UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

CC61-Artificial Intelligence

| Ciencias de la computación |

TRABAJO PARCIAL

Docente

ROSALES HUAMANCHUMO, JAVIER ULISES

Integrantes

Ibrahim Imanol Jordi Arquinigo Jacinto

Ian Joaquin Sanchez Alva

Mauricio Eduardo Vera Castellon

Diego Armando Flores Carrizales

Nathaly Eliane Anaya Vadillo

2024-02

CONTENIDO

1. Descripción del Problema

Desarrollar un programa que determine la ruta más óptima desde la ubicación actual de una persona hacia universidades en Lima Metropolitana. El programa debe tener en cuenta factores como la hora del día, el nivel de tráfico en tiempo real y otras condiciones de tránsito para ofrecer la ruta más eficiente en términos de tiempo y distancia.

Objetivo: Facilitar la planificación de los universitarios que quieran desplazarse entre sedes de la UPC, asegurando que lleguen a su destino de la forma más rápida.

Definición del problema:

La congestión del tráfico en Lima Metropolitana dificulta que los universitarios se desplacen entre universidades de manera eficiente, generando retrasos que afectan su asistencia. Este proyecto busca desarrollar un sistema que, utilizando redes neuronales y algoritmos como SMA* con index priority queue, determine la ruta óptima considerando la hora del día y el tráfico en tiempo real.

El sistema emplea la API de MapBox para recopilar datos de tráfico y transporte público, con acceso a medio millón de consultas gracias a un equipo de cinco personas. Los datos se complementarán con datos sintéticos para cubrir áreas menos representadas. El objetivo es ofrecer a los universitarios una herramienta que facilite su movilidad y optimice el tiempo de viaje para que llegue a tiempo a la universidad.

FATE:

Fairness: El proyecto se planteó de tal forma que tenga en cuenta diversas variables, como la ubicación de las universidades y las condiciones de tráfico. De esta manera se evitan desventajas para los grupos de estudiantes que se

encuentren en áreas con menor acceso al transporte público. Además, las recomendaciones eliminan los sesgos al basarse en datos actualizados sobre el tráfico y transporte público, lo cual asegura que todas las rutas sugeridas sean viables y justas para todos los universitarios.

Accountability: En el proyecto, garantizamos la responsabilidad mediante la implementación de auditorías regulares para evaluar la precisión y seguridad de las rutas recomendadas por el algoritmo. Además, documentamos todos los aspectos del desarrollo del sistema: desde la adquisición y preprocesamiento de datos hasta la implementación y ajuste de los algoritmos. Nos hacemos responsables de la correcta adquisición y manejo de los datos utilizados en el proyecto. Esto implica asegurar que los datos sean obtenidos de fuentes legítimas y que su uso cumpla con las normativas legales y éticas vigentes. En caso de que se identifiquen problemas relacionados con la adquisición de datos o su uso, tomaremos las medidas necesarias para corregir cualquier irregularidad y abordaremos cualquier impacto negativo que pueda surgir.

Transparency: La transparencia se logra mediante la explicación clara de cómo el sistema toma decisiones, lo que incluye detalles sobre los datos utilizados, el proceso de entrenamiento del modelo y cómo se calculan las recomendaciones de rutas. Proporcionamos documentación accesible que describe el uso de datos y cualquier posible limitación, permitiendo a los usuarios entender el funcionamiento del sistema y confiar en sus recomendaciones.

Ethics: Se garantiza la equidad para el acceso al transporte público para todos los estudiantes universitarios, sin importar su ubicación. Utilizando datos actualizados para evitar sesgos y asegurando que las rutas sugeridas sean justas y estas reflejan las condiciones reales.

1.1. Fundamento del Problema

La congestión del tráfico en Lima Metropolitana es un problema significativo que afecta a los universitarios al desplazarse entre distintas universidades. La falta de información en tiempo real sobre las mejores rutas puede resultar en retrasos innecesarios, afectando la asistencia a clases y otras actividades. Este proyecto tiene como objetivo desarrollar un sistema de recomendación con redes neuronales artificiales que nos determinará un grafo distinto dado un tiempo y día determinado, luego con algoritmos de rutas como A * y otras técnicas más avanzadas nos determinará la ruta más óptima para ir a

diferentes sedes de la UPC. **De esta manera el estudiante podrá planificar y ver en tiempo real cómo se van cambiando los tiempos de llegada de una sede a otra si se retrasa o si el usuario cambia la hora en la que irá.**

Esperamos que este proyecto ayude a los estudiantes de la UPC a desplazarse con más información con los datos de Lima.

2. Descripción del conjuntos de datos (Dataset)

El sistema utilizará una API (MapBox) que permite realizar hasta 100,000 solicitudes gratuitas en una cuenta. Como somos un equipo de cinco personas, esto nos dará acceso a medio millón de consultas, pero solo usaremos los datos de las principales avenidas para llegar de una sede de la upc a la otra**. Usaremos las avenidas más necesarias o importantes para llegar a las diferentes sedes de la UPC**

Usamos la API de MapBox debido a que las demás soluciones que hemos probado como Google MAPS o Tom Tom, o no nos da de por sí datos sobre congestión de Lima, o porque no tenían cuentas gratuitas para poder usar sus productos.

2.1. Agente

El agente es un sistema de recomendación que se basará en el tiempo, día y calle para poder predecir la congestión de estas avenidas principales. El entorno incluye todos los elementos que influyen en el agente inteligente como tiempo y día, aunque también podría ser beneficioso tener datos más a largo plazo como meses, en este proyecto solo tendremos 1 semana, excepto el día sábado.

2.2. Dataset

La API de Mapbox nos incluye datos sobre el tiempo, distancia, congestión en cada paso entre distancias largas y congestiones en números. En este proyecto usaremos la etiqueta de congestión que nos dará una característica tipo: `severe`, `heavy`, `moderate`, `low` o `unknown`

Resumen:

annotation.congestion	string	El nivel de congestión, descrito como severo, alto, moderado, bajo o desconocido, entre cada entrada en el arreglo de pares de coordenadas en el tramo de la ruta. Para cualquier perfil que no sea mapbox/driving-traffic, se devolverá una lista de desconocidos.
annotation.congestion_numeric	número o null	El nivel de congestión en forma numérica, de 0 a 100. Un valor de 0 indica que no hay congestión, un valor de 100 indica congestión máxima. Las entradas pueden ser nulas si se desconoce la congestión en ese segmento.
annotation.distance	número	La distancia entre cada par de coordenadas, en metros.
annotation.duration	número	La duración entre cada par de coordenadas, en segundos.

2.3 Entrenamiento

La red neuronal será entrenada utilizando datos históricos sobre el tráfico entre los nodos del grafo (ciudad de Lima Metropolitana). Esto permitirá que el sistema ajuste sus predicciones en función de las condiciones específicas, como la hora y el día. Por ejemplo, para alguien que comienza su trayecto a las 5 a.m., la red neuronal debería de poder

predecir si hay o no hay tráfico a esa hora en esa conexión específicamente. De manera general sabemos los humanos que a las 4 o 5 am no hay nada de tráfico debido a que casi nadie se despierta a esas horas del día.

Inputs (Entradas):

Los inputs serían las variables que afectan al tráfico en Lima. Algunas posibles entradas relevantes podrían ser:

1. **Hora del día**: Las horas pico y fuera de pico afectan el tráfico de manera significativa.
2. **Día de la semana**: El tráfico entre semana puede ser diferente al de los fines de semana.
3. **Ubicación central de la conexión:** las coordenadas de las conexiones para representar las ubicaciones centrales de esta (Longitud y Latitud de esa ubicación).
4. **Calle**: Principales calles para llegar de una sede a otra.

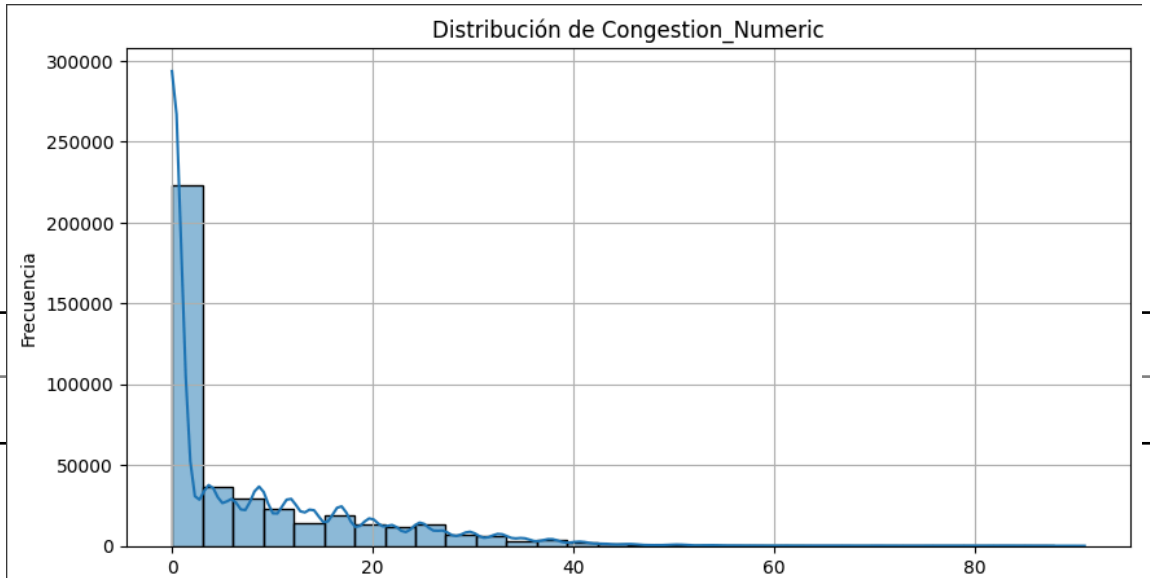
Cada una de estas características las usaremos numéricamente para ser usada como input en la red neuronal.

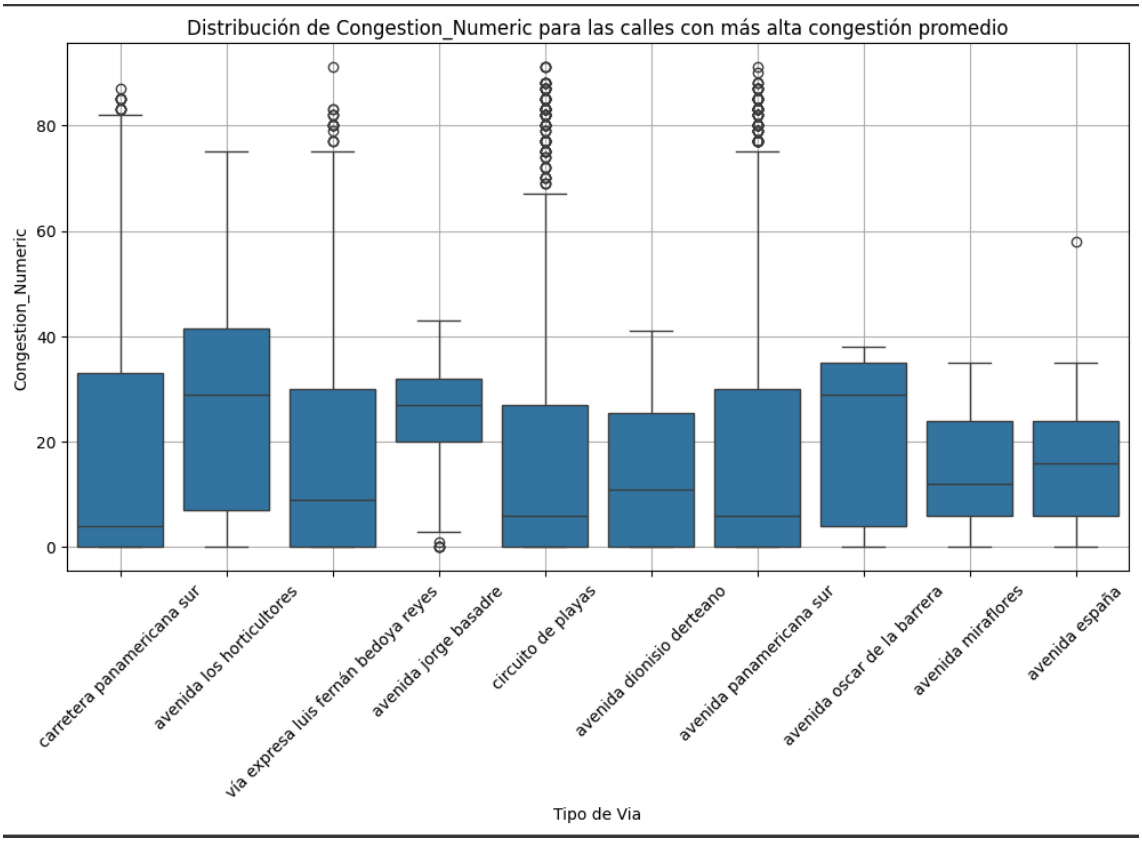
Output (Salida):

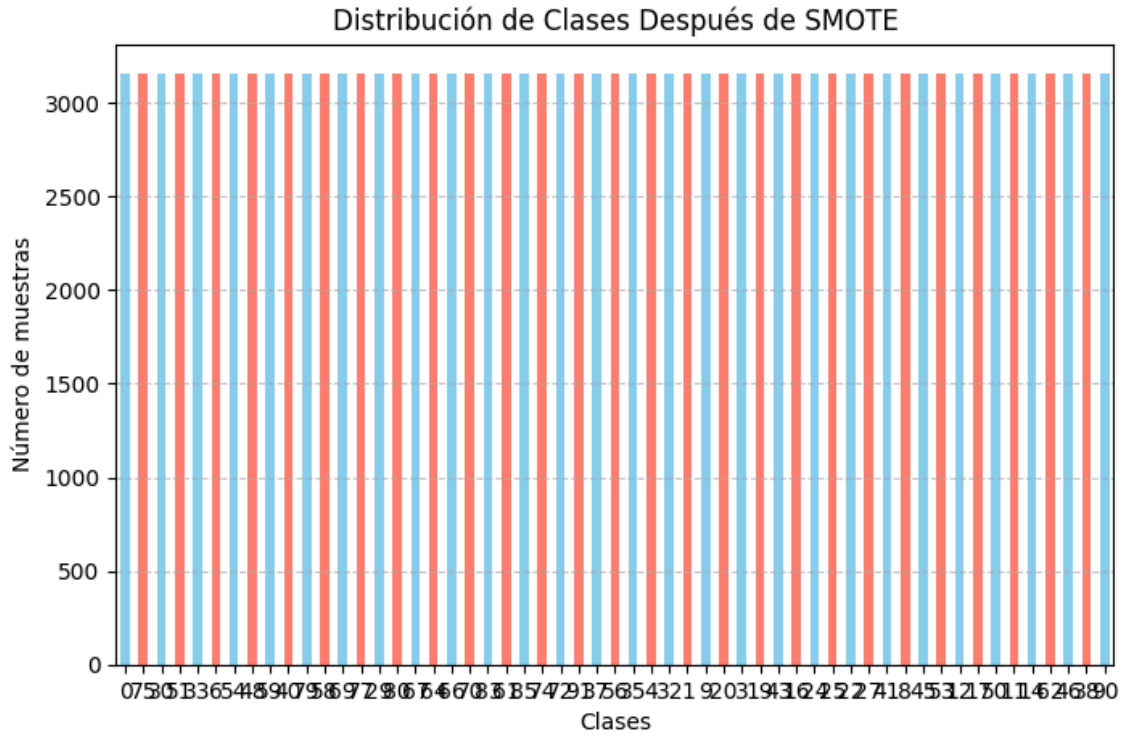
El objetivo principal de la red sería predecir la **congestión de tráfico** entre dos puntos (conexión), tomando en cuenta las condiciones del día de la semana, hora y tipo de calle (conexión).

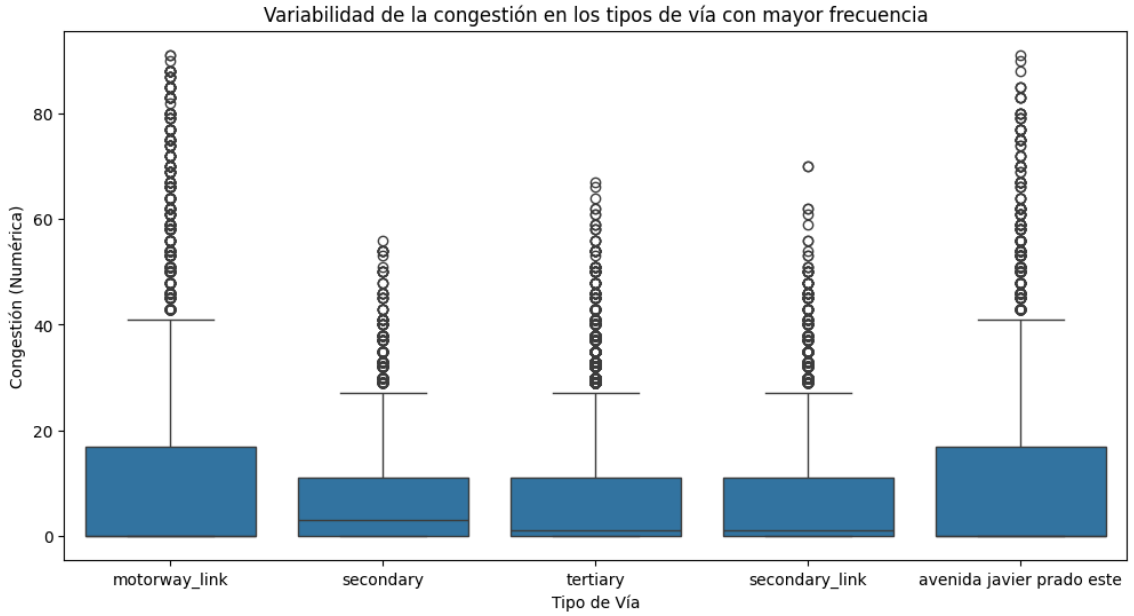
Congestión en Niveles, Congestión Numérica(0-100), Duración de un nodo al otro, Distancia.

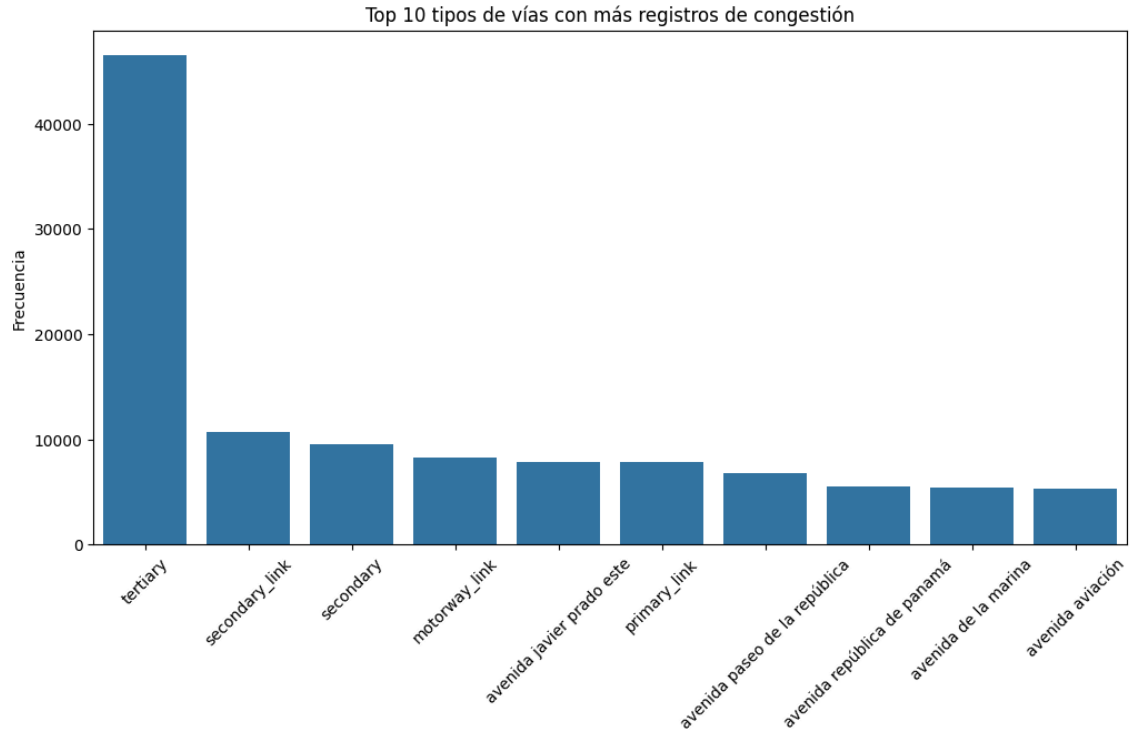
EDA:




Distribución de Congestion_Numeric para las calles con más alta congestión promedio
Congestion_Numeric
0
20
40
60
80
carretera panamericana sur
avenida los horticultores
vía expresa luis fernán bedoya reyes
avenida jorge basadre
circuito de playas
avenida dionisio derteano
avenida panamericana sur
avenida oscar de la barrera
avenida miraflores
avenida españa
Tipo de Via


Distribución de Clases Después de SMOTE
Número de muestras
0
500
1000
1500
2000
2500
3000
Clases

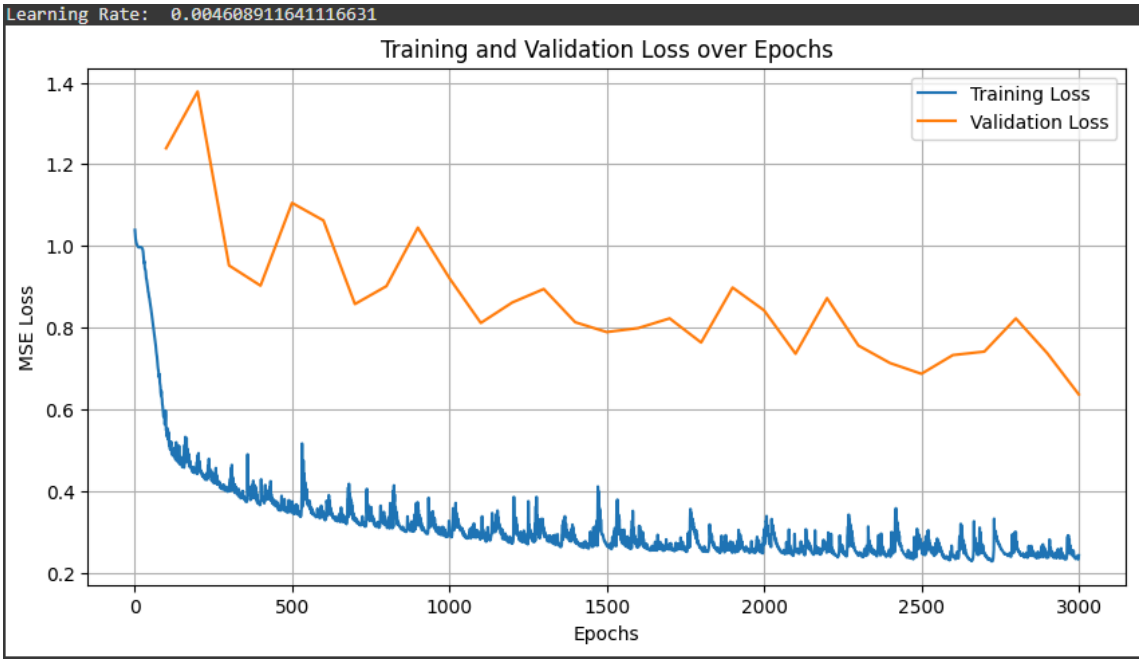

Variabilidad de la congestión en los tipos de vía con mayor frecuencia
Congestión (Numérica)
0
20
40
60
80
motorway_link
secondary
tertiary
secondary_link
avenida javier prado este
Tipo de Vía


Top 10 tipos de vías con más registros de congestión
Frecuencia
0
10000
20000
30000
40000
tertiary
secondary_link
secondary
motorway_link
avenida javier prado este
primary_link
avenida paseo de la república
avenida república de panamá
avenida de la marina
avenida aviación

Hiper Parámetros y entrenamiento:

- **Función de pérdida**: **Mean Squared Error (MSE)**. de la Avenida Javier Prado

- **Función de pérdida**: **Mean Squared Error (MSE)**. para calles principales para llegar de una sede a otra



Basados en el entrenamiento los mejores parámetros fueron para el modelo de predecir sólo la congestión de la Avenida Javier Prado:

LR: 0.00329

capas ocultas: 4

neuronas por capa: 14

El error de validación nos dio: 0.64

Basados en el entrenamiento los mejores parámetros fueron para el modelo grande de predecir las calles principales:

```
calles_principales = [

    'avenida javier prado este', 'avenida javier prado
oeste', 'avenida la marina','avenida antonio josé de
sucre',

'avenida faustino sanchez carrión', 'avenida del
ejército','ovalo josé quiñones',

 'circuito de playas', 'avenida del parque norte', 'avenida
general salaverry', 'avenida la paz', 'avenida san borja
norte', 'avenida lima polo', 'avenida panamericana sur',
'avenida primavera','avenida el derby', 'avenida angamos
este', 'avenida angamos oeste',  'avenida alfredo
benavides'

]
```

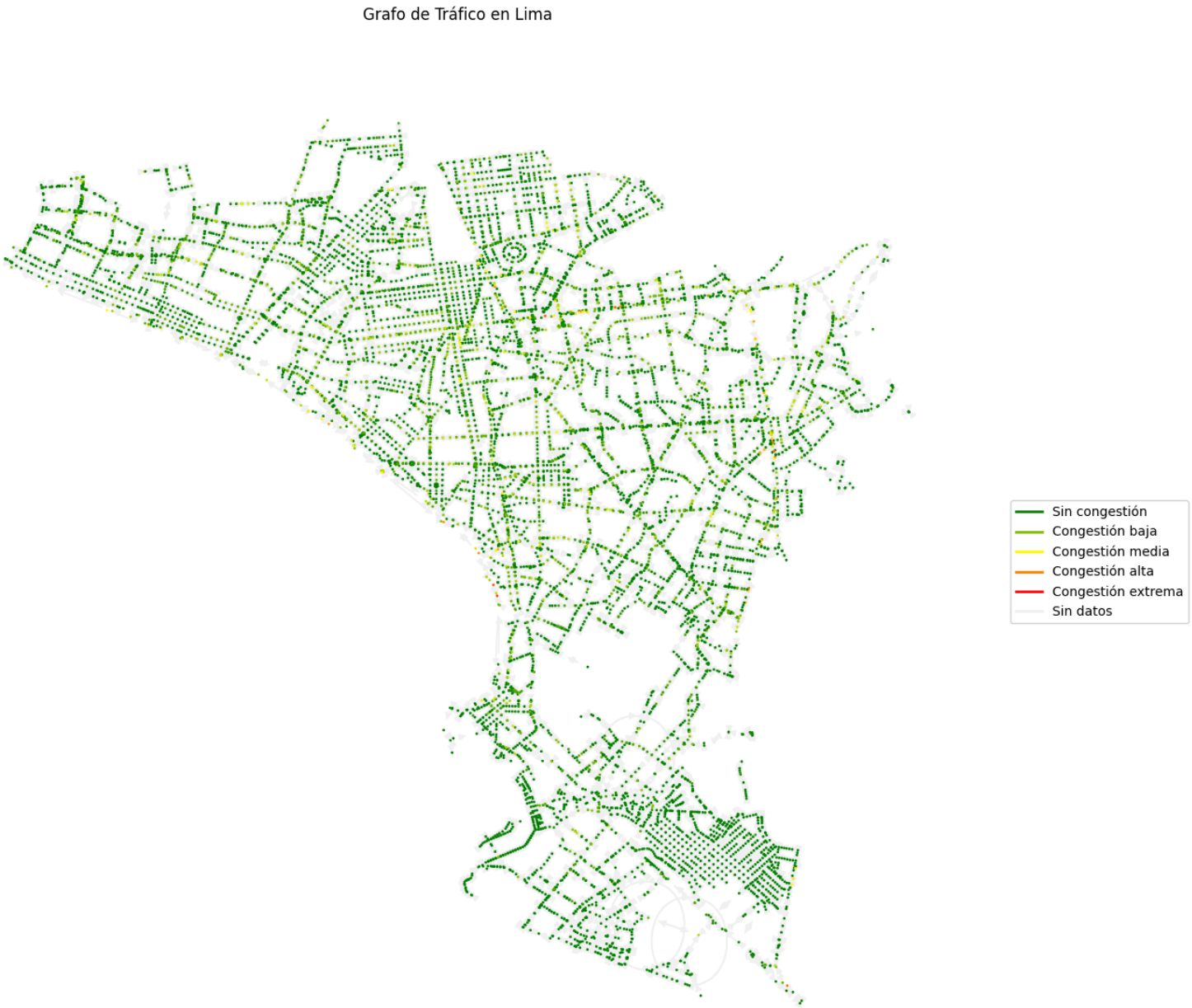
LR: 0.00460

capas ocultas: 14

neuronas por capa: 26

El error de validación nos dio: 0.63628

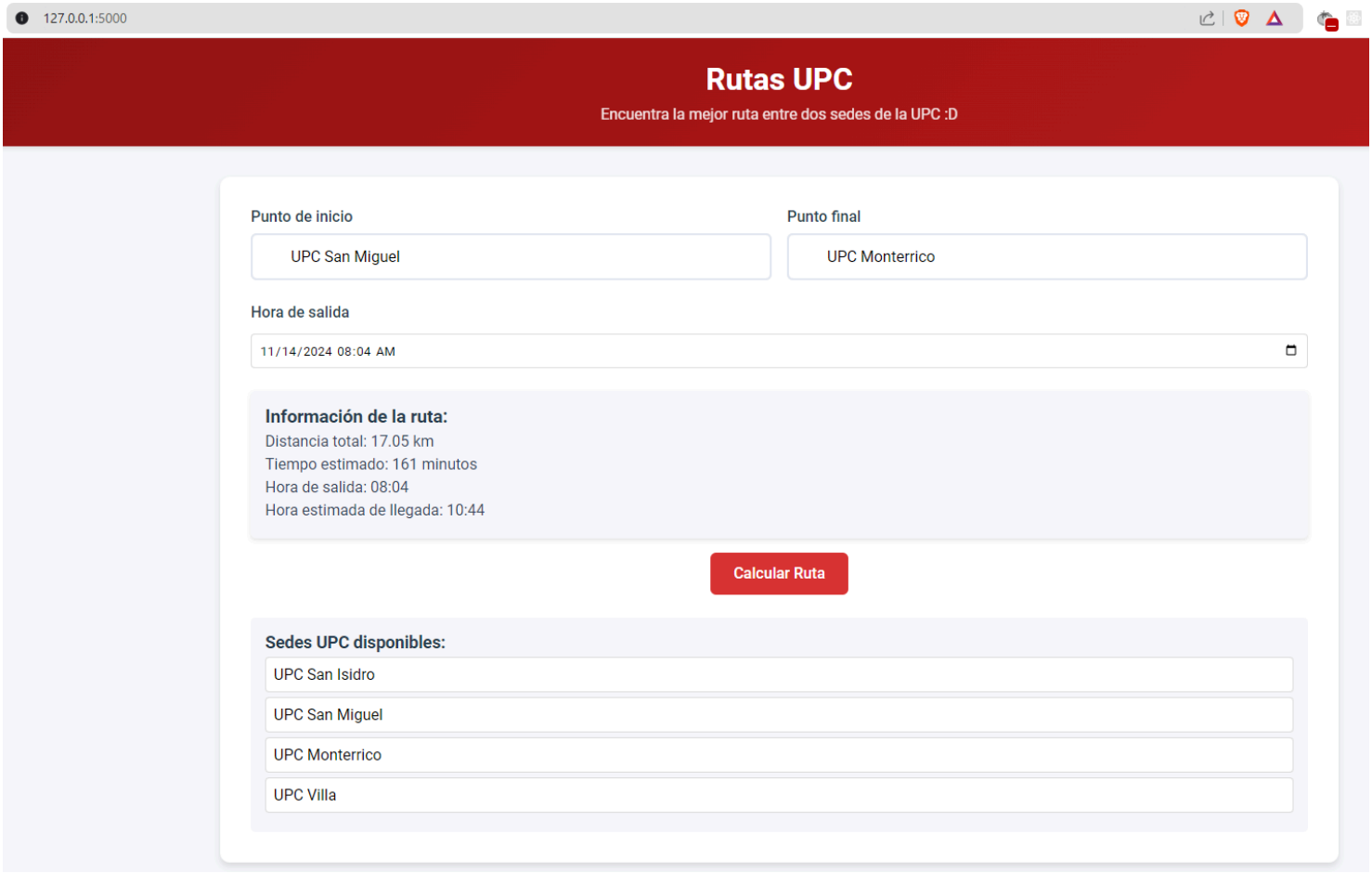
2.4 Visualización de Datos



Aunque hemos recolectado datos de Lima Moderna, solo usaremos las principales avenidas para poder llegar de una sede a otra sede de la UPC, debido al costo computacional de entrenar un modelo de este estilo. Hemos visto lo complicado de entrenar un modelo que incluya toda lima moderna. Ya que habrían muchas características y con ello muchas más neuronas, lo que incrementa el costo computacional de entrenar a un modelo con más de 200 características como calles y diferentes avenidas.

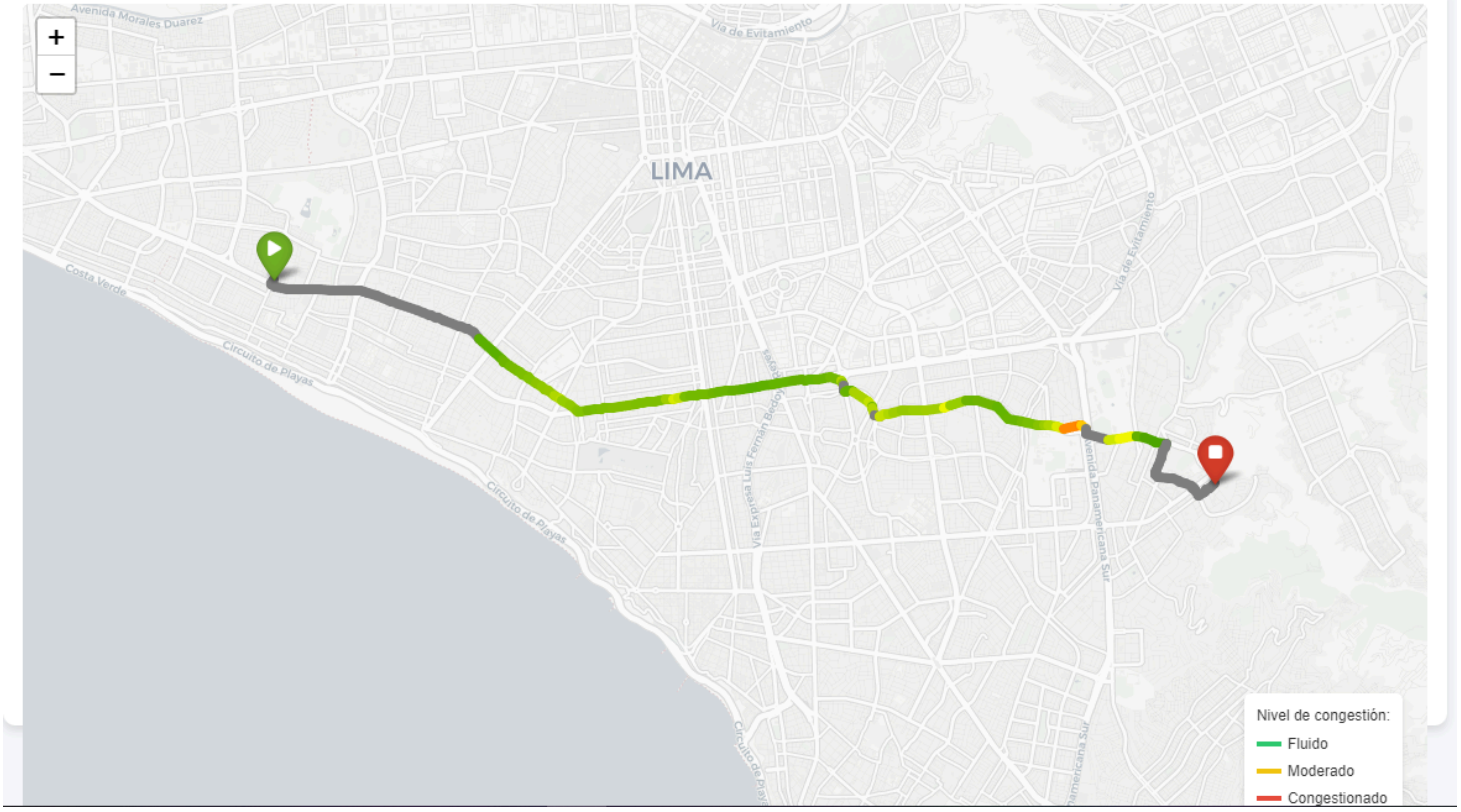
Pruebas de la app funcionando:

El upecino será capaz de poder cambiar la fecha de salida, para poder planificar su viaje, se mostrará claramente la congestión de las calles, tiempo de llegada estimada, además el cálculo de ruta está siendo calculado teniendo en cuenta la congestión del modelo.



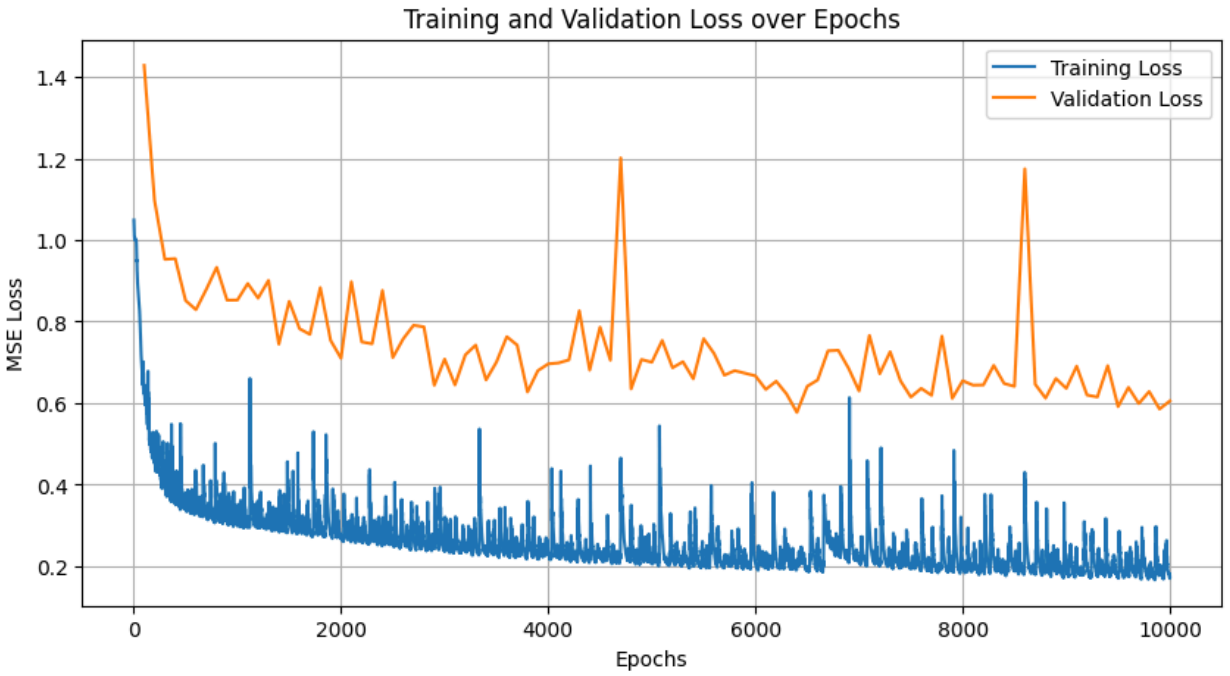
Sedes UPC disponibles:

- UPC San Isidro
- UPC San Miguel
- UPC Monterrico
- UPC Villa



Entrenamiento del modelo grande:

Learning Rate: 0.00455



Validación del error: 0.6051907539367676

3. Propuestas

La propuesta se basa en utilizar la red neuronal para que aprenda a poder reconocer ciertos patrones y ajustarse a los datos reales del tráfico de la ciudad de Lima Metropolitana.

Objetivo de la Propuesta

El objetivo de esta propuesta es desarrollar un sistema basado en **redes neuronales artificiales (RNA)** que permita predecir la **congestión de tráfico** de las principales calles de Lima para que los universitarios vayan de una sede a otra. Este modelo hará la tarea de predicción utilizando datos históricos obtenidos de APIs. Este modelo podrá ser utilizado por los usuarios para planificar rutas de forma eficiente, tomando en cuenta el tráfico previsto en diferentes momentos, también ofrecerá una visualización gráfica de las predicciones del tráfico, lo que facilita la interpretación de los resultados por parte de los usuarios universitarios.

Metodología a Utilizar

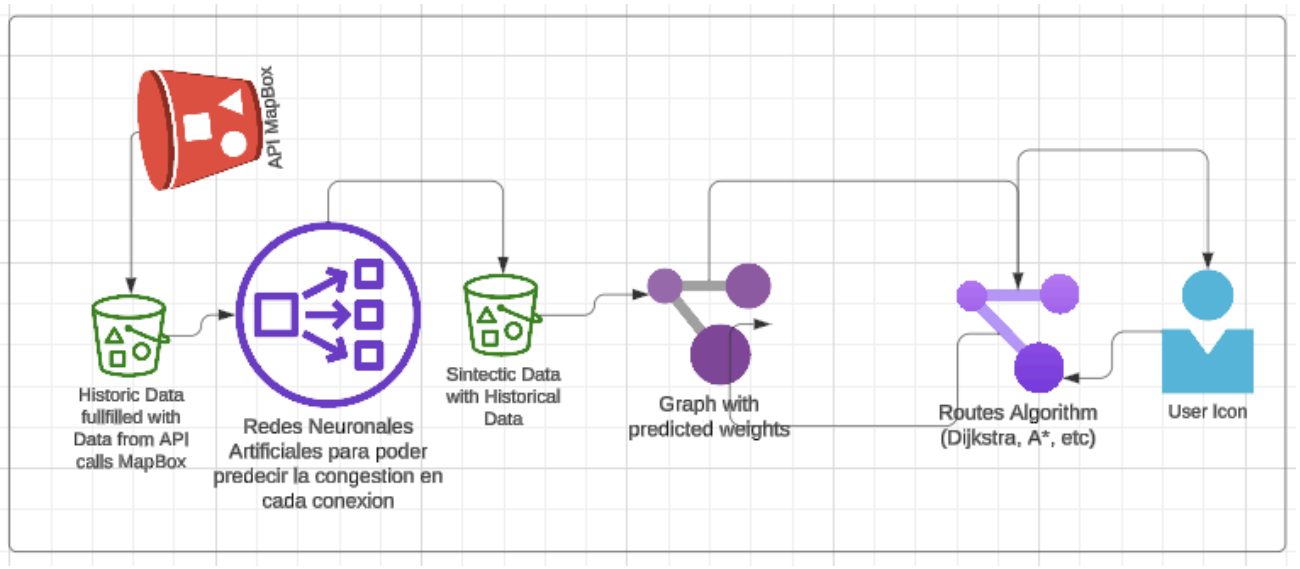
Para alcanzar este objetivo, se utilizarán las siguientes técnicas y pasos metodológicos:

1. **Recolección de Datos**:
    - Se obtendrán datos históricos de la API de **MapBox**. Estos datos incluirán:
        - Tiempos de viaje en diferentes momentos del día y días de la semana.
        - Dia
        - Hora
        - Longitud y Latitud
        - Congestión de tráfico en las calles de Lima Metropolitana
        - Distancia
        - Congestión de Tráfico en un rango de 0 a 100, 0 estaría sin tráfico, 100 cuando hay mucho tráfico

2. **Preprocesamiento de Datos**:

- Se realizará un **análisis exploratorio de datos (EDA)** para entender mejor la distribución y las características de los datos recogidos. Este análisis permitirá identificar patrones de tráfico, estacionalidad y outliers.
- Los datos serán **normalizados** y transformados en formato adecuado para ser utilizados en la red neuronal.
- Variables de entrada como la hora, el día, y el nombre de la calle o avenida se representarán en un formato numérico adecuado para el entrenamiento del modelo.
- Se implementará una fase de limpieza de datos donde se eliminarán los registros en donde haya congestión desconocida, y convertirá las variables en registros numéricos.

3. **Diseño de la Red Neuronal**:
   - Se desarrollará una **red neuronal artificial (RNA)** con múltiples capas para predecir la congestión. El número de **capas ocultas** será determinado mediante pruebas y ajustes experimentales, tomando en cuenta la complejidad de los datos.
   - Las **variables de entrada** incluirán: hora del día, día de la semana, nombre de la calle en esa específica conexión y coordenadas de la conexión (longitud y latitud).
   - La **variable de salida** será **la congestión numérica**.
4. **Entrenamiento y Validación**:
   - El modelo será entrenado usando los datos históricos para aprender a predecir el tiempo de viaje en diferentes condiciones.
   - El rendimiento del modelo se medirá usando métricas como el **error cuadrático medio (MSE)**.



4. Diseño del aplicativo

Los hiper parámetros a definir para la red neuronal son los siguientes:

Estructura de la red neuronal

- Número de capas escondidas
- Número de neuronas por cada capa
- Función de activación

Aprendizaje y optimización

- Tasa de aprendizaje
- Tipo de optimizador
- Inicialización de pesos
- Batch size

Para el ajuste de hiperparámetros se optó por el método de optimización bayesiana, debido a su capacidad para explorar de manera eficiente el espacio de búsqueda. Este enfoque resulta uno de los mejores en problemas de alta dimensionalidad. Al construir un modelo probabilístico, la optimización bayesiana concentra sus evaluaciones en las mejores regiones, evitando exploraciones que resulten computacionalmente

costosas. Este método además permite tomar decisiones informadas y reducir el riesgo de estancarse en óptimos locales. (IEEE, 2016).

A pesar de ser más eficiente que otros algoritmos, el costo computacional se incrementará a medida que se agreguen más hiper parámetros. Por ese motivo algunos fueron seleccionados para ser ajustados por optimización bayesiana, y el resto serán definidos de acuerdo a una investigación.

Hiper Parámetros como el número de capas escondidas, número de neuronas por capa y tasa de aprendizaje serán ajustados usando optimización bayesiana, debido a que los dos primeros definen la estructura y complejidad de la red neuronal. Una tasa de aprendizaje mal ajustada puede resultar en sobreajuste o sobreajuste, por lo tanto se debe priorizar su optimización con un algoritmo eficiente.

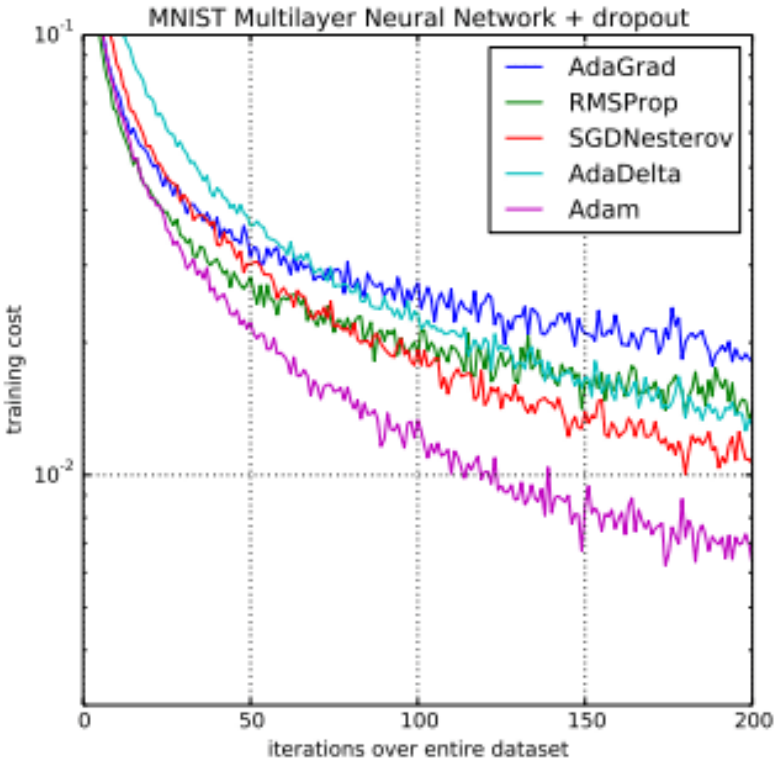
A continuación se sustentará la elección de los siguientes hiper parámetros:

Función de activación

ReLU es la función de activación más efectiva para una amplia gama de problemas (como clasificación y regresión) y la más usada para redes neuronales profundas (Szandała, 2021). La simplicidad de su función permite acelerar el proceso de entrenamiento de la red neuronal y previene el problema del gradiente desvaneciente.

Optimizador

El costo para entrenar usando Adam es menor comparado con los demás optimizadores.



Adam converge más rápido a comparación de otros optimizadores y es menos sensible a la configuración de otros hiper parámetros, lo que facilita su uso (Brownlee, 2021). Además, combina las ventajas del descenso por gradiente estocástico (SGD) y de los métodos de optimización de segundo orden.

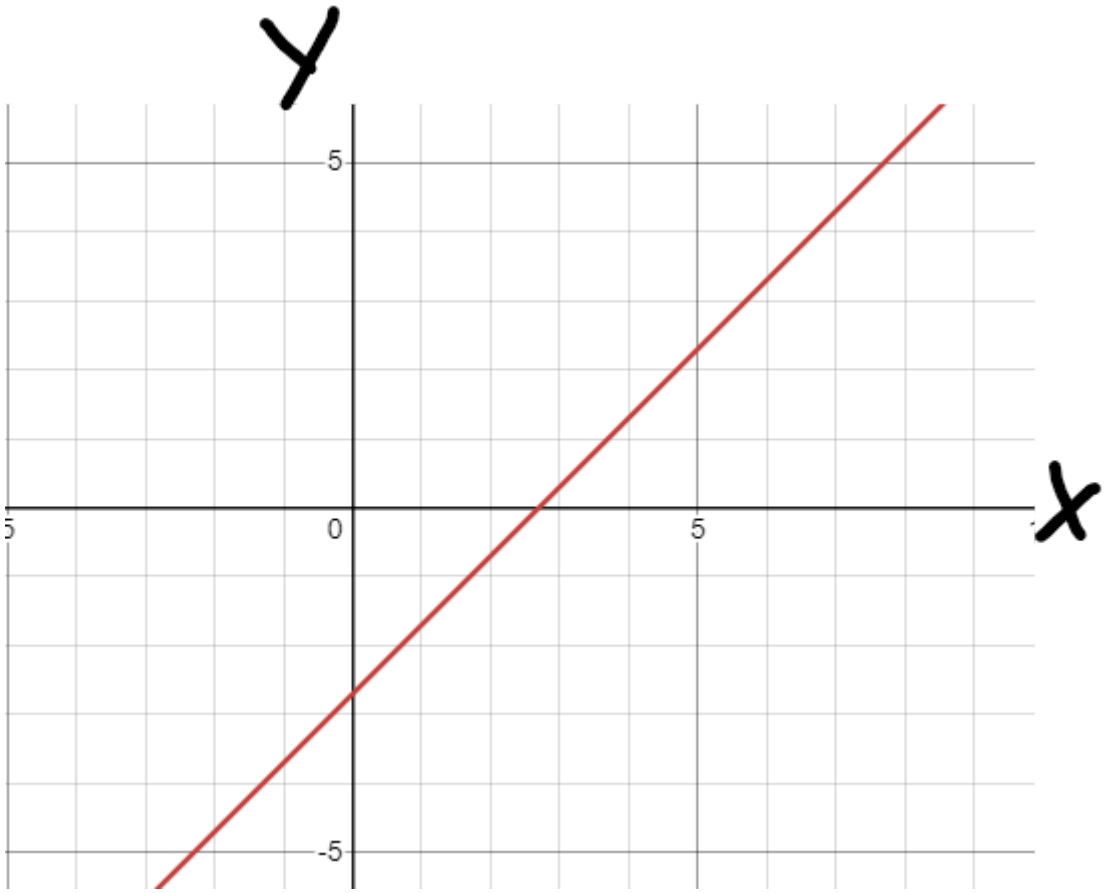
Pesos iniciales

El uso de una distribución uniforme para inicializar los pesos puede ayudar a prevenir el problema del "gradiente desvaneciente", ya que la distribución tiene un rango finito y los pesos se distribuyen uniformemente en ese rango.

5. Explicación Matemática

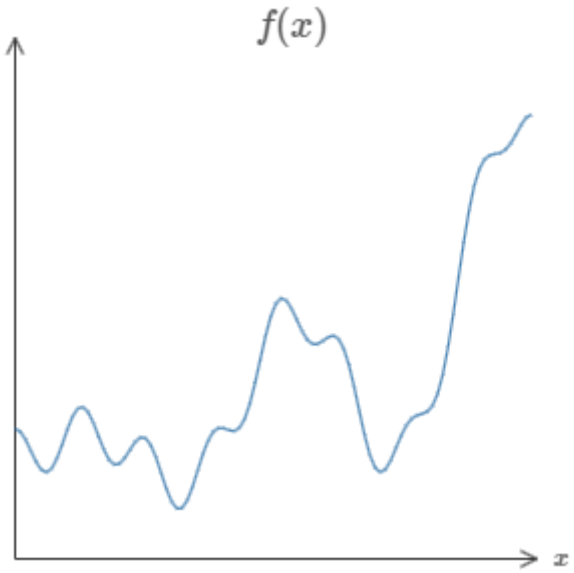
Usamos redes neuronales debido a que podemos predecir estos valores de congestión usando el aproximador de cualquier función continua no lineal. En realidad mostrar porque nos podemos acercar a funciones lineales es sencillo, pero para aproximarnos a una función no lineal es

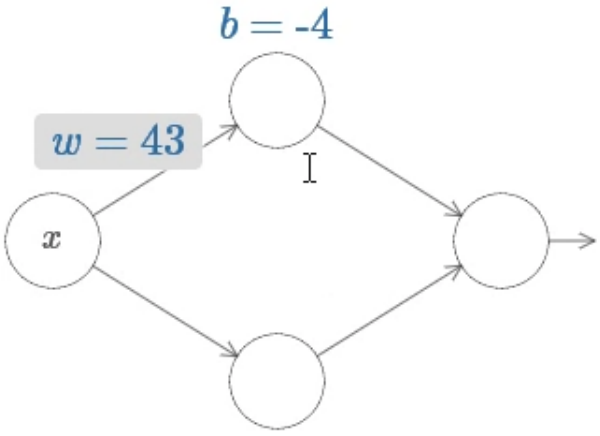
un poco más complejo. Nos podríamos imaginar una función lineal como una recta en el plano 2D en donde moviendo el W y el B podemos aproximarnos a cualquier función lineal dentro de el plano 2D en donde 1D seria nuestra X que queremos convertir a la Dimensión Y. Calculamos la relación que hay entre estas dos dimensiones.

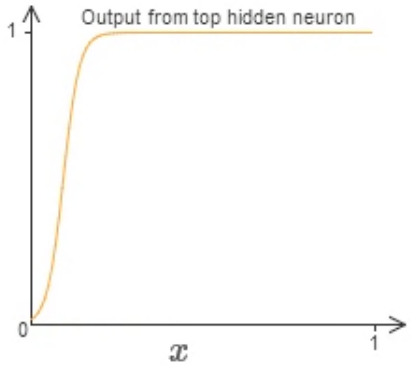


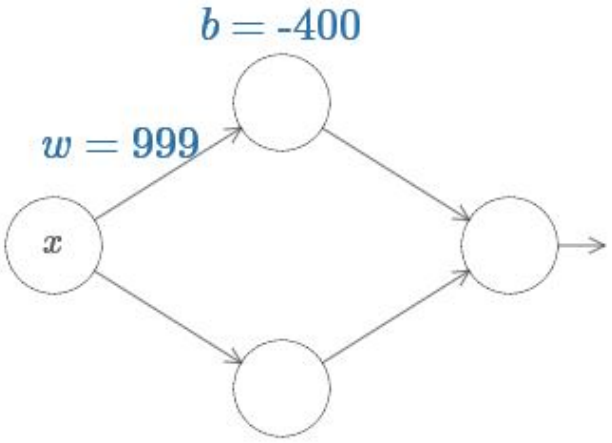
En este caso la aproximación será al 100%, aunque en funciones no lineales esto no será posible debido a que con redes neuronales necesitaríamos una cantidad infinita de neuronas en la primera capa para partir la función hasta que esta sea lineal. Claro, si partimos la función no lineal hasta que esta se parezca lineal, podríamos predecir funciones no lineales. Si nos vamos al infinito entonces podríamos predecir aún mejor la función no lineal. Aunque esto sería solo para poder acercarnos a la función no lineal, pero en la vida real, hay outliers y mucha variación que no nos permitirá inclusive con infinitas neuronas acercarnos por completo a la relación de estos datos(x, y).

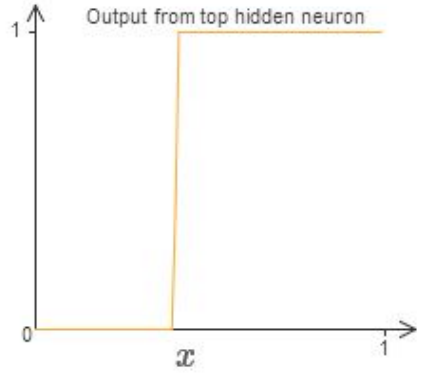
Para acercarnos a estas funciones no lineales necesitaremos funciones de activación para poder ignorar ciertas partes de la neurona. En las imágenes se está usando la función de activación Sigmoide.

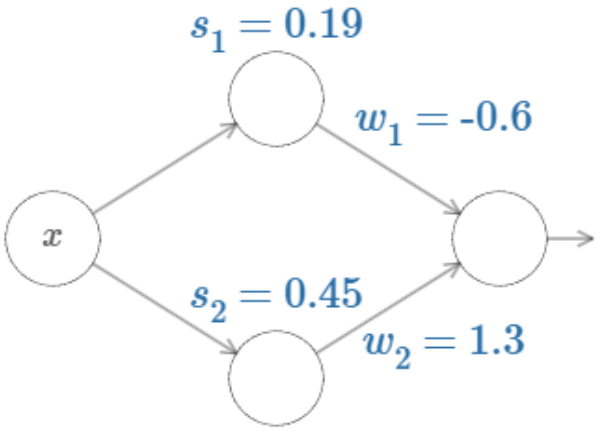


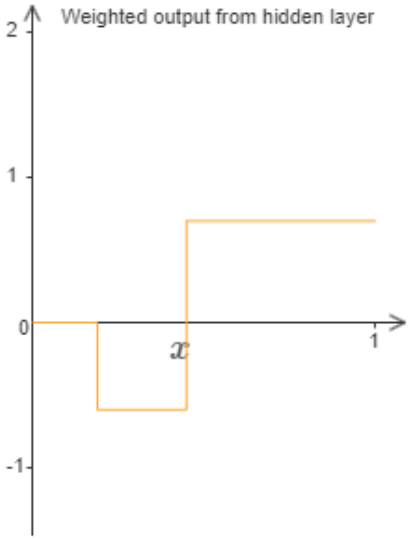


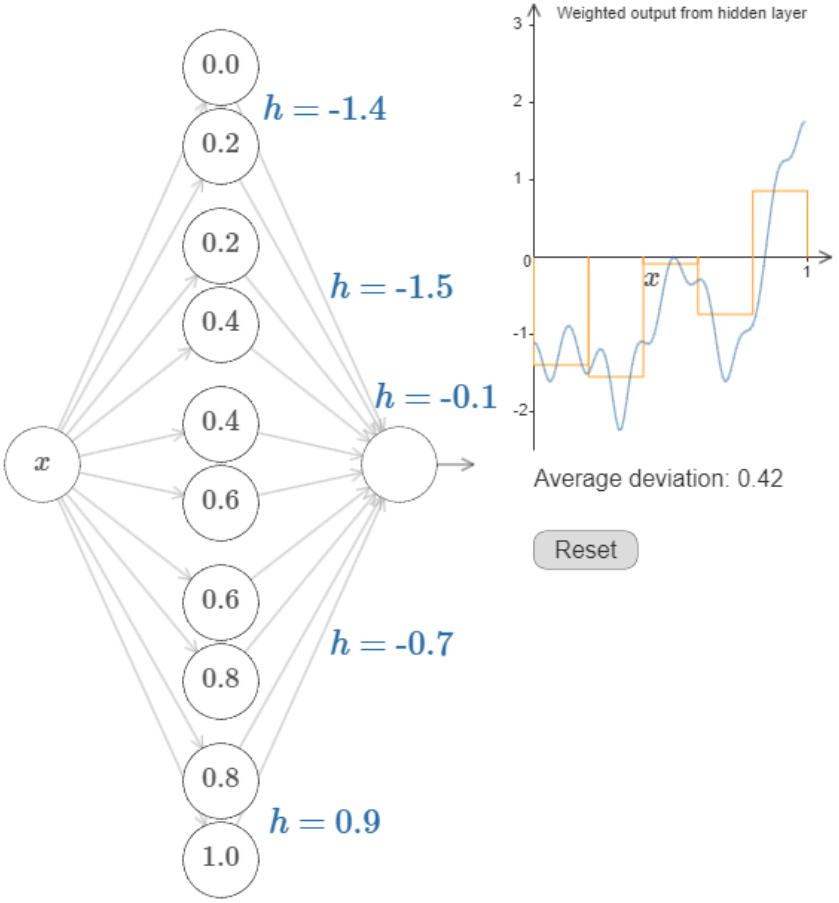


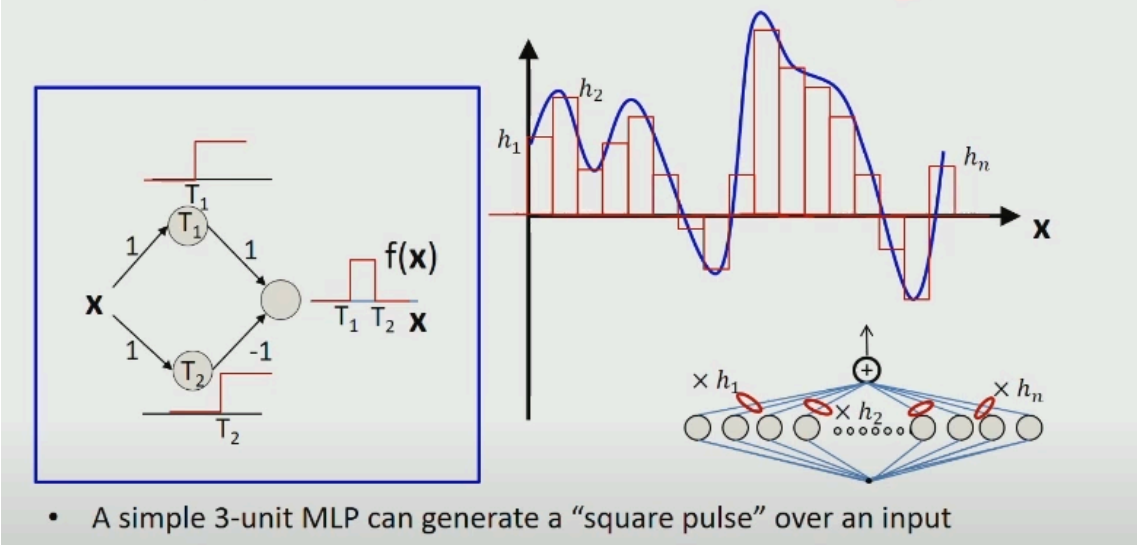

$b = -400$
$w = 999$
x


Output from top hidden neuron
1
0
x
1

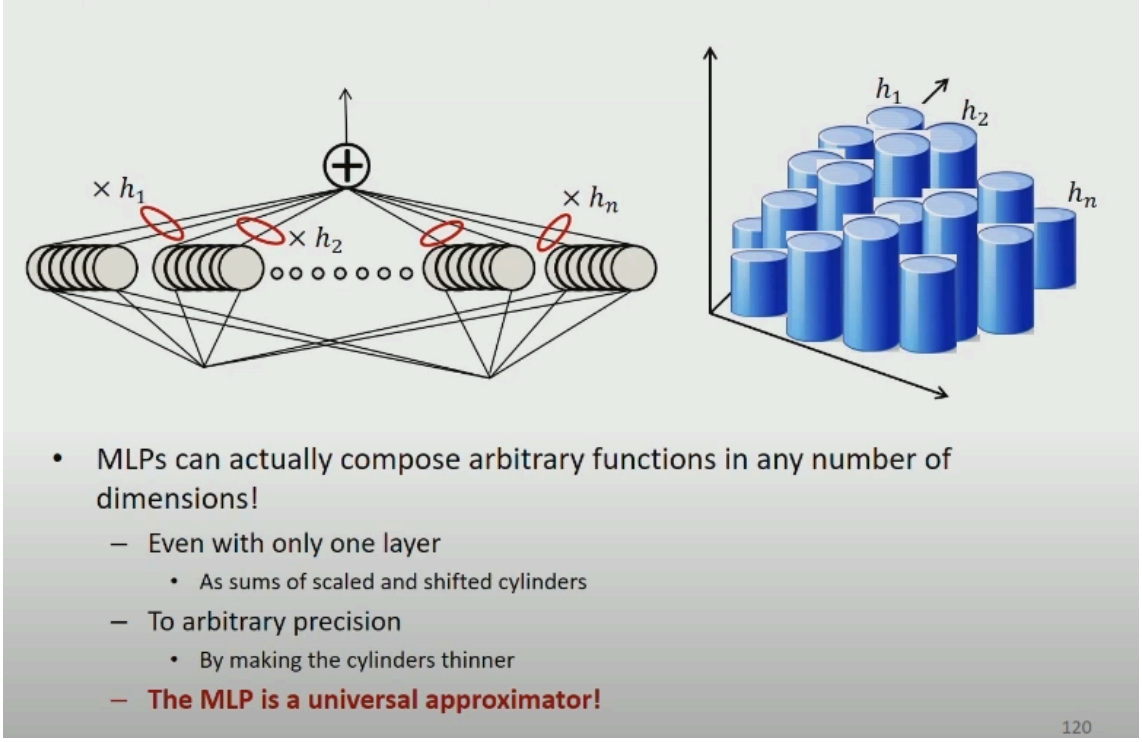

$s_1 = 0.19$
$w_1 = -0.6$
x
$s_2 = 0.45$
$w_2 = 1.3$

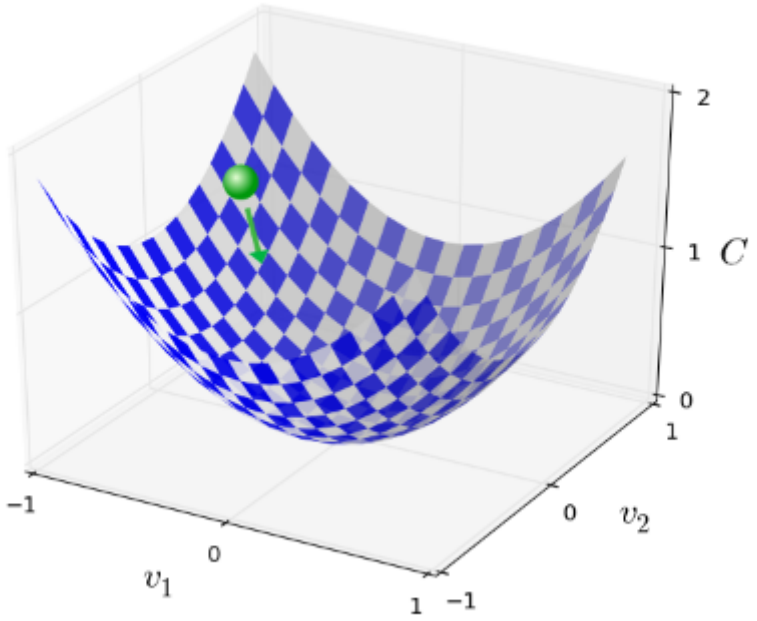

Weighted output from hidden layer
2
1
0
-1
x
1





- A simple 3-unit MLP can generate a "square pulse" over an input





Esta bola representa como el algoritmo va acercándose a la relación de los datos entre X e Y. Poco a poco se va aproximando a la función que

queremos llegar. Una vez tengamos esta función multidimensional podremos convertir los valores X a Y con una cierta tasa de error por supuesto. Si hay una buena relación entre X e Y, la tasa de error será baja. Para nuestros propósitos de predecir la congestión esto es importante. Debido a que habrá calles en donde la congestión es el día a día en algunos puntos de la calle a cierta hora. Eso es lo que intentaremos predecir con el algoritmo de redes neuronales.

Aparte de la utilización de Redes neuronales también usaremos estos algoritmos:

SMA* con Index Priority Queue.

La estructura de Index Priority Queue nos permite ahorrarnos y optimizar tanto la memoria como el tiempo de búsqueda de la ruta más corta. Esto lo hace al actualizar rápidamente y adecuadamente al nodo que está actualmente en la estructura de datos. Es decir si tenemos a un nodo que está actualmente en la estructura de datos y tiene peso mayor, y hemos encontrado un camino menor hacia ese nodo, vamos a actualizar inmediatamente a ese nodo con su nuevo costo menor. Es decir cada vez que nos expandimos hacia los vecinos, chequeamos si ya hay un camino hacia ese nodo, y si es que hay lo actualizaremos en Log(N) si el nuevo costo es menor. Esto representa una optimización especialmente si el grafo es denso (WilliamFiset, 2020).

6. Selección de Algoritmos

La selección de los algoritmos se basa en la necesidad de manejar una gran cantidad de datos de tráfico en tiempo real y realizar predicciones precisas de congestión. Se eligió una red neuronal artificial con capas ocultas, debido a su capacidad para aproximarse a funciones no lineales y ajustar dinámicamente las predicciones en condiciones variables. La arquitectura emplea 4 capas ocultas con activación ReLU, optimizando tanto el rendimiento como la precisión de predicción de congestión. Además, se utiliza el optimizador Adam para una convergencia rápida, lo que es ideal para grandes volúmenes de datos. Para la búsqueda de rutas, el algoritmo SMA* Se seleccionó por su eficiencia en el consumo de memoria.

7. Validación de resultados y pruebas

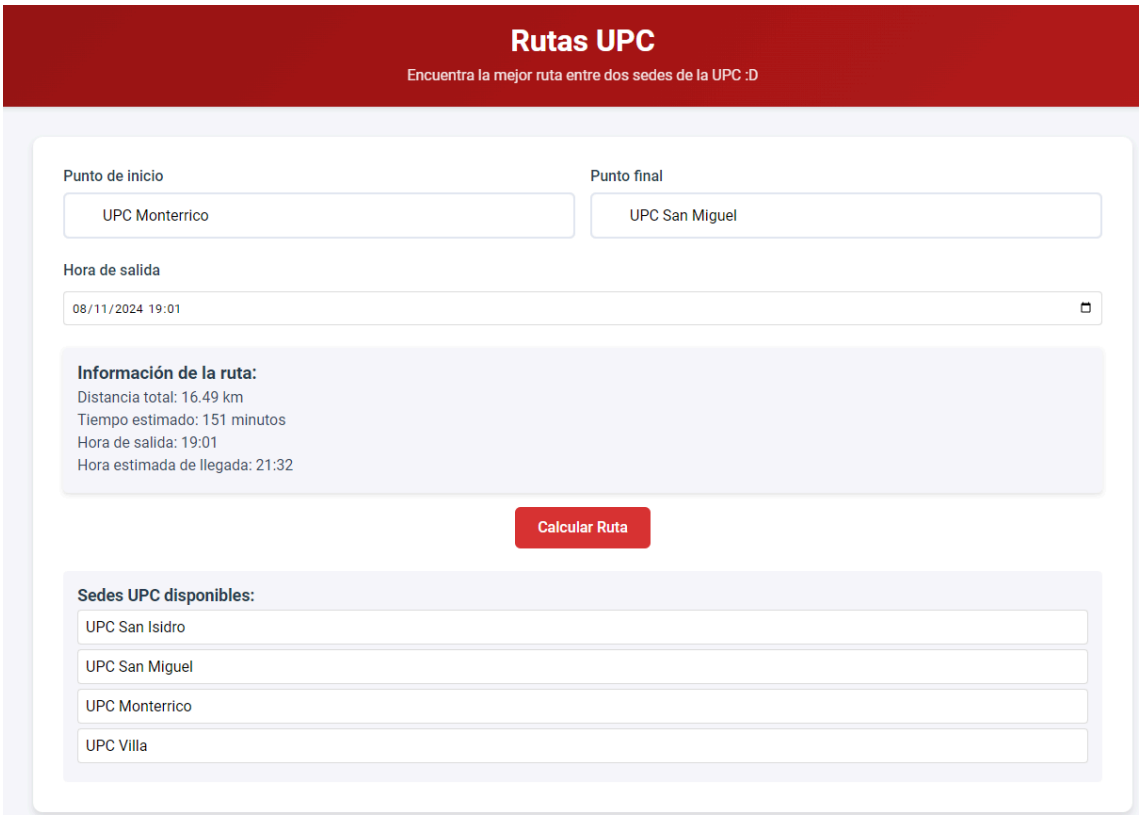
Al momento de iniciar la aplicación se deberán ingresar los siguientes datos de entrada:

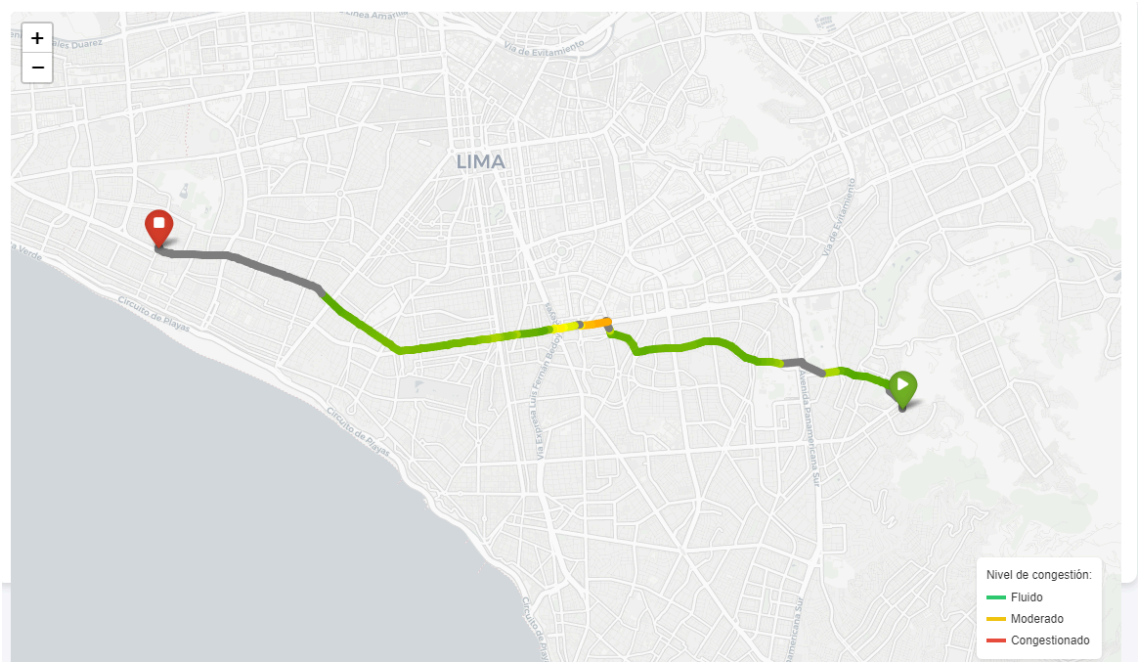
- Punto de inicio
- Punto de llegada
- Fecha de salida
- Hora de salida

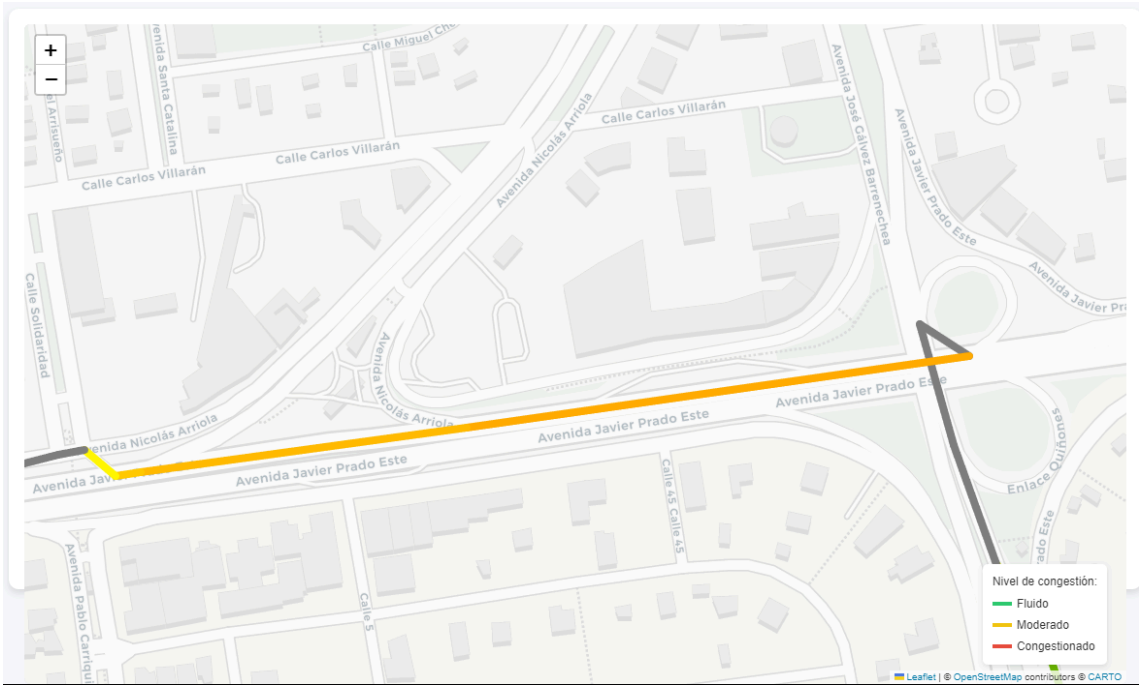
Mientras tanto las salidas indicarán:

- Información de la ruta
- Mapa

Para la siguiente prueba el punto de inicio es UPC Monterrico y el punto de llegada, UPC San Miguel. La fecha de salida es el 8 de noviembre a las 7:01 pm.







De acuerdo a la predicción el tiempo de llegada sería a las 9:32 pm con 151 minutos, también se obtendrá la distancia total recorrida en kilómetros (16.49). El mapa además muestra la mejor ruta y el nivel de congestión de cada zona. Para la Avenida Javier Prado Este el modelo predijo un nivel de congestión moderado.

8. Conclusiones

- Esta aplicación permitirá a los estudiantes planear sus rutas de forma eficiente. Con una predicción basada en datos históricos los estudiantes tienen la posibilidad de planificar con anticipación, de esta forma evitarán contratiempos causados por el tráfico.
- Al momento de planificar la arquitectura es importante investigar acerca de otras arquitecturas creadas para una solución similar y leer acerca del desempeño de los modelos como una guía.
- El sistema podría ser optimizado y expandirse incluyendo otras variables que puedan influir en la congestión del tráfico como eventos locales y accidentes.
- Este proyecto demuestra que la integración de datos históricos, combinada con algoritmos avanzados como redes neuronales y SMA*, puede ofrecer soluciones eficientes a problemas cotidianos como el tráfico en ciudades complejas.
- La integración de herramientas de planificación, como el uso de la API de MapBox, destaca la importancia de las plataformas abiertas y colaborativas para agilizar el desarrollo de proyectos tecnológicos en entornos urbanos.

9. Referencias Bibliográficas

Mapbox. (s.f.). Optional parameters for the mapbox/driving-traffic profile. Mapbox Directions API. Recuperado el 22 de septiembre de 2024, de https://docs.mapbox.com/api/navigation/directions/#optional-parameters-for-the-mapboxdriving-traffic-profile

Mapbox. (2024). Pricing. [Online]. Disponible en: https://www.mapbox.com/pricing#directions-api

Repositorio del Código del Proyecto:

https://github.com/forestgump22/TP-AI

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE, 104(1), 148–175.

Szandała, T. (2021). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In: Bhoi, A., Mallick, P., Liu, CM., Balas, V. (eds) Bio-inspired Neurocomputing. Studies in Computational Intelligence, vol 903. Springer, Singapore. https://arxiv.org/pdf/2010.09458

Bronwlee, J. (2021, 13 de enero). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Machine Learning Mastery. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Luschi, C. & Masters, D. (2018). Revisiting small batch training for deep neural networks. https://arxiv.org/abs/1804.07612

Carnegie Mellon University Deep Learning. (2019, 31 agosto). *Lecture 2 The Universal Approximation Theorem* [Vídeo]. YouTube. https://www.youtube.com/watch?v=lkha188L4Gs

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/chap1.html

WilliamFiset. (2020, 19 abril). *Indexed Priority Queue (UPDATED) | Data Structures* [Vídeo]. YouTube. https://www.youtube.com/watch?v=jND_WJ8r7FE