir de la fecha más reciente para la cual se haya registrado algún crimen. Así mismo, se obtuvo un RMSE promedio de 0.110.

El valor de correlación positiva obtenido es significativo, ya que si se considera la complejidad del comportamiento humano, en particular el comportamiento criminal futuro de un individuo, una correlación positiva de 0.599 es una buena aproximación para la predicción del comportamiento criminal. Es notable que el modelo, en general, identifica correctamente el máximo global en actividad delictiva o cantidad de crímenes por día en el futuro, comprendido dentro de la ventana de tiempo de nueve días generada por la RNN.

Las limitaciones en la tecnología actual son un factor tal, que al estar en continuo desarrollo, permiten visualizar un futuro más seguro para México y el mundo. El diseño de la red neuronal presentado es de carácter general, por lo tanto, su aplicación a series de tiempo de cualquier tipo es posible, siempre y cuando el objetivo sea predecir una ventana de tiempo, a partir de algún tiempo de referencia que distinga el pasado del futuro.

Una limitante importante en el modelo LSTM es que no puede identificar correlaciones espaciales. Debido a que fue diseñado para el análisis de secuencias unidimensionales o bien, series de tiempo. Para resolver este problema se propuso el uso de un nuevo modelo de inteligencia artificial. El modelo convLSTM es el caso general del modelo LSTM tal que no solo es eficaz durante la identificación de correlación temporal, sino también en la identificación de correlaciones espaciales entre los diversos tipos de crimen. Por lo tanto, en mi opinión el modelo convLSTM es el mejor modelo propuesto hasta

el dia de hoy para la identificación, predicción y clasificación de patrones en secuencias espacio-temporales.

Como trabajo futuro se propone la implementación del modelo convLSTM en un mejor hardware, lo cual permitirá aumentar la complejidad del modelo y la cantidad de parámetros entrenables, con el objetivo de mejorar la precisión en las predicciones realizadas. Así mismo, se puede considerar la creación de un sistema que emplee varios modelos de inteligencia artificial los cuales trabajan independientemente pero simultáneamente, obteniendo predicciones de cada uno de ellos que luego son combinadas en una predicción final mediante otro algoritmo de inteligencia artificial.

Apéndice A

Tensores y sus operaciones

A.1. Operaciones entre tensores en Python

Suma de tensores

La suma entre dos tensores cualesquiera está definida si y sólo si ambos tienen la misma forma. El resultado que se obtiene al sumar dos tensores es un nuevo tensor con la misma forma que los sumandos. El procedimiento para calcular la suma es análogo al procedimiento empleado de acuerdo a las reglas establecidas en álgebra lineal, donde cada valor escalar es la suma de elementos de los escalares en los tensores principales. Por ejemplo: considere dos tensores A, B de rango tres, cada uno de los cuales se forma por dos matrices de tamaño 3×2.

$$A = \begin{matrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{131} & a_{132} \end{matrix}$$

$$\begin{matrix} a_{211} & a_{212} \\ a_{221} & a_{222} \\ a_{231} & a_{232} \end{matrix}$$

$$B = \begin{matrix} b_{111} & b_{112} \\ b_{121} & b_{122} \\ b_{131} & b_{132} \end{matrix}$$

$$\begin{matrix} b_{211} & b_{212} \\ b_{221} & b_{222} \\ b_{231} & b_{232} \end{matrix}$$

Sea $C = A + B$, entonces

$$C = \begin{matrix} a_{111} + b_{111} & a_{112} + b_{112} \\ a_{121} + b_{121} & a_{122} + b_{122} \\ a_{131} + b_{131} & a_{132} + b_{132} \end{matrix}$$

$$\begin{matrix} a_{211} + b_{211} & a_{212} + b_{212} \\ a_{221} + b_{221} & a_{222} + b_{222} \\ a_{231} + b_{231} & a_{232} + b_{232} \end{matrix}$$

En NumPy, podemos agregar tensores directamente de la siguiente manera.

```
# tensor addition
C = A + B
```

Resta de tensores

La resta entre dos tensores se realiza restando los valores escalares de los elementos correspondientes de cada tensor. Ambos tensores a restar deben tener la misma forma. El tensor que resulta de la resta entre dos tensores mantiene constante la forma de los tensores originales.

Sean A, B los tensores definidos previamente, entonces la resta $C = A - B$ queda como sigue.

$$C = \begin{matrix} a_{111} - b_{111} & a_{112} - b_{112} \\ a_{121} - b_{121} & a_{122} - b_{122} \\ a_{131} - b_{131} & a_{132} - b_{132} \end{matrix}$$

$$\begin{matrix} a_{211} - b_{211} & a_{212} - b_{212} \\ a_{221} - b_{221} & a_{222} - b_{222} \\ a_{231} - b_{231} & a_{232} - b_{232} \end{matrix}$$

En NumPy, podemos restar tensores directamente de la siguiente manera.

```
# tensor subtraction
C = A - B
```

Producto Hadamard

El producto Hadamard entre dos tensores, consiste en el producto de los valores escalares de los elementos correspondientes entre los tensores a multiplicar. Es posible calcular el producto Hadamard si y sólo si ambos tensores tienen la misma forma.

Sean A, B los tensores definidos previamente, entonces el producto Hadamard $C = A \circ B$ queda como sigue.

$$C = \begin{matrix} a_{111} \cdot b_{111} & a_{112} \cdot b_{112} \\ a_{121} \cdot b_{121} & a_{122} \cdot b_{122} \\ a_{131} \cdot b_{131} & a_{132} \cdot b_{132} \end{matrix}$$
$$\begin{matrix} a_{211} \cdot b_{211} & a_{212} \cdot b_{212} \\ a_{221} \cdot b_{221} & a_{222} \cdot b_{222} \\ a_{231} \cdot b_{231} & a_{232} \cdot b_{232} \end{matrix}$$

En NumPy, podemos multiplicar tensores directamente de la siguiente manera.

```
# tensor Hadamard product
C = A * B
```

División de tensores

La división por elementos de un tensor con respecto a otro tensor de las mismas dimensiones da como resultado un nuevo tensor con las mismas dimensiones, donde cada valor escalar es la división por elementos de los escalares en los tensores principales.

Sean A, B los tensores definidos previamente, entonces la división $C = A/B$ queda como sigue.

$$C = \begin{matrix} a_{111}/b_{111} & a_{112}/b_{112} \\ a_{121}/b_{121} & a_{122}/b_{122} \\ a_{131}/b_{131} & a_{132}/b_{132} \\ \\ a_{211}/b_{211} & a_{212}/b_{212} \\ a_{221}/b_{221} & a_{222}/b_{222} \\ a_{231}/b_{231} & a_{232}/b_{232} \end{matrix}$$

En NumPy, podemos dividir tensores directamente de la siguiente manera.

```
# tensor división
C = A / B
```

Producto tensorial

Dado un tensor A de rango q y un tensor B de rango r, el producto de estos tensores será un nuevo tensor de rango $q + r$. Por ejemplo: considere dos tensores A, B de rango 1, cada uno de los cuales corresponde a una matriz de tamaño 1×2.

$$\begin{aligned} A &= [a_{11}a_{12}] \\ B &= [b_{11}b_{12}] \end{aligned}$$

Sea $C = A \otimes B$, entonces

$$C = \begin{matrix} a_{11} \cdot [b_{11} & b_{12}] \\ a_{12} \cdot [b_{11} & b_{12}] \end{matrix}$$

El producto tensorial se puede implementar en NumPy usando la función tensordot(). La función toma como argumentos los dos tensores que se van a multiplicar y el eje en el que se suman los productos, llamado reducción de suma.

```
C = tensordot(A, B, axes=0)
```

Este producto tensorial es la forma más común de multiplicación entre tensores que se puede encontrar, sin embargo existen otros tipos de productos tensoriales.

A.2. Operaciones entre tensores en Tensorflow

Las operaciones matemáticas básicas con tensores, tales como la suma, el producto de Hadamard y el producto tensorial se muestran a continuación.

```
In:
a = tf.constant([[1, 2],
[3, 4]])
b = tf.constant([[1, 1],
[1, 1]]) # Could have also said ‘tf.ones([2,2])‘

print(tf.add(a, b), "\n")
print(tf.multiply(a, b), "\n")
print(tf.matmul(a, b), "\n")

Out:
tf.Tensor(
[[2 3]
[4 5]], shape=(2, 2), dtype=int32)

tf.Tensor(
[[1 2]
[3 4]], shape=(2, 2), dtype=int32)

tf.Tensor(
[[3 3]
[7 7]], shape=(2, 2), dtype=int32)

In:
print(a + b, "\n") # element-wise addition
print(a * b, "\n") # element-wise multiplication
print(a @ b, "\n") # matrix multiplication

Out:
tf.Tensor(
[[2 3]
[4 5]], shape=(2, 2), dtype=int32)

tf.Tensor(
```

```
[[1 2]
[3 4]], shape=(2, 2), dtype=int32)

tf.Tensor(
[[3 3]
[7 7]], shape=(2, 2), dtype=int32)
```

Así mismo, sobre los tensores se pueden aplicar todo tipo de operaciones que uno encuentra en los lenguajes de programación.

```
In:
c = tf.constant([[4.0, 5.0], [10.0, 1.0]])
# Find the largest value
print(tf.reduce_max(c))
# Find the index of the largest value
print(tf.argmax(c))
# Compute the softmax
print(tf.nn.softmax(c))

Out:
tf.Tensor(10.0, shape=(), dtype=float32)
tf.Tensor([1 0], shape=(2,), dtype=int64)
tf.Tensor(
[[2.6894143e-01 7.3105860e-01]
[9.9987662e-01 1.2339458e-04]], shape=(2, 2), dtype=float32)
```

Los nombres comúnmente asociados a las dimensiones de un tensor se muestran en la Figura A.1.

Por ejemplo para un tensor de rango 4 la definición del mismo se programa como sigue.

```
In:
rank_4_tensor = tf.zeros([3, 2, 4, 5])
```

La visualización de este tensor se puede representar como aparece en la Figura A.1.

Así mismo, es posible explorar las propiedades del tensor y sus elementos.

```
In:
print("Type of every element:", rank_4_tensor.dtype)
```

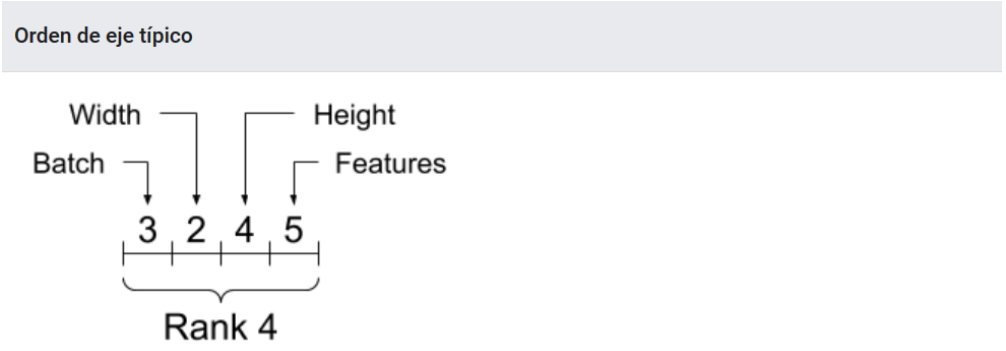


FIGURA A.1: Nombres comúnmente asociados a las dimensiones de un tensor de rango 4.

```
print("Number of dimensions:", rank_4_tensor.ndim)
print("Shape of tensor:", rank_4_tensor.shape)
print("Elements along axis 0 of tensor:", rank_4_tensor.shape[0])
print("Elements along the last axis of tensor:", rank_4_tensor.shape[-1])
print("Total number of elements (3*2*4*5): ", tf.size(rank_4_tensor).numpy())

Out:
Type of every element: <dtype: 'float32'>
Number of dimensions: 4
Shape of tensor: (3, 2, 4, 5)
Elements along axis 0 of tensor: 3
Elements along the last axis of tensor: 5
Total number of elements (3*2*4*5): 120
```

Indexación sobre un solo eje

TensorFlow sigue las reglas de indexación estándares de Python, similares a indexar una lista o una cadena en Python, y las reglas básicas para la indexación de NumPy, tales como: los índices comienzan en 0, los índices negativos cuentan hacia atrás desde el final y los dos puntos, : , se utilizan para los cortes: start:stop:step

```
In:
rank_1_tensor = tf.constant([0, 1, 1, 2, 3, 5, 8, 13, 21, 34])
print(rank_1_tensor.numpy())

Out:
[0 1 1 2 3 5 8 13 21 34]
```

La indexación con un escalar elimina la dimensión:

```
In:
print("First:", rank_1_tensor[0].numpy())
print("Second:", rank_1_tensor[1].numpy())
print("Last:", rank_1_tensor[-1].numpy())

Out:
First: 0
Second: 1
Last: 34
```

La indexación de segmentos, empleando ":", mantiene la dimensión:

```
In:
print("Everything:", rank_1_tensor[:].numpy())
print("Before 4:", rank_1_tensor[:4].numpy())
print("From 4 to the end:", rank_1_tensor[4:].numpy())
print("From 2, before 7:", rank_1_tensor[2:7].numpy())
print("Every other item:", rank_1_tensor[::2].numpy())
print("Reversed:", rank_1_tensor[::-1].numpy())

Out:
Everything: [0 1 1 2 3 5 8 13 21 34]
Before 4: [0 1 1 2]
From 4 to the end: [3 5 8 13 21 34]
From 2, before 7: [1 2 3 5 8]
Every other item: [0 1 3 8 21]
Reversed: [34 21 13 8 5 3 2 1 1 0]
```

Indexación multieje

Los tensores de rango superior se indexan pasando como argumento varios índices. Las reglas de indexación para un solo eje se aplican a cada eje de forma independiente.

```
In:
# Get row and column tensors
print("Second row:", rank_2_tensor[1, :].numpy())
print("Second column:", rank_2_tensor[:, 1].numpy())
print("Last row:", rank_2_tensor[-1, :].numpy())
print("First item in last column:", rank_2_tensor[0, -1].numpy())
```

```
print("Skip the first row:")
print(rank_2_tensor[1:, :].numpy(), "\n")

Out:
Second row: [3. 4.]
Second column: [2. 4. 6.]
Last row: [5. 6.]
First item in last column: 2.0
Skip the first row:
[[3. 4.]
[5. 6.]]
```

En la Figura A.2 se muestran dos ejemplos para la selección de segmentos o conjuntos de elementos.

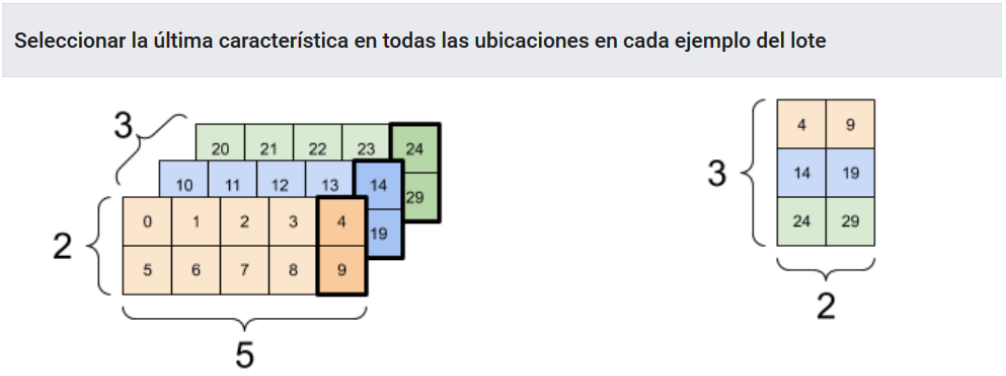


FIGURA A.2: Visualización para la selección de elementos en un tensor.

Manipular formas (reshape)

```
In:
# Shape returns a ‘TensorShape‘ object that shows the size on each dimension
x = tf.constant([[1], [2], [3]])
print(x.shape)

Out:
(3, 1)

In:
# You can convert this object into a Python list, too
print(x.shape.as_list())
```

```
Out:
[3, 1]
```

Normalmente, los únicos usos razonables de tf.reshape son combinar o dividir ejes adyacentes. Por ejemplo: definamos un tensor de 3x2x5 para el cual se desea cambiar la forma a (3x2) x5 o 3x (2x5). Estos cambios en la forma son razonables, ya que los cortes no se mezclan:

```
In:
print(tf.reshape(rank_3_tensor, [3*2, 5]), "\n")
print(tf.reshape(rank_3_tensor, [3, -1]))

tf.Tensor(
[[0 1 2 3 4]
[5 6 7 8 9]
[10 11 12 13 14]
[15 16 17 18 19]
[20 21 22 23 24]
[25 26 27 28 29]], shape=(6, 5), dtype=int32)

tf.Tensor(
[[0 1 2 3 4 5 6 7 8 9]
[10 11 12 13 14 15 16 17 18 19]
[20 21 22 23 24 25 26 27 28 29]], shape=(3, 10), dtype=int32)
```

En la figura A.3 se muestran los cambios en la forma realizados con respecto al tensor definido previamente.

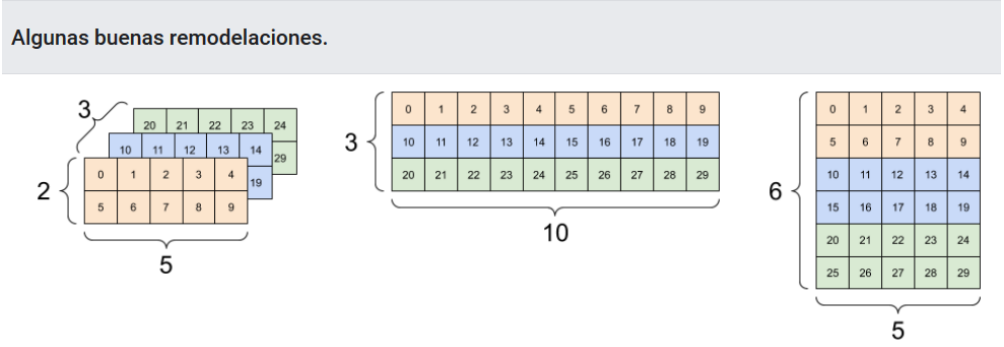


FIGURA A.3: Visualización de transformaciones de forma válidas.

La remodelación funcionará para cualquier forma nueva con el mismo número total de elementos, pero no servirá de nada si no respeta el orden de los ejes, en este caso para el intercambio de ejes es necesario emplear tf.transpose.

En la Figura A.4 se muestran modificaciones incorrectas.

```
In:
# Bad examples: don't do this
# You can't reorder axes with reshape.
print(tf.reshape(rank_3_tensor, [2, 3, 5]), "\n")
# This is a mess
print(tf.reshape(rank_3_tensor, [5, 6]), "\n")
# This doesn't work at all
try:
tf.reshape(rank_3_tensor, [7, -1])
except Exception as e:
print(f"{type(e).__name__}: {e}")
```

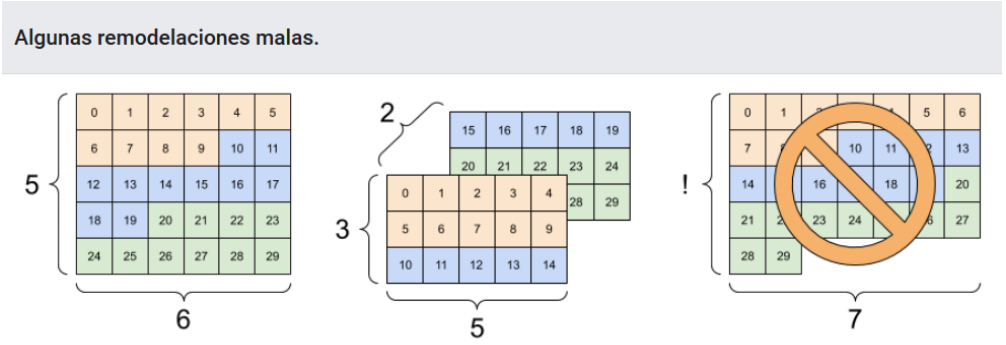


FIGURA A.4: Visualización de transformaciones inválidas.

Bibliografía

[1] I. for Economics and Peace, "Global peace index 2020: Measuring peace in a complex world," http://visionofhumanity.org/reports, 2019, accessed: Jun. 1, 2020.

[2] S. Roel, "Semáforo delictivo," https://www.semaforo.mx, 2020, accessed: Jul. 1, 2020.

[3] G. Mohler, A. L. Bertozzi, J. Carter, M. B. Short, D. Sledge, G. E. Tita, C. D. Uchida, and P. J. Brantingham, "Impact of social distancing during covid-19 pandemic on crime in los angeles and indianapolis," *Journal of Criminal Justice*, vol. 68, p. 101692, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0047235220301860

[4] I. Haller, "Is there a correlation between poverty and criminality? analysis of european data," *European Review of Applied Sociology*, vol. 7, no. 9, December 2014.

[5] Z. F. Haiyun Zhao and C. Castillo-Chavez, "The dynamics of poverty and crime," *Journal of Shanghai Normal University*, vol. 43, no. 5, October 2014.

[6] C. Vilalta, "How did things get so bad so quickly? an assessment of the initial conditions of the war against organized crime in mexico," *European Journal on Criminal Policy and Research*, vol. 20, pp. 137–161, September 2013.

[7] P. Wang, R. Mathieu, J. Ke, and H. J. Cai, "Predicting criminal recidivism with support vector machine," in *2010 International Conference on Management and Service Science*, 2010, pp. 1–9.

[8] K. Kianmehr and R. Alhajj, "Crime hot-spots prediction using support vector machine," in *IEEE International Conference on Computer Systems and Applications, 2006.*, 2006, pp. 952–959.

[9] K. Zhu and J. Zhang, "Predicting the potential locations of the next crime based on data mining: A case study," vol. 6, pp. 574–581, 11 2012.

[10] N. Malleson, A. Heppenstall, and L. See, "Crime reduction through simulation: An agent-based model of burglary," *Computers, Environment and Urban*

Systems, vol. 34, no. 3, pp. 236 – 250, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971509000787

[11] B. Chandra and M. Gupta, "A multivariate time series clustering approach for crime trends prediction," 11 2008, pp. 892 – 896.

[12] R. Liao, X. Wang, L. Li, and Z. Qin, "A novel serial crime prediction model based on bayesian learning theory," in *2010 International Conference on Machine Learning and Cybernetics*, vol. 4, 2010, pp. 1757–1762.

[13] X. Shi and et al., "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *arXiv preprint arXiv:1506.04214*, 2015.

[14] K. Lum and W. Isaac, "To predict and serve?" *The Royal Statistical Society*, October 2016. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.1740-9713.2016.00960.x

[15] P. Thongtae and S. Srisuk, "An analysis of data mining applications in crime domain," in *2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, 2008, pp. 122–126.

[16] G. Mohler, M. Short, P. Brantingham, F. Schoenberg, and G. Tita, "Self-exciting point process modeling of crime," *Journal of the American Statistical Association*, vol. 106, pp. 100–108, 03 2011.

[17] C. P. Haberman and J. H. Ratcliffe, "The Predictive Policing Challenges of Near Repeat Armed Street Robberies," *Policing: A Journal of Policy and Practice*, vol. 6, no. 2, pp. 151–166, 05 2012. [Online]. Available: https://doi.org/10.1093/police/pas012

[18] PredPol. (2019) Predictive policing technology. [Online]. Available: https://www.predpol.com/technology/

[19] R. A. espacial de riesgos. (2019) Rtmdx: Software. [Online]. Available: http://www.riskterrainmodeling.com/espantildeol.html

[20] HunchLab. (2019) Hunchlab: Under the hood. [Online]. Available: https://www.hunchlab.com/resources/

[21] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[22] S. de Seguridad Pública. (2019) Análisis estadístico de incidencia delictiva y llamadas 911 por municipio - enero - agosto 2019. [Online]. Available: http://sspsonora.gob.mx/index.php/analisis-estadistico-municipal.html

[23] C. S. Widom, "The cycle of violence," *Science*, July 1989.

[24] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *arXiv preprint arXiv:1808.02342*, 2016. [Online]. Available: https://arxiv.org/abs/1808.02342

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[26] C. P. Department. (Aug. 2020) Crimes - 2001 to present. [Online]. Available: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2

[27] A. M. A. Graves and G. Hinton, "Speech recognition with deep recurrent neural networks," *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. pp. 6645–6649, 2013.

[28] Plotly. (2020) Introduction to dash. [Online]. Available: https://dash.plotly.com/introduction