JEFFERSON RESENDE FELIX

DESENVOLVIMENTO DE UM FRAMEWORK DE ROTEIRIZAÇÃO A PARTIR DE MAPAS DIGITAIS: ESTUDOS E APLICAÇÃO EM SISTEMA EMBARCADO

LAVRAS – MG

2021

JEFFERSON RESENDE FELIX

DESENVOLVIMENTO DE UM FRAMEWORK DE ROTEIRIZAÇÃO A PARTIR DE MAPAS DIGITAIS: ESTUDOS E APLICAÇÃO EM SISTEMA EMBARCADO

Trabalho de Conclusão de Curso em forma de Relatório Técnico de Estágio apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

Prof. Dr. Arthur de Miranda Neto

Orientador

LAVRAS – MG

2021

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Félix, Jefferson Resende

Desenvolvimento de um framework de Roteirização a partir de Mapas Digitais: estudos e aplicação em sistema embarcado / Jefferson Resende Félix. 1^a ed. rev., atual. e ampl. – Lavras : UFLA, 2021.

68 p. : il.

TCC (graduação) –Universidade Federal de Lavras, 2021.

Orientador: Prof. Dr. Arthur de Miranda Neto.

Bibliografia.

1.Simulação 2.SUMO 3.OpenStreetMap. 4.Inteligência Artificial.

CDD-808.066

JEFFERSON RESENDE FELIX

DESENVOLVIMENTO DE UM FRAMEWORK DE ROTEIRIZAÇÃO A PARTIR DE MAPAS DIGITAIS: ESTUDOS E APLICAÇÃO EM SISTEMA EMBARCADO

Trabalho de Conclusão de Curso em forma de Relatório Técnico de Estágio apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

APROVADA em 25 de Novembro de 2021.

Prof. Dr. Arthur de Miranda Neto	UFLA
Prof. Dr. Danilo Alves de Lima	UFLA
Eng. Artur Passos do Amaral	UFLA

Prof. Dr. Arthur de Miranda Neto
Orientador

LAVRAS – MG
2021

À minha família por todo carinho, amor e apoio em todos esses anos dedico.

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Geraldo (in memoriam) e Marlene, por todo o apoio e suporte que tive durante todos esses anos para um dia me tornar um engenheiro.

Aos meus avós, Eva e Sebastião (in memoriam), que também fizeram parte de minha vida e me ajudaram nesta conquista

À minha irmã Larissa, por toda parceria e amizade, desde que entrei na universidade, me apoiando em meus estudos.

Aos meus amigos presentes durante o curso, em especial ao Caio, Paulinho, Pedro Machado, João Pedro, Alexandra, Ian, Artur Amaral, companheiros de todos os momentos que sempre me motivaram a concluir meus objetivos.

Ao Prof. Dr. Danilo Alves de Lima pela atenção e ajuda prestada durante o período de estágio.

Ao meu orientador, Prof. Dr. Arthur de Miranda Neto, que me auxiliou no desenvolvimento deste trabalho e na sua conclusão.

Não lute mais
Descanse
Não dê força para seus inimigos
Vença-os com o perdão
Não cultive a impaciência
Vença-a com a segurança
Não delapide a paz dos outros
Coopere com o silêncio
Não se afaste do seu coração
Una-se a si mesmo
Não dê trelas aos problemas
Vença-os com a luz interior
Não coopere com as críticas
Supere-as com seu desprezo
Não se deixe vitimar
Assuma sua liberdade de escolha
O bem é saber
que o único meio de vencer
É usar a inteligência
com compaixão
Por isso não lute mais
Descanse
(Luiz Gasparetto)

RESUMO

Em centros urbanos, onde habita a maior parte da população, existe um número elevado de veículos em circulação, sejam eles comerciais ou pessoais. Os prestadores de serviços de mobilidade lidam a todo tempo com questões complexas, para manter a rentabilidade das operações, atender aos critérios de sustentabilidade e garantir o nível de satisfação dos usuários. Nos últimos anos, houve um aumento expressivo do número de demandas para o transporte de pessoas e entregas de encomendas, mas um número reduzido de prestadores dispõe de acesso a ferramentas de auxílio às decisões. Alguns fatores exercem grande influência na obtenção de indicadores positivos em termos de custos e com relação às emissões de gases de efeito estufa. É sabido, porém, que tais impactos podem ser minimizados com o emprego de metodologias que consideram um estilo mais econômico de condução. O presente trabalho apresenta os resultados preliminares de um *framework* de simulação de rotas mais eficientes, por diferentes critérios. Para tanto, experimentos foram realizados, incluindo testes em sistema embarcado. Por fim, um estudo inicial explora a aplicação de uma rede neural a partir de dados do *framework*, tendo como base as acelerações e posições de um veículo simulado e real.

Palavras-chave: Simulação.SUMO.*OpenStreetMap*. Inteligência Artificial.

ABSTRACT

In urban centers, where most of the population lives, there is a high number of vehicles in circulation, whether commercial or personal. Mobility service providers are constantly dealing with complex issues to maintain the profitability of operations, meet sustainability criteria and ensure the level of user satisfaction. In recent years, there has been a significant increase in the number of requests for the transport of people and deliveries, but a reduced number of mobility service providers have access to decision-making tools. Some factors have a great influence on obtaining positive indicators in terms of costs and in relation to greenhouse gas emissions. It is known, however, that such impacts can be minimized with the use of methodologies that consider a more economical driving style. The present work presents the preliminary results of a more efficient route simulation framework, by different criteria. For that, experiments were carried out, including tests in an embedded system. Finally, an initial study explores the application of a neural network from framework data, based on the accelerations and positions of a simulated and real vehicle.

Keywords: Simulation. SUMO. OpenStreetMap. Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

LISTA DE ABREVIATURAS E SÍMBOLOS

AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
DES	*Data Encryption Standard*
PIB	*Produto Interno Bruto*
OSM	*OpenStreetMap*
RNN	*Recurrent Neural Networks*
RNA	*Redes Neurais Artificiais*
UTM	Universal Transversa de Mercator
XML	*eXtensible Markup Language*

1 INTRODUÇÃO

A produção de veículos automotores propagou-se de forma acentuada no final do século XX. Com isso, a busca por novos métodos de prevenção de acidentes e otimização de custos operacionais é necessária. O cenário atual do mercado brasileiro é reconhecido por ser cada vez mais inovador e diversificado em termos de modalidades de transporte, sendo que a frota ultrapassa 47 milhões de veículos no país, sendo que 29 milhões são carros (ANTP, 2016).

Nesse sentido, manter um controle sobre esses veículos pode refletir em um trânsito mais seguro. Ademais, questões relacionadas aos custos variáveis são desafiadores, uma vez que o custo de combustíveis se elevaram.

Os custos logísticos de transporte, incluindo serviços administrativos, consomem cerca $12,7\%$ do produto interno bruto brasileiro (PIB) , que corresponde ao principal indicador de desempenho econômico do país. Em 2014, essa mensuração foi equivalente a 749 bilhões de reais, o que impacta na competitividade brasileira. Nos Estados Unidos, o custo logístico corresponde a $7,8\%$ do PIB (CNT, 2016).

Assim, garantir meios para redução de custos permite a sobrevivência da empresa no mercado logístico. Os sucessivos aumentos dos preços dos combustíveis impactam em até 12% no preço dos fretes, principalmente em setores dependentes, por exemplo no e-commerce. Em 2021 o óleo diesel acumulou uma alta de 50% e a gasolina em 73%, valores que possivelmente serão repassados ao consumidor final (OTEMPO, 2021).

Para contornar essa situação, surge o chamado eco-condução, um recurso para otimizar os custos e tempo de condução, aliado ao respeito a natureza, e com a vantagem de ser implantável em qualquer veículo (GRUPOQUALITY, 2021). De modo geral, o eco-condução é uma opção considerada eficiente de direção com a finalidade de utilizar menos combustível, para atravessar as mesmas distâncias e que permite uma redução de até 20% no consumo total (RENAULT, 2021).

Pesquisas lideradas pelo Centro de Energias Renováveis da Grécia (CRES), treinaram motoristas de ônibus em um curso para execução da eco-condução. Os resultados foram contabilizados como pré e pós-curso. Os cientistas obtiveram uma redução de $4,5$ % no consumo de combustível, apenas alterando os hábitos de direção dos motoristas (ZARKADOULA; ZOIDIS; TRITOPOULOU, 2007).

Nesse contexto, o desenvolvimento de *frameworks* com essa finalidade pode impactar diretamente na redução de custos, tornando as empresas mais competitivas e eficientes do ponto de vista empresarial. Além do que, a criação de mecanismos para predição de custos variáveis melhora a organização contábil das empresas.

1.1 Seven Always Tecnologia LTDA (SEVEN GO)

A *SEVEN GO* é uma empresa brasileira que tem por objetivo propor soluções para frotas comerciais, para auxiliar seus clientes otimizarem seus custos e receitas. Um dos produtos da empresa, SEVEN GO Smart Edge, tem como objetivo minimizar os custos de logística ao calcular rotas baseadas em variáveis que normalmente não são calculadas por seus gestores: frenagens e acelerações desnecessárias, perfil topográfico da região trafegada em função da altitude, entre outras variáveis decisivas para o aumento dos custos referentes aos combustíveis.

Um dos pontos-chave mais importante do produto citado está na possibilidade da tomada de decisões após uma simulação coesa dessas variáveis. Nesse sentido, decisões mais rápidas e concisas são entregues em curto espaço, permitindo a prática do chamado eco-condução. Nesse contexto, é necessário desenvolver um recurso para geração de rotas que provê resultados antes de uma entrega logística se iniciar.

1.2 O SEVEN GO Smart Edge

O *On-Board Diagnostic* (OBD) é um sistema usado pelos veículos para autodiagnósticos, que auxilia descobrir e corrigir as falhas que estejam presentes no carro. O SEVEN GO Smart Edge é o modulo adotado para adquirir os dados dos veículos a partir da porta OBDII. O dispositivo descrito é apresentado na Figura 1.1.

Figura 1.1 – O SEVEN GO Smart Edge.

Fonte: SEVEN GO (2021)

1.3 Objetivos

Este trabalho tem por objetivo apresentar um relatório técnico com alguns dos principais elementos utilizados para o desenvolvimento de um *framework* para geração de rotas, resultado de atividades desenvolvidas durante um período de estágio na empresa SEVEN GO. Os objetivos específicos são definidos a seguir:

- Desenvolver um *framework* de simulação de rotas integrado com mapas digitais abertos;
- Gerar rotas otimizadas com base nos dados obtidos no passo anterior;
- Testar o *framework* em sistema embarcado, a partir de uma rotina computacional automatizada, e em conjunto com uma rede neural.

1.4 Justificativa

Boas práticas de condução no trânsito garantem uma maior eficiência energética ao tirar proveito de todo o potencial do veículo. Antecipar, por meio de simulação, diferentes rotas permite aumentar a competitividade dos provedores de mobilidade. Para as empresas, podem antecipar estratégias comerciais, bem como ter maior controle sobre os seus bens e recursos, além de ferramentas para instruir seus colaboradores.

Essa combinação de fatores, levaram a empresa SEVEN GO a desenvolver uma ferramenta para a geração de rotas mais eficientes, tendo investido na elaboração da proposta por meio da oferta de um estágio.

Desta forma, neste trabalho, no referencial apresentado são disponibilizados os elementos principais utilizados para sustentar a proposta apresentada. No capítulo de metodologia, a arquitetura geral do *framework* é apresentada em conjunto com os seus elementos mais importantes. Os resultados são discutidos na sequência. A conclusão apresenta as considerações finais.

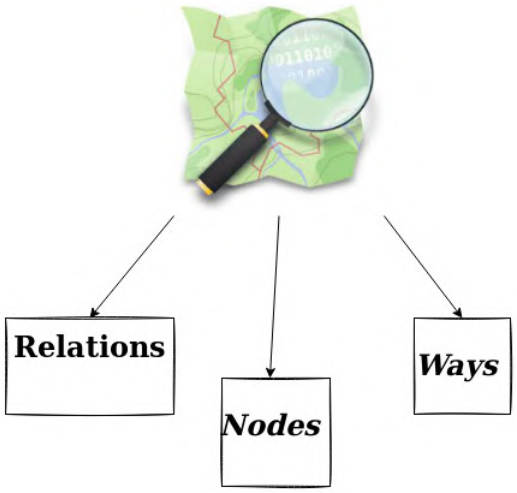
2 REFERENCIAL TEÓRICO

Neste capitulo são apresentados os elementos técnicos e científicos utilizados no desenvolvimento do *framework* proposto para automatização do simulador que será apresentado e a obtenção das rotas.

2.1 OpenStreetMap

O *OpenStreetMap* (OSM) é um projeto de mapeamento colaborativo criado com o objetivo de fornecer informações de georreferenciamento por meio de dados públicos fornecidos por seus usuários. Os dados fornecidos são gerenciados sob a licença da *Open Database License*, sendo administrada pela *OpenStreetMap Foundation*. O site oficial do *OpenStreetMap* permite realizar *downloads* de mapas no formato .OSM a partir da especificação da área desejada, de modo que os *downloads* podem ser realizados com o uso de sua API, permitindo automatizar processos. Os dados baixados possuem três tipos básicos de primitivas: *nodes*, *ways* e *relations*. Essas primitivas compõem o arquivo .OSM e seu formato de representação, normalmente, é o XML (*eXtensible Markup Language*) (BENNETT, 2010). A representação em XML é muito usada em linguagens de internet por possuir chaves, sendo possível ler atributos com facilidade. A estrutura dos dados do *OpenStreetMap* é apresentado na Figura 2.1.

Figura 2.1 – Estrutura dos arquivos do *OpenStreetMap*.



Fonte: Do autor (2021).

Os dados do *OpenStreetMap* são vistos como objetos e cada versão de um objeto recebe uma ID. Associado a esse objeto, existe a ID do usuário que permite localizar o último usuário que editou o mapa. Além disso, existe um atributo conhecido como *visible*, o qual é responsável por mostrar se a versão do objeto é a mais atual. Se o valor do atributo for igual a *FALSE*, isso indica que a versão do objeto não é a mais recente. Se o valor for igual a *TRUE*, indica a versão mais atualizada. Além disso, é possível, também, obter outras informações, tais como, a data e a hora de cada modificação realizada pelo usuário no objeto e o histórico das modificações.

2.2 A primitiva Node

Os nós ou *nodes* são dados primitivos do *OpenStreetMap* que possuem basicamente a função de indicar a localização no espaço (BENNETT, 2010). Além disso, é o único ente primitivo que possui informações de posição. De modo análogo, esses dados desempenham funções semelhantes as coordenadas $< x, y >$ pre-

sentes nas formulações matemáticas. Na Figura 2.2 esta ilustrado o trecho de um arquivo .OSM com a primitiva nó em destaque.

Figura 2.2 – Descrição do arquivo de coordenadas.

```
<osm version="0.6" generator="osmconvert 0.8.10" timestamp="2021-07-27T20:21:43Z">
        <node id="173170723" lat="-21.1761159" lon="-45.1251076" version="4" timestamp="2020-10-16T08:22:10Z" changeset="0"/>
        <node id="173170739" lat="-21.1867763" lon="-45.1172426" version="4" timestamp="2013-12-08T01:19:03Z" changeset="0"/>
        .                       .                       .                       .                       .               .
        .                       .                       .                       .                       .               .
        .                       .                       .                       .                       .               .
        .                       .                       .                       .                       .               .
        <node id="173170756" lat="-21.2306464" lon="-45.1342217" version="5" timestamp="2020-09-01T08:59:15Z" changeset="0"/>
</osm>
```

Fonte: Do autor (2021).

A Figura 2.2 apresenta dados pertencentes à região de Lavras. Cada *node* fornece informações da latitude e longitude, com precisão de 7 casas decimais, bem como a versão e a data e hora (*timestamp*) da última modificação. No entanto, uso de dados do *OpenStreetMap* necessita de cuidados especiais, principalmente em relação aos *nodes* repetidos em que a API do *OpenStreetMap* não verifica, sendo necessário o uso de outras ferramentas disponíveis na WEB com o objetivo de solucionar o problema (BENNETT, 2010).

2.3 A primitiva way

Caminhos ou ways são definidos por um conjunto de *nodes*, a partir de um conjunto de pontos no espaço reunidos podem ser geradas rotas, estradas, vias para navegação ou mesmo para delimitar uma área de interesse. Para se traçar um caminho é necessário apenas dois nós. Um nó pode pertencer a mais de um caminho. Caso aconteça do *node* de origem ser igual ao de destino do caminho, tem-se a representação de uma área. Pode-se usar o caminho para descrever figuras geométricas mais complexas, como o contorno de um prédio ou a delimitação de uma área de mata por exemplo. Além disso, os caminhos podem se cruzar sem necessariamente estar unidos. Esse caso somente ocorre se um ou mais nós são compartilhados entre os caminhos, por exemplo, caso haja uma ponte por dois caminhos diferentes não há como parar no meio da ponte. Nesse caso, a origem e

o final da ponte são pontos compartilhados, na qual podem ser unidos. Como os dados foram criados por vários usuários, pode haver inconsistência nas relações dos nós. Para isso, existem ferramentas adequadas para detecção e correção dos erros.

2.4 A primitiva relation

Relations ou relações são um conjunto de nós, caminhos ou mesmo outras relações, que possibilitam que descrições mais complexas possam ser implementadas pelos mapeadores. Normalmente, são usadas para inserir restrições em um caminho, como por exemplo, na definição do sentido de uma via de mão única ou a direção adequada de uma rotatória.

2.5 A primitiva tag

A representação do mundo real em mapas é por meio dos nós, caminhos e relações. No entanto, é por meio do uso de *tags* que se especifica a função de cada ente primitivo. No site do *OpenStreetMap* existe uma documentação sobre o que cada *tag* representa, sendo que algumas *tags* podem não estar documentadas. Uma *tag* é escrita na seguinte notação chave=valor. Quando o objetivo não é especificar algum valor para a *tag*, utiliza-se a notação chave=*(apenas uma chave pode ser associada a um único valor). Outro aspecto importante é que o *OpenStreetMap* permite que os projetistas tenham a liberdade de criar suas próprias *tags* de acordo com seu interesse, ampliando as possibilidades no desenvolvimento de softwares.

Existem também serviços de terceiros que lançam estatísticas sobre quais marcações estão sendo mais usadas. Um exemplo deste serviço é o *TagInfo*, que permite consultar exemplos de *tags* para ideias em projetos de autoria própria. Entretanto, a plataforma não oferece documentação sobre o significado da *tag* envolvida, sendo necessário consultar a documentação. Uma curiosidade ao se verificar a documentação é que as *tags* foram baseadas no sistema britânico (país originário

do *OpenStreetMap*), como por exemplo a "*highway*", que significa qualquer via de trânsito, seja ela rodovia, avenida, ruas ou até mesmo estradas rurais.

2.6 Simulation of Urban Mobility (SUMO)

O software *Simulation of Urban Mobility* (SUMO) é um ambiente de simulação para tráfego microscópico e fluxo de meios de transportes, permitindo a construção de modelos de simulação. Foi desenvolvido pelo *Institute of Transportation Systems*, no Centro Aeroespacial Alemão, cujo código do projeto é aberto e está disponível para *download* a qualquer interessado (FATECLOG, 2019).

2.6.1 Funcionamento básico do SUMO

Conforme discutido nas seções anteriores, é necessário realizar o *download* por meio da plataforma do *OpenStreetMap*, para que um arquivo com extensão .OSM seja gerado. Entretanto, antes que se possa utilizá-lo, é necessário realizar um pré-processamento com o objetivo de torná-lo legível para o SUMO. Para isso, é necessário utilizar as ferramentas NETCONVERT e POLYCONVERT para conectar o interfaceamento entre o SUMO e o mapa obtido no *OpenStreetMap*. A seguir, são apresentados os comandos que realizam as funcionalidades citadas.

Primeiramente, é imprescindível realizar o *download* de dois arquivos (osmNetconvert.typ.xml e typemap.xml) auxiliares que são utilizados para cumprir a tarefa. Esses arquivos já foram implementados por outros usuários e são necessários para o pleno funcionamento dos comandos descritos. Além disso, não é necessário realizar nenhuma alteração ou edição, apenas exige-se que sejam copiados dentro da pasta que contém o arquivo .OSM. Eles encontram-se disponíveis também para *download* na página oficial do SUMO.

- netconvert –osm-files map.osm -o test.net.xml -t osmNetconvert.typ.xml –xml-validation never

O primeiro parâmetro realiza a chamada do *framework NETCONVERT*. O segundo parâmetro –osm-files indica que um arquivo .OSM do *OpenStreetMap*, nomeado de map.osm, será carregado no NETCONVERT. Já o terceiro parâmetro -o está relacionado com a saída, ou seja, o comando gera uma rede (mapa legível ao SUMO) com o nome test.net.xml. Os dois últimos parâmetros são padrões para todas as execuções, sendo assim, é necessário apenas repeti-los todas as vezes que utilizar esse comando.

- polyconvert –net-file test.net.xml –osm-files map.osm –type-file typemap.xml -o map.poly.xml –xml-validation never

Novamente, o primeiro parâmetro realiza a chamada do *framework* em questão. O segundo parâmetro –net-file indica que um arquivo nomeado de test.net .file.xml será carregado no POLYCONVERT. Nota-se a importância da ordem de execução, pois se o comando do *framework* NETCONVERT não for sido invocado primeiro, não será criado o arquivo de extensão .NET.XML e, assim, um erro é apresentado no terminal. Quanto ao segundo parâmetro (-o), o mesmo indica a saída do comando que irá gerar uma rede (mapa legível ao SUMO) com o nome test.net.xml. O terceiro parâmetro também é padrão, bastando apenas repeti-lo em todas as execuções. Por fim, o parâmetro -o (saída) gera um arquivo com os polígonos (extensão .poly.xml) do mapa com o nome de map.poly.xml.

2.6.2 Geração de rotas aleatórias

Rotas aleatórias podem ser geradas com auxílio do *Python* e inseridas na simulação. Por exemplo:

- python randomTrips.py -n test.net.xml -r map.rou.xml -e 1000 -l –validate

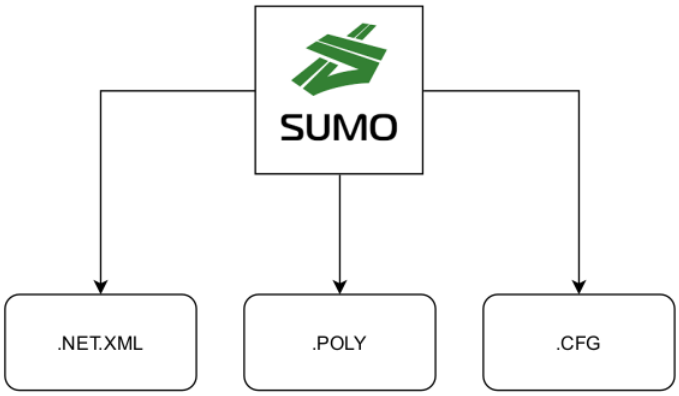
Entretanto, é primordial inserir o arquivo .NET.XML na mesma pasta do arquivo "randomTrips.py". Esta rotina é a responsável por gerar as rotas aleatórias.

Basicamente, seu funcionamento é semelhante aos comandos anteriores: o trecho “python randomTrips.py” relata a execução de um arquivo ".py". O parâmetro -n carrega um arquivo .NET.XML. O segundo parâmetro -r carrega o arquivo de rotas e, por fim, o trecho “-e 1000 -l –validate”, gera 1000 rotas aleatórias ao longo do mapa e com a validação se as informações do arquivo XML estão corretas.

2.7 Estrutura do SUMO

Conforme a seção anterior, é possível verificar que o SUMO trabalha com arquivos do tipo XML, sendo uma linguagem muito utilizada para serviços *World Wide Web* (WEB) ou para descrição de objetos com o uso do conceito de marcações. Existem três arquivos gerados após o pré-tratamento do mapa obtido no *OpenStreetMap*, a saber: “.ROU.XML”, “.NET.” e “.SUMO.CFG”. A seguir, são apresentados de maneira individual as peculiaridades de cada um. Ademais, na Figura 2.3 é apresentada a estrutura de funcionamento do SUMO.

Figura 2.3 – Estrutura de funcionamento do simulador.



Fonte: Do autor (2021).

2.7.1 O arquivo ROU.XML

No arquivo ROU.XML estão descritas todas as rotas do mapa, geradas de maneira manual ou de maneira automatizada pelo algoritmo “randomTrips.py”.

No SUMO, as ruas e avenidas são tratadas como objeto *edge*. Dessa forma, o objeto *edge* possui uma identificação própria não trivial para o entendimento humano. Também está descrito nesse arquivo algumas informações adicionais, bem como quais são os arquivos de entrada e de saída.

2.7.2 O arquivo NET.XML

O arquivo NET.XML, também conhecido por NETWORK, representa o conjunto de descrições de um mapa, possibilitando a obtenção de diversas informações, tais como:

- Quais vias estão interligadas entre si?
- A partir de que ponto uma via inicia e em qual termina?
- Quantas faixas existem na via?
- Quais veículos podem trafegar na via e qual a velocidade máxima permitida?
- Qual o tipo de via ? (ciclovia, avenida, trilho de trem, etc.)
- Qual a sua localização geográfica em latitude e longitude?

Com essas informações, é possível coletar elementos importantes para melhorar o conjunto de variáveis. Com isso, é importante ressaltar que existem outras descrições presentes nesse arquivo que podem ser exploradas. .

2.7.3 O arquivo SUMO.CFG

O arquivo SUMO.CFG, também conhecido como arquivo de configuração, é o menor dos três arquivos citados. Basicamente, ele descreve a localização dos outros dois arquivos e do arquivo adicional de polígonos, além de descrever o tempo de simulação que será decorrido e o passo que será empregado. Na Figura 2.4 é descrito um trecho desse arquivo para a região de Lavras.

Figura 2.4 – Descrição do arquivo de configuração.

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

    <input>
        <net-file value="test.net.xml"/>
        <route-files value="map.rou.xml"/>
        <additional-files value="map.poly.xml"/>
    </input>

    <time>
        <begin value="0"/>
        <end value="1000"/>
        <step-length value="0.1"/>
    </time>



</configuration>
```

Fonte: Do autor (2021).

Com os arquivos prontos, é possível carregar os mapas dentro do SUMO, conforme demonstrado na Figura 2.5.

Figura 2.5 – Interface gráfica do simulador.

Fonte: Do autor (2021).

2.8 A API Traas

O Traas (JAVA) ou TraCI (C++) são APIs muito utilizadas para controlar uma simulação de tráfego, alterando o comportamento da simulação. Além das APIs, existem também outras bibliotecas que possuem as mesmas funcionalidades apresentadas, porém em outras linguagens de programação. Todas compartilham a mesma arquitetura de trabalho no formato cliente/servidor, baseado no *Transmission Control Protocol* (TCP). Quando o SUMO é utilizado, uma forma rápida de iniciá-lo como servidor TCP é adicionando o parâmetro –remote-port <INT>, com um inteiro associado, representando a porta que será utilizada. É possível também associar múltiplos clientes a partir de diferentes *sockets*, bastando apenas inserir o parâmetro –num-clients <INT>. Um detalhe muito importante é que a simulação se inicia somente quando todos os clientes estiverem conectados, de modo que problemas de conexão nos *edges* envolvidos pode representar um problema.

2.9 Noções básicas de cartografia

Nesta seção são abordados alguns conceitos básicos de cartografia, utilizados durante o desenvolvimento do *framework*, especificamente ligados ao simulador.

2.9.1 Sistemas de referência

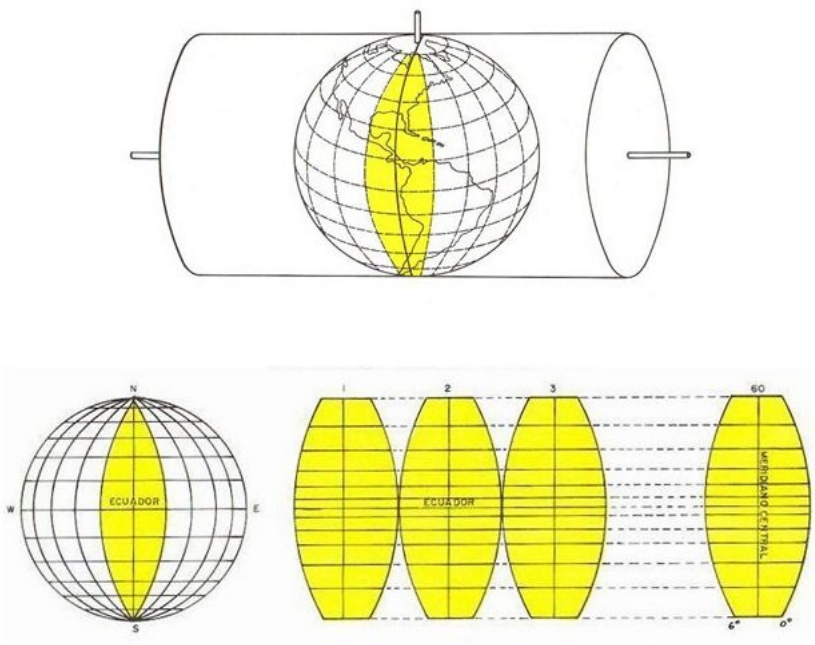
Os sistemas de georreferenciamento podem ser do tipo coordenadas geográficas (latitude e longitude), sendo comumente representadas em graus, minutos e segundos, ou por projeções (planares, cônicas ou cilíndricas), na qual podem ser representadas pelo sistema UTM (GEO, 2013). Dessa forma, é necessário definir um modelo matemático conhecido como DATUM, para representar as coordenadas de georeferenciamento ao nível do mar. Existem diferentes formas de representação de um DATUM, sendo que de forma análoga à matemática seriam

as coordenadas cartesiana, polar e esférica. Existem diferentes formas de representação de um ponto no espaço com a utilização dos DATUM (plural de DATA), os mais conhecidos são o Córrego Alegre, WGS84, SAD69 e SIRGAS2000 (SIRGAS).

2.9.2 O sistema de coordenadas Universal Transverse Mercator (UTM)

O sistema de coordenadas UTM consiste em realizar divisões baseadas em fusos, utilizando um cilindro como referência, o qual divide a Terra em 60 fusos conhecidos como zonas. Essas regiões possuem uma largura de 6 graus na longitude e 4 graus na latitude (EDISCIPLINAS, 2015). A Figura 2.6 ilustra esse conceito apresentado.

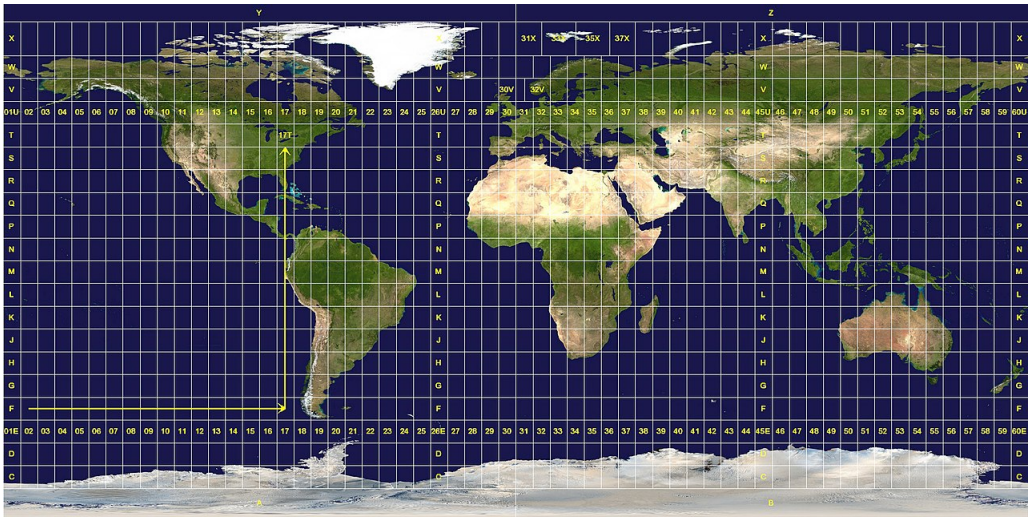
Figura 2.6 – A projeção cilíndrica obtida pelo sistema de coordenadas UTM.



Fonte: (BRANDALIZE,2021)

Os fusos pertencentes ao Brasil variam do 18 ao 25, sendo que alguns estados da federação estão localizados em mais de um fuso. Apesar de manter a precisão nos ângulos, o sistema UTM não é necessário nas medições de áreas e distâncias. A seguir, é apresentada a fórmula matemática que converte latitude e longitude para o fuso correspondente no sistema UTM, onde ϕ representa a longitude no sistema geográfico cartesiano com respeito ao *Greenwich*. Na Figura 2.7 são apresentados todos os fusos obtidos à partir da projeção cilíndrica descrita anteriormente.

Figura 2.7 – Planificação obtida pelo sistema UTM



Fonte: (WIKIMEDIA,2021)

As equações que relacionam os fusos UTM são apresentadas e relacionadas pelas Equações 2.1 e 2.2.

$$N = 30 + int\left(\frac{\phi}{6}\right) \tag{2.1}$$

A Equação 2.1 é valida para o hemisfério ocidental com respeito ao *Greenwich*.

$$N = 31 + int\left(\frac{\phi}{6}\right) \tag{2.2}$$

Da mesma forma que a Equação 2.1, a Equação 2.2 é somente válida para o hemisfério oriental, também com respeito ao *Greenwich*.

Caso a conversão seja no meridiano central (linha do Equador), o número do fuso pode ser obtido a partir da seguinte equação:

$$N = \frac{183 + \phi_{MC}}{6} \tag{2.3}$$

A medida do meridiano central também pode ser encontrada a partir do fuso, na qual a fórmula inversa é dada por:

$$\phi_{MC} = -183 + (6 \times N) \tag{2.4}$$

Além disso, no sistema UTM não há coordenadas negativas, de modo que é atribuída a origem como 500000 *Km* para abcissa E.

2.9.3 O sistema de coordenadas WGS 84

O sistema WGS84 é o padrão utilizado pelo GPS, também classificado como um sistema geodésico. Utiliza como referência um elipsóide (APRH, 2007). Tal fato se relaciona com a forma geométrica do mesmo, uma vez que o elipsóide se adequa melhor à forma da Terra e, com isso, consegue alcançar uma precisão muito maior se comparada com as outras projeções. O WGS84 foi instituído pelo Departamento de Defesa Americano (DOD) na década de 60 e tinha como meta a navegação e localização em qualquer parte do globo, sendo caracterizado também como um sistema geocêntrico (APRH, 2007).

2.10 Conceitos de criptografia

Nesta seção são descritos alguns conceitos importantes de criptografia utilizados para a integração e comunicação do *framework* com outros sistemas.

2.10.1 O algoritmo Advanced Encryption Standard (AES)

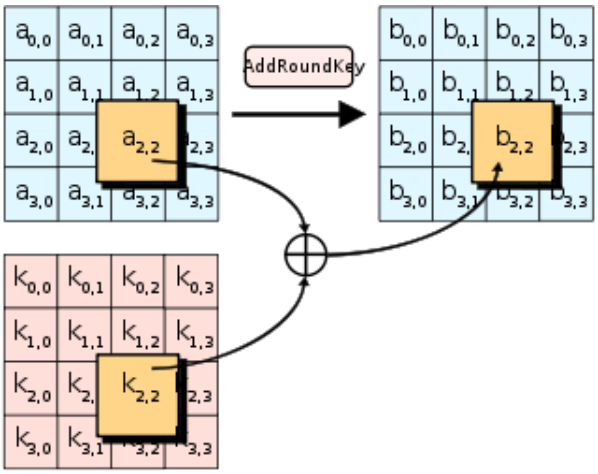
O algoritmo AES surgiu a partir de um concurso idealizado pelo *National Institute of Standards and Technology* (NIST) para substituir o seu antecessor, o algoritmo *Data Encryption Standart* (DES), o qual não era fidedigno em questões de segurança devido à evolução da capacidade computacional. O ano de 1997 foi marcado pelo início do seu desenvolvimento, de modo que as exigências iniciais eram que o AES deveria ser um algoritmo de bloco composto por 128 bits, público e com chaves variáveis de 128, 192 e 256 bits (UFRJ, 2005). Com isso, em agosto do ano seguinte, houve a Primeira Conferência dos candidatos aptos ao AES (AES1), sendo que no ano seguinte, houve a segunda conferência (AES2) e os resultados obtidos seguiram para análise de toda comunidade de criptógrafos.

Após a segunda etapa, foram selecionados cinco algoritmos finalistas: MARS, RC6, Rijndael, Serpent e Twofish, de modo que em outubro de 2000 o algoritmo vencedor foi o Rjndael (BIBLITA, 2021). A seguir, são descritos as etapas e o funcionamento do algoritmo.

2.10.2 A etapa AddRoundKey do algoritmo AES

Nessa etapa, a chave é dividida em subchaves de 1 byte, de modo que os elementos a_{ij} são chamados de estados, com tamanho 1 *byte*. Durante esse processo, apresentado na Figura 2.8, é realizada a operação conhecida como "ou exclusivo" (XOR - $(\oplus)$) para todo a_{ij} e K_{ij} conforme ilustrado nessa mesma figura, resultando nos elementos b_{ij}.

Figura 2.8 – Etapa AddRoundKey do algoritmo AES.

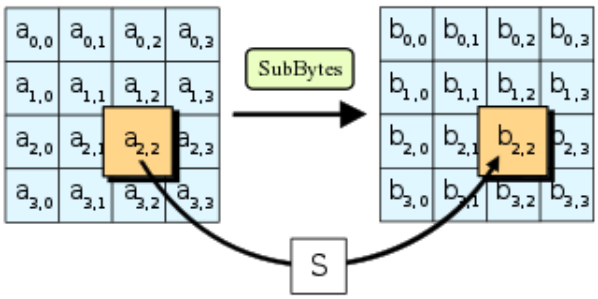


Fonte: (MATCRYPTO, 2021).

2.10.3 A etapa SubBytes

Nessa etapa, para cada *byte* do estado é aplicada uma função $S(\mathbf{x})$ com a finalidade de eliminar a linearidade do sistema e, assim, evitar pontos fixos que seriam suscetíveis a ataques de identificação de padrões. Os estados são ilustrados na Figura 2.9 e, nada mais é, que uma sequências de 8 bits representados pelos elementos a_{ij} e b_{ij}, obtidos a partir do objeto que será criptografado.

Figura 2.9 – Etapa SubBytes do algoritmo AES.

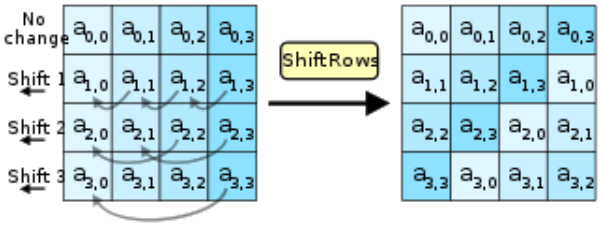


Fonte: (MATCRYPTO, 2021).

2.10.4 A etapa ShiftRows

Nessa etapa, em cada *byte* de estado é realizado um *shift* à esquerda de $N-1$ referente a linha, onde N representa a linha em questão. Por exemplo, a primeira linha não sofre alteração, a segunda sofre um *shift* de um *byte* e assim por diante, de acordo com a posição das próximas linhas, conforme Figura 2.10.

Figura 2.10 – Etapa ShiftRows do algoritmo AES.

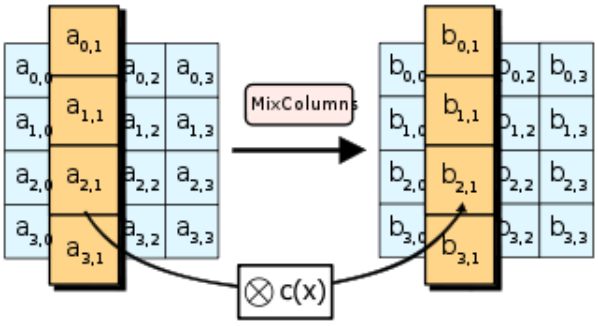


Fonte: (MATCRYPTO, 2021).

2.10.5 A etapa MixColumns

Nessa etapa, cada coluna do vetor de estados é multiplicado por uma matriz fixa de elementos c_{ij}. Sua principal função é adicionar uma cifra de forma aleatória, vez que diferentes combinações de c_{ij} garantem resultados totalmente diferentes ao final do processo. A etapa elucidada é apresentada na Figura 2.11.

Figura 2.11 – Etapa MixColumns do algoritmo AES.

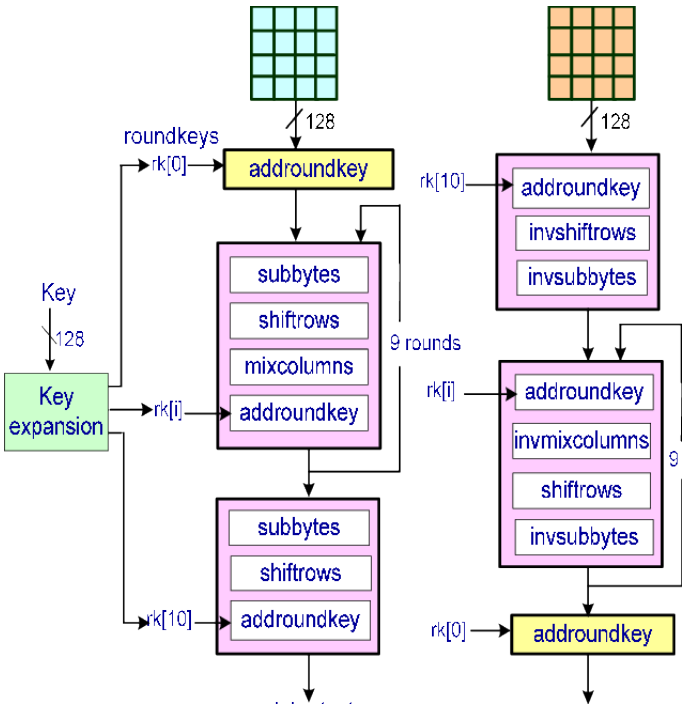


Fonte: (MATCRYPTO, 2021).

2.11 Funcionamento Geral do AES

Na Figura 2.12 são apresentadas todas as etapas combinadas que compõem o algoritmo AES. Porém, nota-se que o número de *rounds* irá variar de acordo com o tamanho da chave em questão.

Figura 2.12 – Funcionamento do AES.



Fonte: (MATCRYPTO, 2021).

2.12 O uso de séries temporais em problemas de predição

As séries temporais são utilizadas em aplicações de economia, previsões de mercados, entre outros. No caso do aprendizado de máquina, os vetores de entrada **X** do algoritmo de predição são séries univariadas ou multivariadas.

Uma série temporal com N observações é uma sequência de dados ordenados e equidistantes cronologicamente, sendo que normalmente compartilham uma característica (série invariada) ou várias características (série multivariada)

(NADA, a). A notação utilizada para representar uma série temporal univariada é dada por uma sequência $y_1, y_2, ..., y_n$, onde y_t é a observação para todo $1 < t < N$, sendo N o tamanho da série. As N observações $y_1, y_2, ..., y_n$ podem ser escritas na forma de um vetor coluna $[y_1, y_2, ..., y_n]'$ de ordem $N \times 1$.

A representação matemática para uma série temporal multivariada é dada por $y_1, y_2, ..., y_n$, onde $y_t = [y_{t1}, y_{t2}, ..., y_{tM}]'$ $(M \geq 2)$ e $1 < t < N$. Sendo assim, as N observações podem ser escritas em uma matriz Y de ordem $N \times M$, dada por:

$$\mathbf{Y} = \begin{bmatrix} y_1{}' \\ y_2{}' \\ . \\ . \\ . \\ y_n{}' \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \\ y_{21} & y_{22} & \cdots & y_{2M} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ y_{N1} & y_{N2} & \cdots & y_{NM} \end{bmatrix} \tag{2.5}$$

Segundo Maurício (1992), o objetivo principal de uma série temporal é elaborar um modelo estatístico, em que seja possível descrever o comportamento dos valores amostrados e, a partir de então, elaborar um modelo que tenha condições de descrever, prever e contrastar características acerca da série. Em níveis, o modelo pode apresentar divergências entre o real e o modelado. Dessa forma, o objetivo é minimizar o erro fazendo com que a série temporal atinja um nível de sofisticação muito próximo do real.

2.13 Aprendizado de máquina e inteligência artificial

O aprendizado de máquina diz respeito à ciência que busca simular processos biológicos em processos computacionais e, desta forma, aprender a partir de dados de treinamento (MICROSOFT, 2021). O aprendizado de máquina é muito útil para problemas em que não existam algoritmos conhecidos ou soluções. O algoritmo é capaz de solucionar problemas a partir de um conjunto de entrada. De

acordo com Maurício (1992), os algoritmos de aprendizado de máquina possuem muitas variedades, e por isso, foram classificados em segmentos segundo a sua estrutura de funcionamento:

- Podem ser do tipo supervisionado, não supervisionado, misto ou por aprendizagem por reforço, os quais remetem a capacidade do algoritmo de aprender sem a intervenção humana ou em partes;

- Quanto a velocidade de aprendizagem, de forma que se classificam como incremental ou por lotes;

- Podem ser baseados em modelos de predição dos dados ou simplesmente reconhecem padrões a partir de comparações com os dados de entrada.

2.14 Aprendizado supervisionado e não supervisionado

No aprendizado supervisionado, os dados são fornecidos na forma de classes. Por exemplo, em problemas de classificação, uma variável associada à saída **Y** recebe 1 se pertencer a classe de interesse e 0 caso contrário. Dessa forma, nesse tipo de classificação, o algoritmo aprende a partir dos exemplos. Já no aprendizado não supervisionado, os dados são fornecidos sem a atribuição de uma classificação prévia (0 ou 1), assim não há exemplos para aprendizagem e o algoritmo deve solucionar o problema de forma independente. No aprendizado misto (combinação de supervisionado e não supervisionado), o algoritmo lida com dados de treinamento com atribuição de classificações e outra porção sem atribuição dessa característica, obtendo-se uma observação parcial do problema em que uma parte é conhecida e a outra permanece desconhecida. Por fim, no algoritmo de aprendizado por reforço, a operação ocorre por meio de regras pré-especificadas a priori, de modo que essas regras definem qual será o comportamento do algoritmo ao lidar com determinada situação.

Existe também o critério de aprendizagem por lote, que refere-se ao fato de que todos os dados disponíveis são treinados de forma inicial e, em seguida, o algoritmo aplica todo o conhecimento adquirido em produção, sendo que os dados de treinamento são estáticos. Assim, qualquer demanda diferente dos dados apresentados anteriormente não reflete no sistema, ou seja, o algoritmo não aprende novas informações. Essa arquitetura de aprendizagem pode ser lenta (caso haja muitos parâmetros). Além disso, caso o sistema necessite de atualizações para um novo conjunto de dados, é necessário realizar o treinamento novamente utilizando não apenas os dados novos, mas também os dados antigos. Sendo assim, a aprendizagem por lotes não é indicada caso o conjunto de dados seja grande, necessite de constantes atualizações ou de uma resposta rápida frente à um novo parâmetro desconhecido.

O critério de aprendizagem incremental é o oposto do critério por lote, de modo que o algoritmo incremental particiona os dados em pequenas porções e assim, as treina em pequenas porções de maneira separada. Essa abordagem é útil quando se comparada ao esforço computacional despendido, pois ao realizar a tarefa em etapas permite o treinamento em computadores de baixo rendimento, quando se realiza à chamada memória externa de aprendizagem, ou do inglês, *out-of-core learning* (GéRON, 2007). A memória externa de aprendizagem é quando os dados de treinamento são grandes a ponto de não ser viável o carregamento em memória, ocasionando o *overflow* da memória, que é a impossibilidade temporária de uso da memória devido ao uso que exceda sua capacidade. Nesse sentido, a memória externa de aprendizagem é uma solução para esse problema elucidado.

Dessa forma, o algoritmo apenas carrega uma pequena parte do conjunto e processa os dados. Esse processo é repetido até que seja processado todo o conjunto de dados. A vantagem dessa abordagem é que o sistema apresenta uma reatividade intensa às anomalias externas. Com isso, o sistema deve se adaptar a essa transição intensa.

De acordo com essa abordagem, é possível definir uma taxa de aprendizagem, impactando na velocidade de aprendizagem do algoritmo. Porém, é possível observar que uma taxa alta de aprendizado não necessariamente define que o algoritmo será bom ou ruim. Nesse sentido, se uma taxa muito alta de aprendizado for aplicada, o algoritmo irá se adaptar melhor aos novos dados entrantes. No entanto, os dados antigos serão desconsiderados, o que pode representar um problema, uma vez que os dados passados são importantes para o delineamento dos dados futuros.

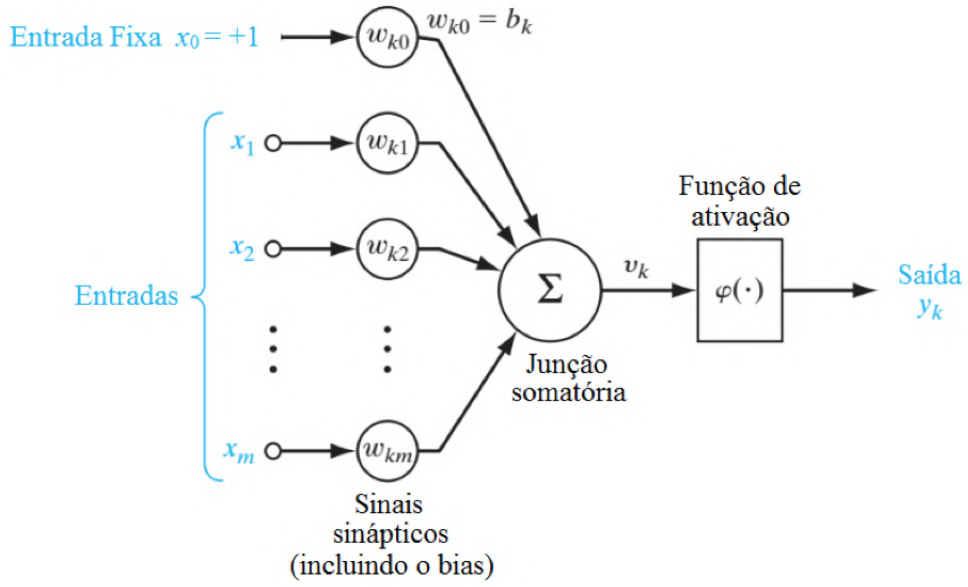
2.15 As redes neurais artificiais

As redes neurais foram inspiradas nos sistemas biológicos, tendo como base de inspiração os neurônios do córtex humano. Esse conceito de reprodução de atividades biológicas em âmbito computacional foi introduzido em 1943 por McCulloch e Pitts, em que foi apresentada a primeira arquitetura de RNA (Redes Neural Artificial) . Com o passar do tempo, as arquiteturas de RNA foram aprimoradas e aliadas ao avanço da capacidade computacional de processamento. Com a produção de unidades de processamento gráficos (GPUs) cada vez mais sofisticadas, foi possível executar redes mais robustas e com tempo de ação reduzido.

Durante o estudo, McCulloch e Pitts propuseram seu próprio neurônio, conhecido por sua concepção em que a saída é ativada segundo uma lógica booleana, a qual já era capaz de realizar na época operações complexas. Anos mais tarde, Frank Rosenblatt propôs uma arquitetura semelhante à sua antecessora. Nesse caso, associou-se expressões booleanas como uma energia de ativação, sendo que a partir de um valor de limiar do neurônio ocorre a ativação. Esse modelo ficou conhecido como Perceptron. A energia de ativação do neurônio é dada pela média ponderada das entradas, segundo um vetor de pesos W e R_n, seguido pela aplicação de uma função de ativação, em que normalmente para o Perceptron é o degrau. O classificador Perceptron é de natureza linear, sendo assim, os dados para esse tipo de classificação devem ser separados segundo uma reta para o melhor funciona-

mento do mesmo. A Figura 2.13 mostra o neurônio desenvolvido por McCulloch e Pitts.

Figura 2.13 – Neurônio artificial baseado no modelo de McCulloch e Pitts.



Fonte: (ESPACIOS, 2021).

2.16 Funções de ativação

O objetivo das funções de ativação é mapear a saída dentro de um intervalo fixo. As funções ativações nem sempre seguem esse intervalo, pois algumas delas apresentam intervalos que tendem para o infinito. Além disso, preferencialmente, as funções de ativação devem ser contínuas e diferenciáveis para evitar singularidades, sofismas, entre outros problemas de existência (Mauricio,1992).

- Função Sigmóide é uma curva em formato de S de domínio $x \in \mathbb{R} \setminus \{x_0\}$ e imagem limitada por K, sendo K um escalar e x_0 é o ponto médio da função sigmoide.

$$\sigma(x) = \frac{L}{1 - e^{-k(x-x_0)}} \tag{2.6}$$

- Função tangente hiperbólica é outra função comumente usada, com a vantagem de ser limitada entre -1 e 1.

- *ReLu* é outra função de ativação inspirada por problemas na área de biologia nos anos 60 que define 0 se o argumento é negativo e o próprio número caso seja positivo.

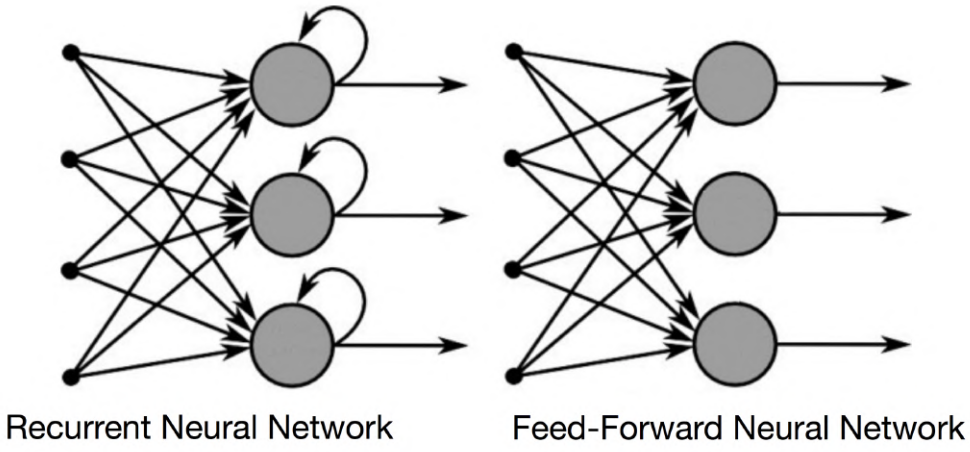
$$\text{ReLU(x)} = max\left\{0, x\right\} \tag{2.7}$$

2.16.1 Redes neurais recorrentes e não recorrentes

As redes neurais recorrentes têm como objetivo analisar séries temporais, e com isso, prever saídas no tempo. Essas redes têm como variável principal o tempo, ou seja, a saída em um tempo posterior está diretamente ligada à entrada do tempo passado. Essa classe de redes não se limita às entradas de tamanho fixo, sendo assim, entradas de diferentes tamanhos, como frases, áudios ou documentos, podem ser entradas dessas redes.

Um neurônio recorrente é aquele que conserva os seus estados, e a partir disso pode reconstituir uma saída considerando as saídas e entradas anteriores. Entretanto, essa abordagem apresenta alguns impasses que serão explicados posteriormente. Os neurônios recorrentes podem ser do tipo *backfoward* ou em português "para trás", ou seja, a direção de fluxo se dá em ambos sentidos de entrada/saída e saída/entrada. Diferentemente dos anteriores, neurônios que não apresentam recorrência atuam em apenas um sentido, sendo classificados como do tipo *feedforward* (GéRON, 2007). A Figura 2.14 mostra a estrutura dessas duas relações apresentadas.

Figura 2.14 – Estrutura tipica de redes neurais recorrentes e convencionais.



Fonte: (DEEPLEARNBOOK, 2021).

2.16.2 Long Short-Term Memory (LSTM)

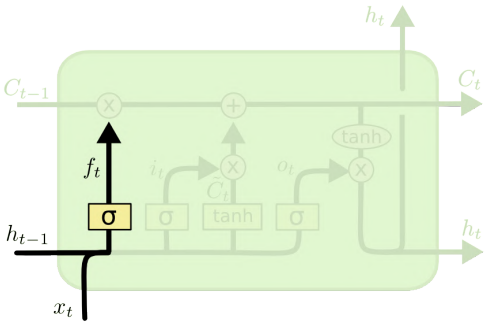
Devido a natureza de recorrência das RNN (Redes Neurais Recorrentes), é necessário um grande esforço computacional para que as informações anteriores sejam consideradas. Um problema comum das RNN é o *vanish/exploding gradient*, operações matemáticas com valores muito pequenos ou muito elevados. Respectivamente, eles resultam em operações com capacidade inviáveis de processamento. Sendo assim, a *Long Short-Term Memory* (LSTM) é uma rede neural que tem por objetivo solucionar o problema elucidado, de modo que o principal objetivo dessa rede é evidenciar as informações que são necessárias para a modelagem do problema (GéRON, 2007).

A LSTM foi idealizada por Hochreiter e Schmidhuber em 1997, com a função de solucionar problemas de longo prazo e, com isso, tornar possível para implementação computacional. A LSTM decide quais informações são relevantes de acordo com seus *gates* (saída após a aplicação de uma função sigmóide), que

tem a função de controlar o fluxo de informações relevantes do passado. Dessa forma, existem três tipos de *gates* na implementação da LSTM.

O *Forget Gate* tem a função de selecionar quais informações são relevantes para seguir em frente. Primeiramente, o estado anterior é multiplicado por um peso W_f de dimensão adequada e, em seguida, é aplicada uma função sigmóide. A função sigmóide é representada por $Img(\sigma) \in [0,1]$, sendo assim, o valor 0 significa que a informação será "totalmente desnecessária" e o valor 1 que será "totalmente relevante".(Colah Github, 2015) , de acordo com a Figura 2.15.

Figura 2.15 – Etapa do Forget Gate.

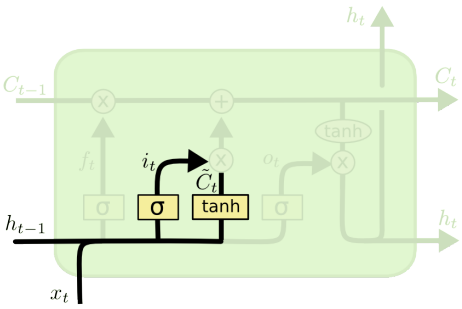


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fonte: (Colah Github, 2015).

O *Remember Gate* diz respeito à memória de curto prazo que será inserida na próxima célula de estado e pode ser utilizada para inserir informações ou atualizá-las. A operação ocorre pela concatenação de $[h_{t-1}, x_t]$, seguido pela multiplicação de um vetor de peso W_i com aplicação na função sigmóide. O resultado é multiplicado por uma célula $\tilde{C}_t$ dada pela aplicação de uma função de ativação tangente hiperbólica em um peso W_C (Colah Github, 2015). A Figura 2.16 mostra o funcionamento da etapa descrita.

Figura 2.16 – Etapa do Remember Gate.

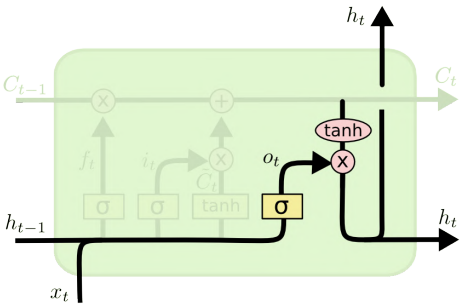


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Fonte: (Colah Github, 2015).

O *Output gate* está relacionado com as informações do momento atual e que são repassadas à célula de estado, sendo diretamente influenciada pelos *gates* anteriores. A próxima célula de estado é dada pela aplicação de uma função tangente hiperbólica, conforme demonstrada pela Figura 2.17 (Colah Github, 2015).

Figura 2.17 – Etapa do Output Gate.



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

Fonte: (Colah Github, 2015).

2.17 Formas de estimação de perfomance

Para concluir se a RNN se adequou corretamente aos objetivos propostos, devem ser realizadas estimações avaliativas dos resultados obtidos. Nesse sentido, as avaliações propostas para a análise de desempenho tem como foco o erro médio absoluto e relativo. De acordo com Heizer (2004), o erro absoluto considera

apenas a natureza dos valores absolutos, de forma que a soma de valores absolutos geram grandezas sempre positivas, podendo gerar um erro absoluto com valores elevados.

$$\varepsilon_a = \sum_{k=1}^{N} \frac{|y_k - \hat{y_k}|}{N} \tag{2.8}$$

Quanto ao erro absoluto quadrático, os valores elevados à potência tendem a diminuir o efeito causado no erro absoluto. Dessa forma, os valores são atenuados caso o erro se torne uma grandeza com alto valor (HEIZER, 2004).

$$\varepsilon_q = \sum_{k=1}^{N} \frac{|y_k - \hat{y_k}|^2}{N} \tag{2.9}$$

Nas Equações 2.8 e 2.9 , y_k e $\hat{y_k}$ não necessariamente devem ser escalares. As saídas podem ser vetores $y_k, \hat{y_k} \in \mathbb{R}^d$ (em caso de múltiplas saídas), sendo d o número de saídas. Nas respectivas equações, y_k representa o k-ésimo valor $\hat{y_k}$ obtido e o k-ésimo valor predito pelo algoritmo.

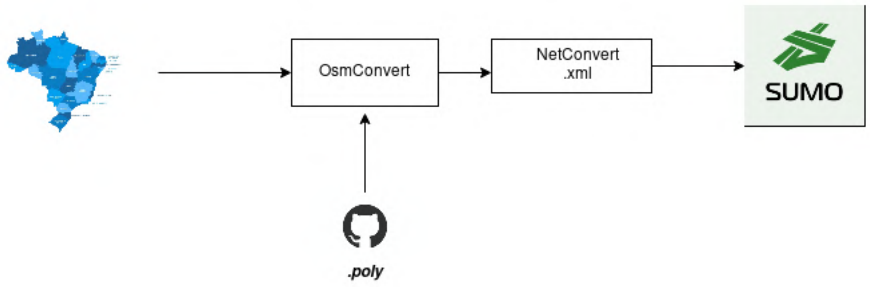
3 METODOLOGIA

Nesse capítulo são descritos os passos para o desenvolvimento do *framework*, incluindo o detalhamento da solução para a utilização de uma rede neural a partir de variáveis de interesse (aceleração, latitude e longitude), a fim de fomentar estudos posteriores.

3.1 Arquitetura do funcionamento do framework

A Figura 3.1 ilustra o funcionamento do *framework* proposta na forma de fluxograma. A partir da utilização do mapa do Brasil, o *framework* OSMCONVERT realiza a extração do mapa da região de interesse. Em seguida, com o auxílio de ferramentas disponíveis no SUMO, é gerada a NETWORK do mapa, que será responsável por inicializar o SUMO. Dessa forma, uma vez que o arquivo .NET.XML é carregado no SUMO, as variáveis de interesse são simuladas e seus resultados são exportados no formato XML (*Extensible Markup Language*).

Figura 3.1 – Diagrama de funcionamento do *framework* criado.

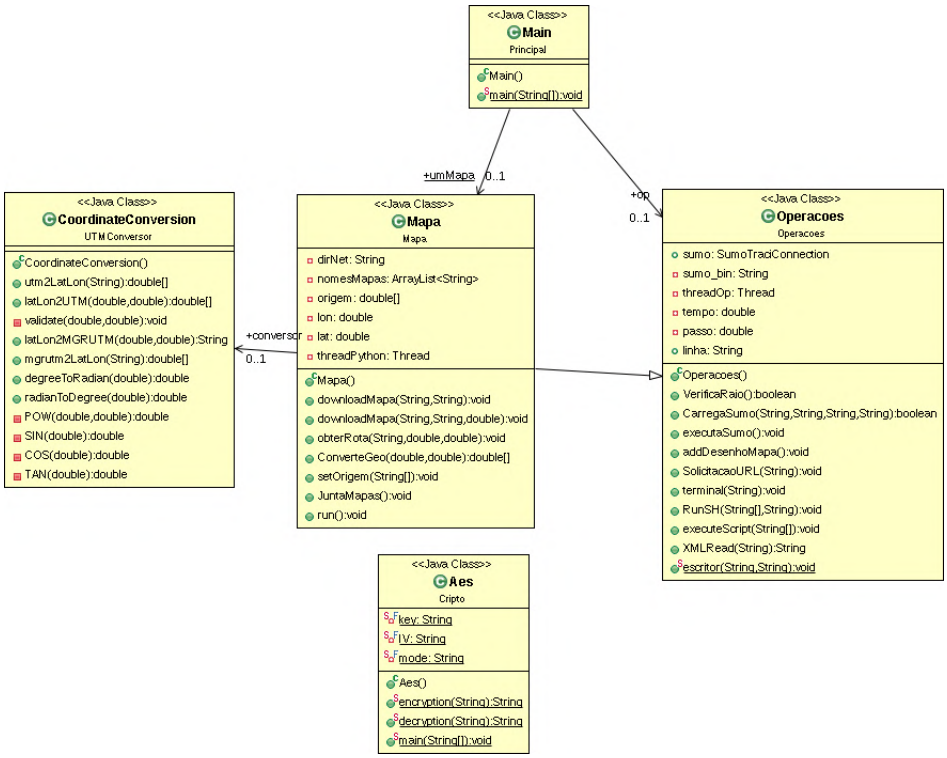


Fonte: Do autor (2021).

3.2 Descrição da hierarquia de classes do framework

Para a compreensão dos desenvolvimentos realizados em prol do *framework* proposto, um diagrama de classes é apresentado na Figura 3.2.

Figura 3.2 – Diagrama de classes UML do *framework* proposto.



Fonte: Do autor (2021).

O *framework* desenvolvido tem como objetivo auxiliar na geração de rotas para problemas de logística. Para isso, as ferramentas utilizadas foram: o simulador SUMO e mapas digitais extraídos do *OpenStreetMap*.

Na classe Operações são realizadas operações matemáticas, conversão entre sistemas de coordenadas com auxílio da biblioteca IBM Geo da *International Business Machines Corporation* (IBM), além do carregamento e execução do SUMO. A classe possui suporte para requisições *Hypertext Transfer Protocol* (HTTP) e *Hypertext Transfer Protocol Secure* (HTTPS), sendo a diferença entre a duas que esta ultima é criptografada. Desta forma, é possível também a comunicação por meio de API externa e a integração com sistemas operacionais com método próprio, mediante suporte a *Shell scripts* (arquivo .sh).

A classe Mapa permite a digitalização do arquivo .OSM. Os atributos da classe possuem, dentre outras informações, o nome do mapa de interesse (no caso a cidade), informações de latitude e longitude de cada rua disponível no mapa, informações do tamanho da área do mapa, dentre outros.

A classe Mapa possui suporte para a realização de *downloads* de forma sequencial ou paralela (análogo ao *pipeline*), de acordo com o critério do usuário. Ela também permite realizar interações com mapas do *OpenStreetMap*. Um dos seus métodos, permite a extração de mapas menores a partir de um mapa maior. Isto é, a partir de um mapa base é possível extrair apenas mapas de interesse do usuário, tal como um bairro, uma cidade ou um estado.

Tendo em vista que há disponibilidade do mapa do Brasil para *download* no *OpenStreetMap* em um arquivo único, o mesmo foi instalado em sistema embarcado, permitindo a redução de processos de leituras em bases remotas e, consequentemente, economizando no tráfego de dados. Com o mapa do Brasil instalado é possível extrair ruas, cidades, distritos e estradas rurais.

A classe Veículo é responsável por fazer a interligação com recursos externos ao *framework*, em especial, com a porta OBD2 de veículos comuns, onde os dados são coletados e atualizados no simulador. O retorno dos dados é no formato *JavaScript Object Notation* (JSON). Todos os dados são criptografados com o algoritmo AES-128 bits com modo *Cipher Block Chaining* (CBC) .

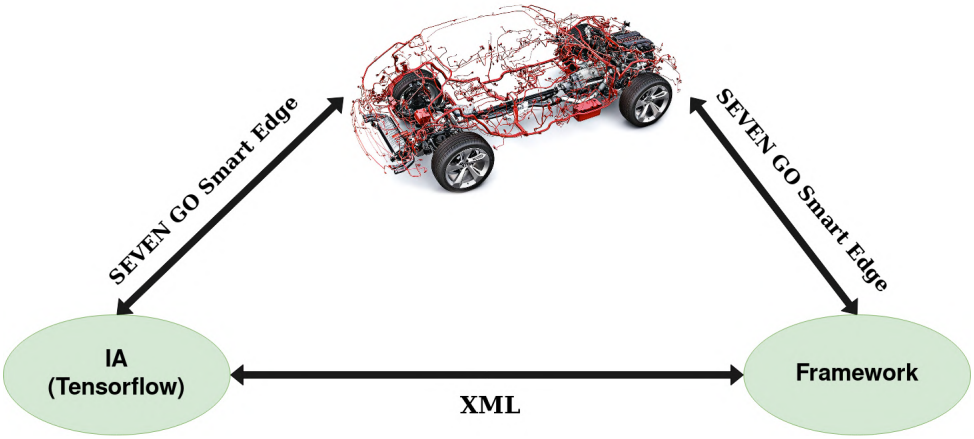
A empresa SEVEN GO dispõe de uma tecnologia para conexão remota de veículos, SEVEN GO Smart Edge. O *framework* proposto pode se integrar em tempo real com a solução. A taxa de atualização de dados entre as soluções é de 10 segundos. Esse intervalo foi definido com o objetivo de permitir a execução em sistemas embarcados não dedicados ao *framework*, isto é, onde existem outras aplicações executadas concorrentemente.

Finalmente, a classe *Main* controla as interações entre as classes, realiza as instâncias dos objetos, direciona os *paths* para o *framework* e arquivos de configuração, bem como recebe os parâmetros de entrada da aplicação.

3.3 Arquitetura da integração entre os serviços

Na Figura 3.3 é apresentado o diagrama de relação entre os serviços implementados para fins de exemplificação. Observa-se que o SEVEN GO Smart Edge realiza a alimentação dos dados para as duas plataformas desenvolvidas. Por conseguinte, as duas plataformas comunicam-se entre si através de arquivos comuns, tal como o XML.

Figura 3.3 – Diagrama da relação entre os serviços implementados.



Fonte: Do autor (2021).

3.4 Aprendizado de máquina: TensorFlow

O TensorFlow é uma biblioteca desenvolvida para o aprendizado de máquina. A versão Lite é a mais indicada para sistemas embarcados. A linguagem utilizada para o desenvolvimento é o Python, juntamente com outras bibliotecas nativas da linguagem. Foram utilizadas as bibliotecas *Scikit − Learn* e *NumPy*.

3.4.1 Pré-processamento dos dados

Para que os dados sejam úteis e passíveis de serem utilizados, é necessário realizar um pré-processamento com o objetivo de retirar valores nulos e desnecessários. Para isso, utilizou-se o método $to\_datetime()$ do Pandas. Em seguida, a normalização é realizada utilizando o método do *TensorFlow*, denominado $MinMaxScaler()$. O método citado realiza uma normalização dada pela seguinte equação:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

3.4.2 Construção da rede neural recorrente

Utilizando o Jupyter Notebook, um modelo de predição é criado, de modo que todas as variáveis preditas utilizam as mesmas configurações. Assim, os dados transformados foram combinados de tal forma que 90% dos dados totais são de treinamento e os outros 10% representam o conjunto de teste. Ademais, foram definidos empiricamente 100 neurônios na camada de entrada, com taxa de aprendizado de $0,01$ e limitação do número de iterações em 4000.

4 RESULTADOS E DISCUSSÕES

4.1 Desempenho computacional do *framework* proposto

O SUMO é um simulador que no contexto da proposta caso atua como interface entre os dados do veículo e o *framework* e, a partir deste ponto, obtém estimativas futuras das variáveis do veículo.

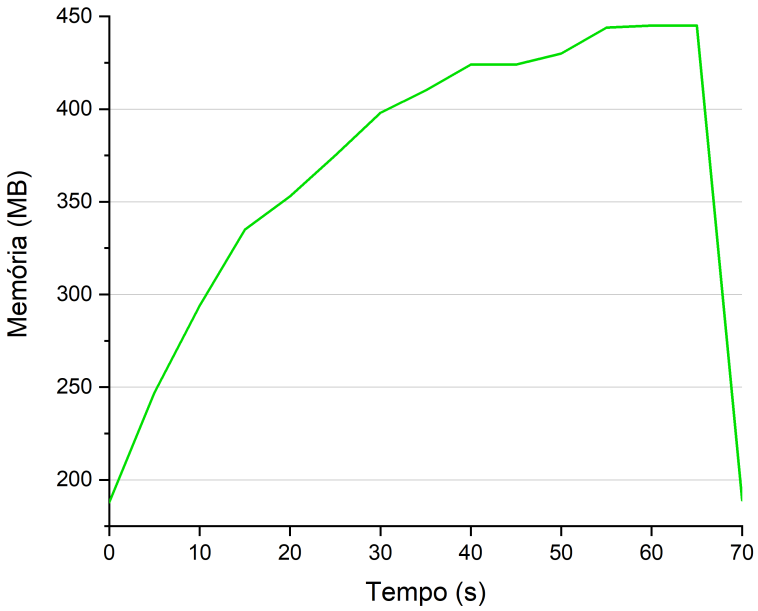
Os testes a seguir exploram questões relacionadas aos requisitos mínimos de execução do *framework* para sistemas embarcados. Testes de memória e de processamento foram realizados utilizando um Raspberry Pi 3 que possui 1 GB de memória com um processador *Cortex* A-53. Quanto ao Orange Pi Zero, o mesmo é constituído de um processador *ARM Cortex A7 Quad Core* com 1 GB de memória RAM. Ambos são microcomputadores comerciais controlados por um microcontrolador, que permite a integração de sensores e outros dispositivos. Ambos os microcomputadores mencionados possuem 4 núcleos que permitem realizar 4 *threads* simultâneas. Dessa forma, os dispositivos utilizados são apresentados na Figura 4,1.

Figura 4.1 – O *hardware* utilizado.

Fonte: AMAZON (2021).

Nas Figuras 4.2 a 4.4 são apresentados os gráficos relativos ao uso de memória:

Figura 4.2 – Medida do uso de memória do Raspberry Pi 3.



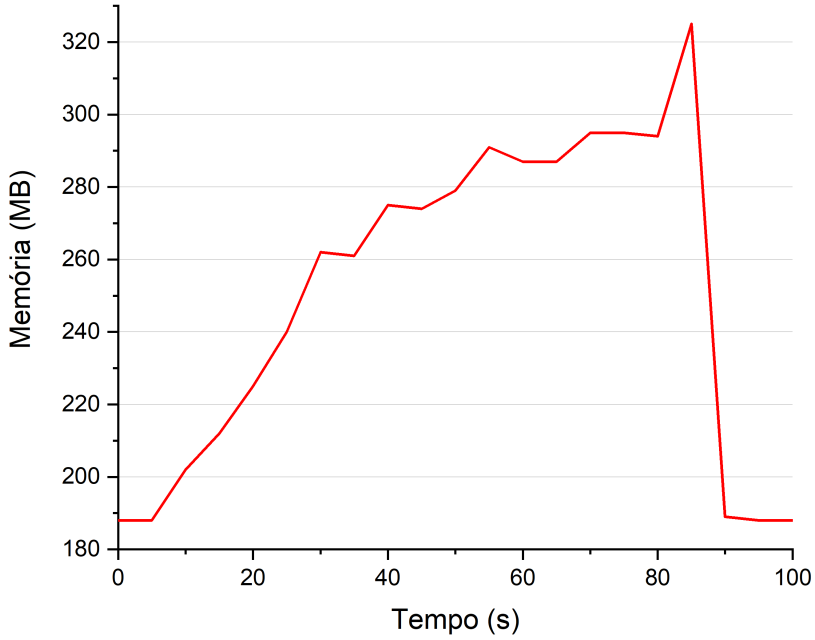
Fonte: Do autor (2021).

Conforme observado na Figura 4.1, o *framework* apresenta um pico de memória ao final do processamento, com aproximadamente 450 *MB* de paginação máxima. O Raspberry Pi 3 possui um total de 926 *MB* de memória total disponível. É possível observar que para esse dispositivo não foi necessário o uso da memória de *SWAP*. O *SWAP* é uma área de troca usada para aumentar a quantidade de memória RAM do sistema, escrevendo as informações no disco físico do dispositivo. No entanto, devido ao disco físico ser centenas de vezes mais lento que a memória RAM, a paginação se torna lenta afetando a performance de execução dos programas.

Na mesma perspectiva de análise, foram coletados os dados do desempenho de memoria utilizando o Orange Pi Zero H2, a qual este hardware possui 512 *MB* de memória RAM e 4 núcleos de processamento *Quad Core*. Para esse dis-

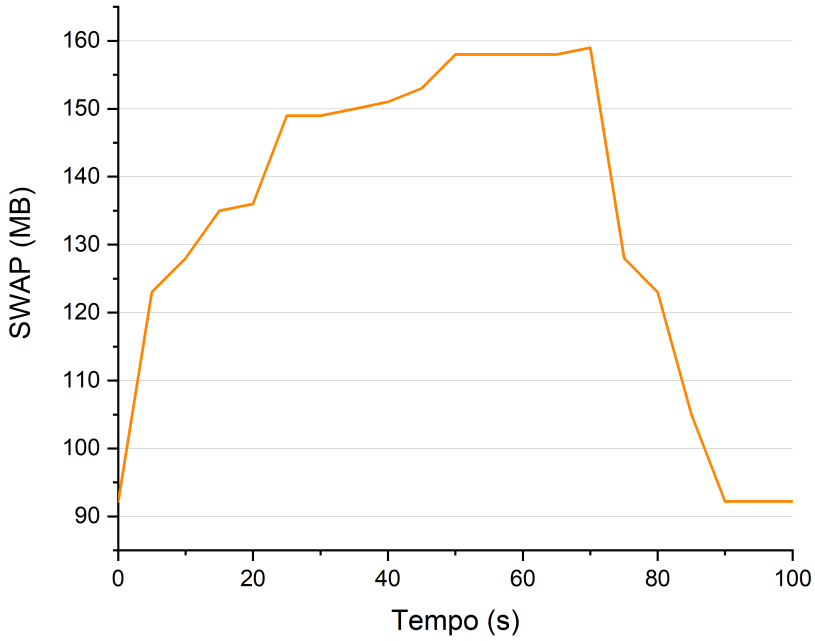
positivo, houve a necessidade do uso da memória *SWAP*. Com isso, os dados são apresentados na Figura 4.3 e 4.4.

Figura 4.3 – Medida do uso de memória do Orange Pi Zero H2.



Fonte: Do autor(2021).

Figura 4.4 – Medida do uso de memória Swap no Raspberry Pi.



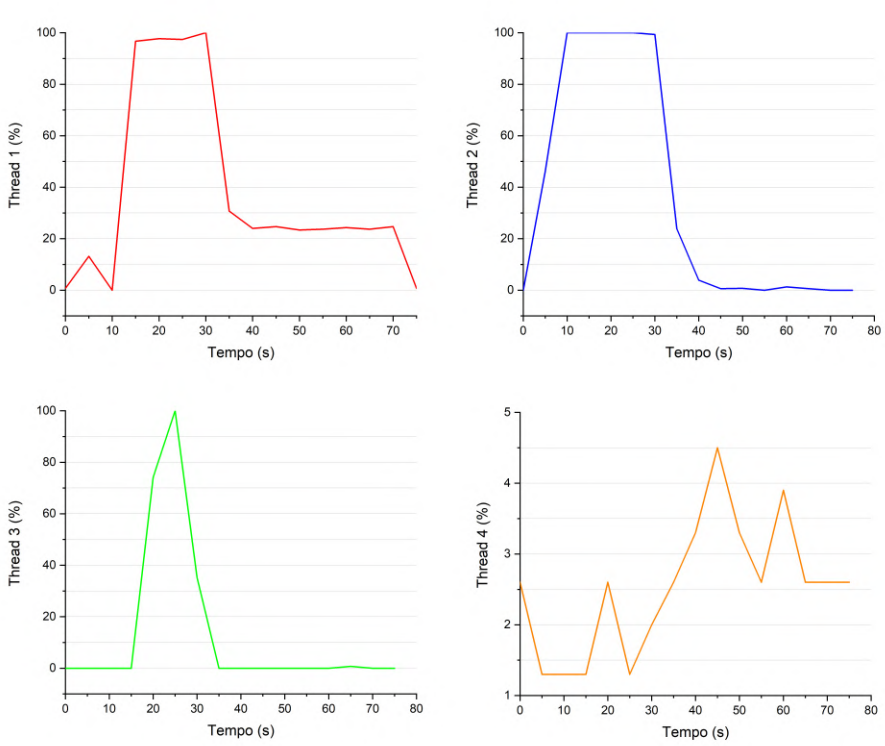
Fonte: Do autor(2021).

De acordo com as Figuras 4.2 e 4.3, observa-se que o uso de memória no Orange PI H2 Zero foi perceptivelmente maior quando comparado ao uso do Raspberry Pi 3, uma vez que o *SWAP* é somente utilizado quando é esgotada toda a memória principal. Todavia, a baixa paginação na memória principal foi motivada pelo uso da memória *SWAP*, que é mais lenta que a memória RAM. Dessa forma, o tempo de processamento total do *framework* é afetada, gerando um acréscimo de 30 segundos quando comparado ao Raspberry Pi 3.

Por fim, foram coletados os dados de processamento para efeitos de validação de cada uma das 4 *threads*. As threads estão relacionadas com o núcleo do processador, apresentando a carga de uso do processador separadas por *thread*. As *threads* nesse caso funcionam como se fossem um processador independente, e realizam a divisão das tarefas entre sí, o que permite um ambiente multitarefas. Os

dados dessas *threads* foram coletados utilizando o HTOP (programa visualizador de processos) do Linux em um intervalo de 5 em 5 segundos, conforme a Figura 4.5.

Figura 4.5 – Medida do uso do processador no Raspberry Pi 3.



Fonte: Do autor (2021).

A Figura 4.5 apresenta a carga de uso do processador do Raspberry Pi 3 em cada umas 4 *threads* do dispositivo. É possível observar que os núcleos do dispositivo não operam na capacidade máxima. Além disso, os gráficos apresentados são referentes a todos os processos do dispositivo (Sistema Operacional e demais aplicações instaladas). Ademais, os testes anteriores consideram os tempos de carregamento do arquivo .NET.XML, que é o maior entre todos os arquivos necessários para a inicialização do SUMO. Desse modo, infere-se que os dois dispositivos apresentados são elegíveis para a execução do *framework*, uma vez que não houve a saturação dos núcleos.

4.2 Desempenho do framework proposto para a geração de rotas

A seguir, são apresentados os resultados de estimativas de rotas comparando as rotas obtidas no *Google Maps* e no *OpenStreeMap*. O resultado obtido pelo *framework* proposto é apresentado na rota traçada entre os dois pontos no mapa inferior na Figura 4.6.

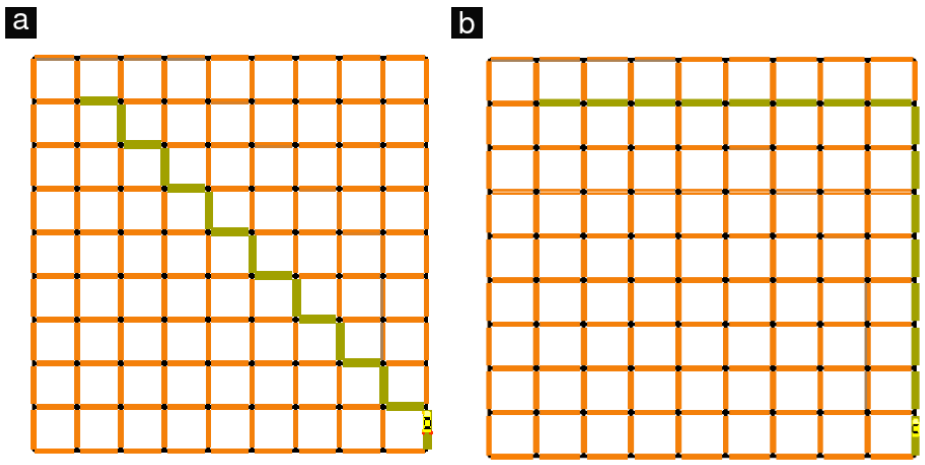
Figura 4.6 – Ilustração das rotas traçadas entre as plataformas: (a) *GoogleMaps*, (b) *OpenStreetMap* e (c) SUMO.

Fonte: Do autor (2021).

4.3 Geração de rotas no framework proposto

Conforme discutido nas seções anteriores, o *framework* é capaz de traçar rotas otimizadas através de simulações. Assim, dada a origem e o destino para a rota, o *framework* oferece como saída as rotas mais curta e mais rápida. Na Figura 4.7, é mostrado o resultado obtido pelo simulador ao estabelecer a rota mais curta e a mais rápida. Para o caso da rota mais rápida, o simulador é capaz de realizar a otimização segundo a menor distância ou com respeito ao menor tempo de viagem.

Figura 4.7 – Resultados produzidos pelo *framework*:(a) Rota mais curta, (b) Rota mais rápida.



Fonte: Do autor (2021).

4.4 Desempenho do processamento de rotas

Com o carregamento da NETWORK do mapa, o tempo para geração de rotas se torna ínfimo, sendo que a medida que as rotas tornam-se extensas, o tamanho do arquivo aumenta de forma considerável, o que gera um desafio para o processo de simulação. A Tabela 4.1 mostra os resultados obtidos a partir de rotas geradas.

Tabela 4.1 – Resultados obtidos do processamento das rotas.

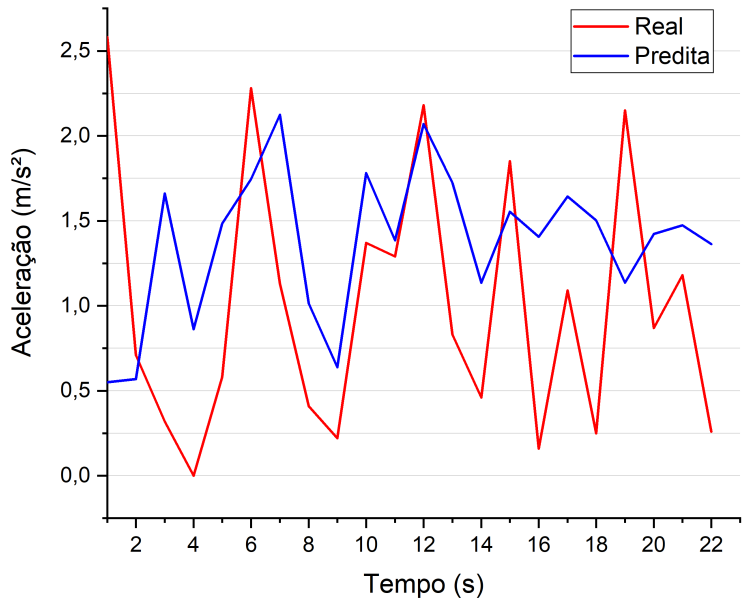
Rota	Distancia Percorrida [m]	Tempo execução [ms] FindRoute()
1	815.28	42
2	126.45	5
3	180.32	6
4	569.56	26
5	682.27	32

Fonte: Do autor (2021).

4.5 Resultados obtidos com o *TensorFlow*

Com o auxílio do *TensorFlow*, uma rede neural foi projetada para a predição das variáveis de interesse (aceleração, latitude e longitude). O primeiro teste consistiu em gerar valores de aceleração a partir de uma rota gerada pelo *Python* e simulada no SUMO. Os resultados são apresentados na Figura 4.8.

Figura 4.8 – Aceleração predita pela rede neural.



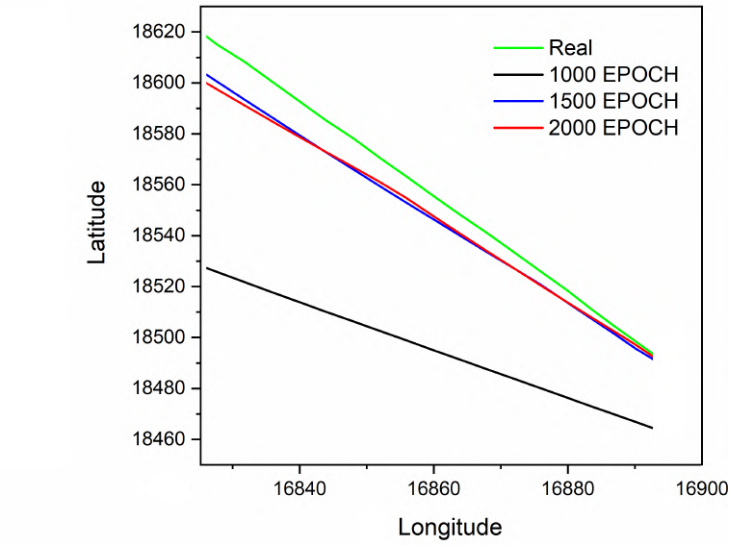
Fonte: Do autor (2021).

A Figura 4.8 mostra que a rede neural conseguiu se adaptar às tendências da aceleração (se existe tendência de crescimento da função, a saída da rede neural acompanhou a mesma tendência) em alguns intervalos. Entretanto, ao analisar os valores do conjunto de teste e a saída desejada, observou-se que a predição apresentou acentuada discrepância.

As coordenadas $< x, y >$ do SUMO consistem no módulo da diferença da coordenada UTM subtraído pelo *offset* do mapa. O *offset* do mapa é definido como a origem do mapa e também é representado no formato UTM.

O último teste consistiu em treinar a rede neural correlacionando as variáveis em um único vetor de entradas **X**. Nos ensaios seguintes, as variáveis de latitude e longitude foram treinadas com correlação entre si. Para fins de comparação, foram definidos outros parâmetros diferentes dos anteriores. A configuração consiste em 100 neurônios de entrada, com variação do número de épocas para observação da adaptação da rede ao conjunto de treinamento. Ademais, o *batch size* (treino em lote) foi alterado para 32, diferentemente da configuração anterior que era sequencial. Na Figura 4.9, são apresentados os resultados obtidos.

Figura 4.9 – Resultados obtidos da rede neural com dados correlacionados.



Fonte: Do autor (2021).

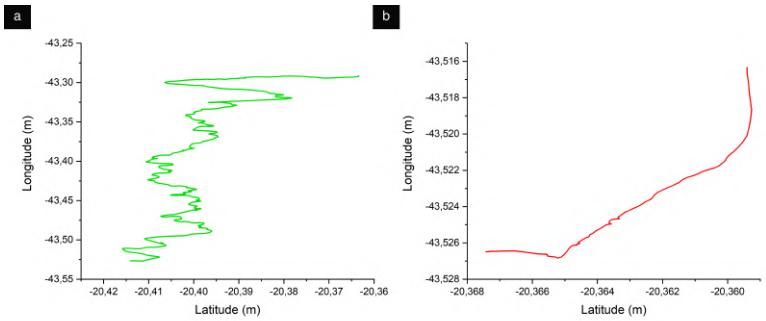
A Figura 4.9 apresenta o conjunto de teste submetido à rede neural e também os valores preditos seguindo o número de 1000, 1500 e 2000 épocas. A medida em que o número de épocas aumenta, os resultados divergem dos valores esperados. Um esforço computacional maior não melhorou os resultados da rede. Os parâmetros definidos para essa rede também foram determinados empiricamente. Para fins de simplificação, as coordenadas do $SUMO < x, y >$ foram nomeadas como latitude e longitude.

Pelo caráter linear apresentado na Figura 4.9, nota-se que a rede foi capaz de associar a origem e o destino para as épocas 1500 e 2000, uma vez que as retas convergem para o mesmo ponto na interseção. Entretanto, para o caso em que 1000 épocas foram treinadas, essa associação de origem e destino não foi obtida pela rede.

4.6 Predição de rotas utilizando os dados de um veículo real

Utilizando os dados coletados de um veículo real, obtidos a partir do SEVEN GO Smart Edge, foi realizada novamente a execução do algoritmo de predição LSTM. Entretanto, dessa vez o *batch size* foi alterado para 1 ao invés de variar o número de épocas e, com isso, observa-se que para esse conjunto de dados a adaptação foi mais harmônica. Ao utilizar redes neurais, as configurações variam de acordo com a disposição dos dados. Nesse sentido, ao contrário da seção anterior que utilizou dados gerados pelo simulador, os dados produzidos por sensores reais são de certa maneira diferentes, e por esta razão, a rede neural foi alterada. A seguir, utilizando um conjunto de dados de 38962 elementos, foram dispostos 35066 elementos de dados para treinamento e 3896 elementos para predição dos dados. A Figura 4.10 apresenta a rota real e a rota predita pela rede neural mencionada, utilizando as coordenadas geográficas convencionais de latitude e longitude.

Figura 4.10 – Predição de rotas utilizando o LSTM.



Fonte: Do autor (2021).

Na Figura 4.10, à partir de uma rota de um veículo, obtidos a partir do SEVEN GO Smart Edge, foi utilizado o algoritmo LSTM para a predição de rotas. A vantagem dessas redes neurais profundas é que em poucos passos a rede se adapta à estrutura dos dados, que nesse caso foram de 4 passos. A ordem do erro quadrático médio do algoritmo foi de 10^{-8}, mas como coordenadas geográficas necessitam de pelo menos 7 casas decimais de precisão para uma boa acurácia, a rota predita não seguiu a forma geométrica da rota real. Desta forma, esse erro obtido deve ser ainda menor, para que a rota predita tenha a mesma forma geométrica e precisão da rota real. Entretanto, ao aumentar o número de passos a rede tende a melhorar a sua acurácia, evidentemente com o custo de um longo tempo de treinamento computacional.

5 CONCLUSÃO

O presente trabalho apresentou resultados preliminares de um *framework* de simulação de rotas. Um processo automatizado foi criado para permitir a geração de rotas tendo como entrada as coordenadas geográficas de origem e destino. As principais ferramentas utilizadas na construção foram: JAVA, SUMO e *OpenStreetMap*.

As rotas obtidas pelo *framework* proposto podem ser classificadas após o processo de simulação em mais curta, rápida e econômica. Os resultados podem ser exportados para o formato XML e integrados por aplicações de terceiros.

Testes foram apresentados para verificar a viabilidade da utilização do *framework* proposto em sistemas embarcados. Para isso se utilizou o HTop para a coleta dos dados de processamento e de uso de memória dos dispositivos selecionados para os testes. Em seguida, para que se pudesse verificar o funcionamento do *framework* em conjunto com uma rede neural, o *TensorFlow* foi utilizado para a implementação de um modelo de rede neural recorrente. Os parâmetros da rede neural foram definidos empiricamente, sem a intenção de se obter o melhor resultado.

Conclui-se que os estudos preliminares empenhados no desenvolvimento do *framework* proposto podem fomentar a criação futura de uma série de novos recursos e cenários nos mercados de interesse da empresa SEVEN GO. A arquitetura proposta permite a sua utilização em sistemas embarcados de forma otimizada e adaptativa, o que é uma vantagem, em conjunto com aplicações concorrentes em computadores de baixo rendimento e baixo custo, além da compatibilidade com aplicações de terceiros.

REFERÊNCIAS

APRH. **Revista de gestão costeira integrada**. 2007. Disponível em: <https://www.aprh.pt/rgci/glossario/WGS84.html>. Acesso em: 12/06/2021

BENNETT, J. **OpenStreetMap**. Packt Pub., 2010. ISBN 9781847197511. Disponívelem:<https://books.google.com.br/books?id=SZfqRcPXApoC>. Acesso em: 23/08/2021.

BIBLITA. **Análise DO ALGORITMO VENCEDOR DO AES: O RIJNDAEL**. 2021. Disponível em:<http://www.bibl.ita.br/ixencita/artigos/FundRafaelAntonio1.pdf>.Acesso em: 23/07/2021.

CNT. Custo logístico consome 12,7% do PIB do Brasil.Disponível em:<https://www.cnt.org.br/agencia-cnt/custo-logistico-consome-12-do-pib-do-brasil>. Acesso em: 13/11/2021.

DEEPLEARNBOOK. **Redes Neurais Recorrentes**. 2021. Disponível em: <https://www.deeplearningbook.com.br/redes-neurais-recorrentes/>. Acesso em: 06/09/2021.

EDISCIPLINAS. **Sistema de projeção UTM**. 2015. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/1738554/mod_resource/content/1/PTR0101%20-%20Proje%C3%A7%C3%A3o%20UTM%20v2015.pdf>. Acesso em: 15/08/2021.

ESPACIOS. **Revista Espacios**. 2021. Disponível em:<https://www.revistaespacios.com/a17v38n34/31-02.png>. Acesso em : 17/09/2021.

FATECLOG. **Análise DE TRÁFEGO NA PN DE RIO GRANDE DA SERRA: UM ESTUDO DE SIMULAÇÃO COM O SOFTWARE SUMO (SIMULATOR OF URBAN MOBILITY)**. 2019. Disponível em: <https://fateclog.com.br/anais/2019/AN%C3%81LISE%20DE%20TR%C3%81FEGO%20NA%20PN%20DE%20RIO%20GRANDE%20DA%20SERRA%20UM%20ESTUDO%20DE%20SIMULA%C3%87%C3%83O%20COM%20O%20SOFTWARE%20SUMO%20(SIMULATOR%20OF%20URBAN%20MOBILITY)..pdf>. Acesso em : 12/11/2021.

G1. **Gasolina nas alturas: até quando o preço do combustível vai subir?** 2021. Disponível em:<https://g1.globo.com/economia/noticia/2021/10/25/gasolina-nas-alturas.ghtml>.Acesso em: 17/11/2021.

GEO, M. **Coordenadas topográficas X Coordenadas UTM**. 2013. Disponível em:<https://mundogeo.com/2013/06/05/coordenadas-topograficas-x-coordenadas-utm/>. Acesso em : 26/10/2021.

GRUPOQUALITY. **Curso JAVA**. 2021. Disponível em:<https://grupoqualityambiental.com.br/2021/08/27/ecodriving-o-que-e-essa-modalidade-sustentavel-de-conducao-de-carros/>. Acesso em: 18/07/2021

GERON, A. **Mãos a obra: Aprendizado de Máquina com Scikit-Learn e TensorFlow**. Alta Books, 2007. ISBN 9788550809021. Disponível em: <https://books.google.com.br/books?id=Z0mvDwAAQBAJ>.

HEIZER, J. **Operations management**. Upper Saddle River, N.J: Pearson Prentice Hall, 2004. ISBN 978-0-13-120974-9.

MATCRYPTO. **AES Advanced Standard Encryption**. 2021. Disponível em: <https://commons.wikimedia.org/wiki/File>.Acesso em 19/11/2021.

MAURICIO, J.A. **Análisis de Series Temporales**. [S.l.]: Universidad Complutense de Madrid, 1992.

MICROSOFT. **Aprendizado de máquina no HDInsight**. 2021. Disponível em:<https://docs.microsoft.com/pt-br/azure/hdinsight/hdinsight-machine-learning-overview>.Acesso em : 23/09/2021.

OTEMPO. **Combustível em alta no Brasil impacta fretes e deliveries**. 2021. Disponível em:<https://www.otempo.com.br/economia/combustivel-em-alta-no-brasil-impacta-fretes-e-deliveries-1.2450066>.Acesso em 13/11/2021

POSADA, F.; FAÇANHA, C. Brazil passenger vehicle market statistics: International comparative assessment of technology adoption and energy consumption. 2015.

RENAULT. **DRIVING ECO AJUDA A REDUZIR EM ATÉ 20% O CONSUMO DE COMBUSTÍVEL**. 2021. Disponível em:<https://imprensa.renault.com.br/release/item/driving-eco-ajuda-a-reduzir-em-ate-20-o-consumo-de-combustivel/pt>. Acesso em: 21/11/2021

Colah Github. **Understanding LSTM Networks**. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.Acesso em: 24/10/2021.

UFRJ. **Algoritmo de criptografia AES**. 2005. Disponível em:<https://www.gta.ufrj.br/grad/05_2/aes/>.Acesso em: 27/07/2021.

ZARKADOULA, M.; ZOIDIS, G.; TRITOPOULOU, E. Training urban bus drivers to promote smart driving: A note on a greek eco-driving pilot program. **Transportation Research Part D: Transport and Environment**, Elsevier, v.