Universidade Federal
do Rio de Janeiro
Escola Politécnica

IDENTIFICAÇÃO DE OBSTÁCULOS EM VIAS PAVIMENTADAS POR MEIO DA AQUISIÇÃO DE DADOS VIA APLICATIVO ANDROID COM USO DE ALGORITMO DE APRENDIZADO DE MÁQUINA

Lucas Cavalcanti Adorno

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Flávio Luís de Mello

Rio de Janeiro

Junho de 2018

IDENTIFICAÇÃO DE OBSTÁCULOS EM VIAS PAVIMENTADAS POR MEIO DA AQUISIÇÃO DE DADOS VIA APLICATIVO ANDROID COM USO DE ALGORITMO DE APRENDIZADO DE MÁQUINA

Flávio Luís de Mello

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

_____

Lucas Cavalcanti Adorno

Orientador:

_____

Flávio Luís de Mello, D. Sc.

Examinador:

_____

Luiz Coutinho Ferreira da Silva, DE

Examinador:

_____

Laudelino Amaral de Oliveira Lima

Rio de Janeiro – RJ, Brasil

Junho de 2018

Declaração de Autoria e de Direitos

Eu, *Lucas Cavalcanti Adorno*, CPF *147294527-12*, autor da monografia *IDENTIFICAÇÃO DE OBSTÁCULOS EM VIAS PAVIMENTADAS POR MEIO DA AQUISIÇÃO DE DADOS VIA APLICATIVO ANDROID COM USO DE ALGORITMO DE APRENDIZADO DE MÁQUINA*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1. Eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.

_____

Lucas Cavalcanti Adorno

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

DEDICATÓRIA

Dedico a elaboração do trabalho à minha família e à comunidade acadêmica, em especial ao meu orientador Flávio Luis de Mello e também ao Fabrício Firmino por estar sempre disposto em ajudar e de elaborar um projeto importante para a sociedade como um todo.

AGRADECIMENTO

Agradeço a todas as pessoas presentes nesses longos anos de caminhada na faculdade. É notório que este foi o momento da minha vida de maior transformação psicológica e caráter como um todo. Por todos os momentos presenciados na UFRJ, sou grato a todos que participaram direta e indiretamente.

RESUMO

O presente trabalho tem como finalidade desenvolver um aplicativo para dispositivos Android que permita, de forma intuitiva, a captura de informações das vias pavimentadas. A identificação dos eventos existentes nos trajetos percorridos, como buracos e quebra-molas, será realizada de forma automatizada através do uso de um algoritmo de aprendizado de máquina. Além disso, não menos importante, será desenvolvido um serviço público hospedado na internet que possibilite o consumo por qualquer pessoa de quaisquer interesses nos dados adquiridos pelo aplicativo mobile. O projeto como um todo utiliza as linguagens de programação Python, C# e Java. O aplicativo é desenvolvido para smartphones Android acima da versão Kitkat (4.x). Neste, é concentrado a lógica para captura de dados do sensor acelerômetro e também das coordenadas do GPS presente no dispositivo móvel. A etapa do processamento dos dados se dá através de processos desenvolvidos em C# que estão hospedados na *cloud* e também de algoritmos em Python responsável por toda a lógica de *machine learning*. A respeito do acesso aos registros salvos no banco de dados, foi criado uma *Web API* em C# com a IDE Visual Studio 2017 que realizará o intermédio entre o dado e o usuário. Nessa aplicação web, o usuário poderá acessar dados de quaisquer obstáculos, em qualquer intervalo de tempo, de uma localidade específica e outras funcionalidades que serão apresentadas ao longo do documento. Neste trabalho, são abordados tópicos a respeito das linguagens de programação, tecnologias empregadas, assim como os conceitos matemáticos e estatísticos manuseados para a construção do projeto. Por fim, são descritas e apresentadas as conclusões a respeito dos resultados obtidos, além da descrição de todas as etapas construídas para o funcionamento da aplicação.

Palavras-Chave: aprendizado de máquina, captura de obstáculos na pista, identificação automática de obstáculos na pista, mapeamento da qualidade asfáltica, linguagem de programação, aplicativo android, desenvolvimento web API.

ABSTRACT

The present work aims to develop an Android application that captures paved roads data while driving a vehicle. The identification of the events, such as holes and speed bumps, that occur during the driving will be performed in an automated way through the use of a machine learning algorithm. Furthermore, but not less important, a public web servisse will be developed allowing anyone, despite the interest, to visualize and use the captured data by the mobile application. The Project as a whole uses Python, C# and Java programming languages. The application, called Lunar, has been developed in order to work on smartphones running on Kitkat version (4.x) or greater Android OS version. In this step, is concentrated the logic to capture accelerometer's data and the respective GPS coordinates. The data processing step is done through processes developed in C# that are hosted in the cloud. Moreover, the machine learning logic has been developed through Python coding. Concerning the access records stored on this database, a Web API will be created using C# and Visual Studio 2017 IDE, where this component will work as the intermediate between the stored data and the final user. During this process the user will be able to access the data of any obstacles (event) at any given moment from any location along with other features that will be discussed throughout this document. Along the text are chapters concerning the use of the programming languages adopted, cloud technologies, as well as the mathematical and statistical concepts handled to develop the Lunar Project. Finally, personal conclusions will be taken and the results will be described and explained together with a description of all stages built for the mobile app operation.

Key-words: machine learning, obstacle capture on paved roads, automatic obstacle identification, asphalt quality mapping, programming language, android application, web api development.

SIGLAS

API	-	Interface de Programação de Aplicações
AWS	-	Amazon Web Service
DNIT	-	Departamento Nacional de Infraestrutura de Transportes.
FTP	-	Protocolo de Transferência de Arquivos
HTTP	-	Protocolo de Transferência de Hipertexto
IDE	-	Ambiente de Desenvolvimento Integrado
IEEE	-	Instituto de Engenheiros Eletricistas e Eletrônicos
IoT	-	Internet das Coisas
JSON	-	Notação de Objetos JavaScript
LSTM	-	Memória de Longo Prazo
MBA	-	Mestre em Administração de Negócios
OMS	-	Organização Mundial da Saúde
REST	-	Transferência de Estado Representacional
RNN	-	Rede Neural Recorrente
RPC	-	Chamada de Procedimento Remoto
SMTP	-	Protocolo de Transferência de Correio Simples
SOAP	-	Protocolo Simples de Acesso a Objetos
UFRJ	-	Universidade Federal do Rio de Janeiro
URL	-	Localizador Uniforme de Recursos
WSDL	-	Descrição de Linguagem de Web Services
XML	-	Linguagem Extensível de Marcação Genérica

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Tema

O tema deste projeto é a utilização do sensor acelerômetro e do GPS do sistema operacional Android para captura dos obstáculos em vias pavimentadas e a criação de um serviço público web para facilitar, aos usuários, o acesso aos dados. Neste sentido, o problema a ser resolvido é a criação de uma aplicação de baixo custo capaz de capturar as informações necessárias e também de identificar automaticamente os eventos percebidos, com o propósito de realizar a medição das condições de rodagem de uma malha rodoviária utilizando smartphone.

1.2 – Delimitação

Ao que se refere ao processo da captura dos dados, o público-alvo é divido em dois diferentes grupos de usuários. Os usuários comuns de dispositivos móveis que possuem os recursos necessários no aparelho e que querem ampliar a base de dados de forma voluntária. O segundo grupo está relacionado com os usuários que trabalham para alguma instituição pública e que almejam elaborar estudosa respeito do mapeamento das condições das vias pavimentadas contribuindo com a inserção de novos registros ao banco de dados.

No que tange na utilização do serviço web para acessar os dados, é destinado aos funcionários de serviços públicos, corpo discente e docente de quaisquer instituições de ensino, como também interessados em trabalhar com os dados para algum projeto pessoal, empresarial ou governamental.

O tipo de rua a ser analisado também contém algumas restrições. Somente estradas pavimentadas serão estudadas pela aplicação. Pistas de terra e paralelepípedo não se enquadram no escopo do projeto, devendo ser objeto de trabalho a parte.

Apesar desta delimitação, toda a sociedade poderá tirar proveito dos resultados obtidos pela aplicação que permitirá a elaboração de um mapa da qualidade asfáltica de qualquer região que tenha sido percorrida por algum usuário pelo menos uma única vez e, consequentemente, facilitará a decisão de medidas para realizar as correções necessárias nos trajetos registrados pelo projeto.

1.3 – Justificativa

É sabido por todos os brasileiros que a qualidade das estradas pavimentadas no Brasil não é sinônimo de orgulho ou de um trabalho bem feito. Deixando os motivos de lado, constantemente são encontrados, mesmo nas estradas recém-inauguradas, problemas como fissuras, deformações e buracos.

Além da má impressão que passa ao se deparar com estes eventos, o principal problema é a contribuição direta para o número de acidentes em vias, tanto no caso brasileiro como para os outros países do mundo. Segundo um estudo publicado em 2015 pela *World Health Organization* (OMS) [0], o Brasil, em 2013, ficou na primeira posição com o maior número de mortes no trânsito a cada 100 mil habitantes considerando os países da América do Sul, seja por problemas na via ou devido à imprudência dos motoristas. Ao mesmo momento, o Brasil é destacado por aplicar numerosas leis de controle de risco.

O governo brasileiro, mais especificamente o DNIT, é o responsável pela regularização e normas de manutenção das vias. Além disso, este departamento realiza o mapeamento das condições dos pavimentos. O processo é custoso e exige uma série de ferramentas, como também etapas penosas para escanear as estradas. O interesse é público, porém os dados adquiridos são privados.

A aplicação desenvolvida neste projeto, inspirada nos apontamentos de Laudelino Amaral de Oliveira Lima [2], apresentará uma forma de custo ínfimo para permitir o mapeamento das regiões que possuem vias pavimentadas e, ao mesmo tempo, os dados poderão ser analisados, observados por qualquer cidadão interessado.

1.4 – Objetivos

O trabalho, na sua essência, possui três objetivos primordiais que são: elaboração de um aplicativo mobile, desenvolvimento de algoritmo de aprendizado de máquina e a criação de um serviço web que faça o intermédio entre o usuário final e a base de dados. Assim, os objetivos específicos do trabalho são:

1. Criação de uma aplicação Android para a coleta dos dados dos sensores acelerômetro e GPS.
2. Criação de processos internos para processar os dados enviados pelo smartphone e salvá-los em um banco de dados.
3. Estudo e aplicação de serviços na nuvem para a hospedagem tanto dos processos como do banco de dados.
4. Criação de um serviço web para facilitar o acesso dos usuários aos dados.

1.5 – Metodologia

O foco inicial foi em como e quais dados deveriam ser capturados para que a base de dados tornasse o projeto viável. A aprendizagem foi iniciada estudando o sensor acelerômetro presente do sistema operacional Android pelo fato de fornecer as acelerações nos eixos x, y e z do dispositivo em que está sendo observado. O sistema operacional Android foi escolhido devido a sua presença abundante no mercado mundial.

Foi realizada uma aplicação teste para os dispositivos Android com o intuito de verificar se este componente eletrônico era sensível o suficiente para a medição dos dados do projeto. Constatado a alta sensibilidade, os testes para captura e análise dos dados foram iniciados.

O celular com o aplicativo instalado foi inserido em um suporte para smartphone no ar-condicionado permanecendo-o na posição vertical. Com o auxílio de uma câmera fixada no capô do automóvel para realizar a filmagem de todo o percurso, tornou-se possível uma validação e comparação dos dados capturados com as imagens do vídeo adquirido.

Anteposto à elaboração dos gráficos após aquisição dos dados, foram realizadas pesquisas em artigos científicos relacionados ao tema proposto, a fim de obter uma maior clareza sobre os métodos atuais para o mapeamento das malhas de rodagem tanto no

Brasil como nos demais países. Além disso, iniciou-se um estudo para encontrar o melhor método de aprendizado de máquina que atendesse as exigências do projeto.

Após a pesquisa e a elaboração dos gráficos comparando as informações dos sensores com os eventos que estavam ocorrendo no automóvel nos mesmos instantes de tempo, foi constatado a viabilidade da realização de um mapeamento de baixo custo através de algoritmos de aprendizagem de máquina para detectar, de forma automatizada, os obstáculos nas vias pavimentadas.

A partir da viabilidade do mapeamento ser observada tanto de modo manual como automatizado, permitiu dar a continuidade para as próximas etapas e também ao projeto como um todo. Baseado nesta factibilidade, deu início à elaboração de uma plataforma Web, mais especificamente de uma Web API, permitindo a qualquer usuário solicitar e estudar os dados simplesmente chamando uma URL em um formato específico.

1.6 – Descrição

No capítulo 2 será introduzido os fundamentos teóricos a partir dos quais constroem-se algumas das tecnologias e soluções desenvolvidas no presente trabalho. É feita uma breve descrição de cada linguagem de programação utilizada e de conceitos tanto relacionados aos serviços web, como também do funcionamento dos tipos mais comuns do sensor acelerômetro. Ainda no capítulo 2, um outro tema relevante levantado é a apresentação de artigos científicos que foram estudados e utilizados como base para a construção do projeto Lunar.

O capítulo 3 trata-se especificamente das tecnologias da nuvem. Nele, será primeiro comentado a respeito da história da *Amazon Web Service* (AWS), de forma a clarear o que se trata uma empresa especialista na *Cloud* e como são estruturados e disponibilizados alguns dos seus serviços. Depois disso, será falado sobre os dois principais serviços que foram utilizados no projeto que são: EC2 e SQS.

O capítulo 4 será o momento da explanação do que se trata o conceito de *Machine Learning* e de como ele será útil para o objetivo do trabalho. É feito uma introdução ao modelo de aprendizagem de máquina chamado *Long Short-Term Memory* (LSTM).

O capítulo 5 é dedicado exclusivamente para apresentar todos os passos executados para criar a aplicação Lunar. A primeira seção deste capítulo expõe o que foi feito para estabelecer quais dados deveriam ser capturados e quais poderiam ser

ignorados. Seguido disso, fala-se do método desenvolvido para realizar a aquisição dos dados, dos artefatos tecnológicos empregados para tornar viável as primeiras validações das informações. Por intermédio de gráficos das assinaturas do sensor acelerômetro e de imagens tiradas dos obstáculos sofridos pelo veículo enquanto realizava o percurso, é feita uma análise diagnóstica aonde o foco está na relação de causas e consequências percebidas ao longo do tempo dos trajetos executados. Posteriormente, é apresentado os filtros aplicados tanto nos códigos relacionados à aprendizagem de máquina, como também àqueles inseridos nos processos em que recebem e enviam os dados para serviços e máquinas hospedados na Nuvem. Por fim, não menos importante, é exposto um diagrama de blocos contendo as principais funções desenvolvidas e a explicação de cada uma delas, imagens que servirão como base para a análise de dados, um serviço web para disponibilizar todo o conteúdo capturado e das informações extraídas.

Na conclusão serão comentados os resultados obtidos, assim como opiniões a respeito dos processos que deram certo, obstáculos encontrados ao longo do desenvolvimento e das limitações existentes no projeto Lunar.

Capítulo 2

Fundamentação Teórica

Para uma melhor compreensão dos assuntos abordados no projeto, são apresentados nas seções seguintes os conceitos fundamentais e suas premissas. Nas seções 2.1, 2.2 e 2.3 são dedicadas à apresentação das linguagens de programação Python, C# e Java respectivamente, nas quais foram utilizadas ao longo de toda a aplicação. Na seção 2.4 é apresentado o conceito de Web API e alguns exemplos com a utilização dos métodos principais de requisições HTTP. Em seguida, é realizada um panorama sobre o sensor acelerômetro e como será empregado no projeto. Adiante, é apresentado a concepção de uma das técnicas de normalização conhecida como *Sliding Window* e o porquê de utilizar técnicas do gênero em base de dados cruas. Por fim, são feitas uma análise e uma breve descrição da monografia do ex-aluno de MBA Laudelino Amaral de Oliveira Lima, como também periódicos da IEEE relacionados ao escopo do projeto.

2.1 – Linguagem Python

Criada em 1991 por Guido Van Rossum, Python é uma linguagem de programação de alto nível, tipagem dinâmica e orientada a objeto. Reconhecida pela sintaxe clara, excelente documentação e com uma comunidade ativa, Python vem sendo empregada em uma ampla variedade de projetos de quaisquer complexidades.

A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código à velocidade ou expressividade. Combina uma sintaxe clara e concisa com os recursos nativos e também de pacotes desenvolvidos por terceiros já que o seu modelo de desenvolvimento é comunitário e aberto as demais pessoas que queiram contribuir para a comunidade.

Python torna-se uma das linguagens mais utilizadas pela comunidade científica devido as qualidades mencionadas. Existem bibliotecas específicas para cálculos científicos e plotagem de gráficos fazendo pouco uso de linhas de código. A comunidade desenvolveu e continuam desenvolvendo o *SciPy* assim como *Numpy*, *Pandas*, *Matplotlib*

que são ferramentas *open-source* voltadas para o meio acadêmico. A utilização também é abundante fora do meio científico. Por apresentar uma sintaxe clara, a linguagem é comumente apresentada para estudo em disciplinas de introdução à programação nas Universidades e cursos técnicos tornando-se porta de entrada dos estudantes ao ambiente da programação.

A biblioteca *Scipy* será necessária para os interessados em utilizar as classes e os métodos referentes ao aprendizado de máquina. Ao realizar a instalação do pacote do *Scipy*, uma série de outros importantes módulos são adicionados ao Python instalado no computador do usuário. O *Numpy* incrementa o Python com um suporte para trabalhar com matrizes multidimensionais, operações matemáticas complexas, cálculos entre outros conceitos da Engenharia. O módulo *Pandas* auxilia na manipulação e análise dos dados, nas manipulações nas tabelas e na estrutura dos dados.

Um dos módulos incorporados na biblioteca *Scipy* e que, no escopo em questão, serve para trabalhar em um sistema de detecção automática de obstáculos é o *Scikit-Learn*. Na Figura 1 é apresentada a página inicial deste módulo.

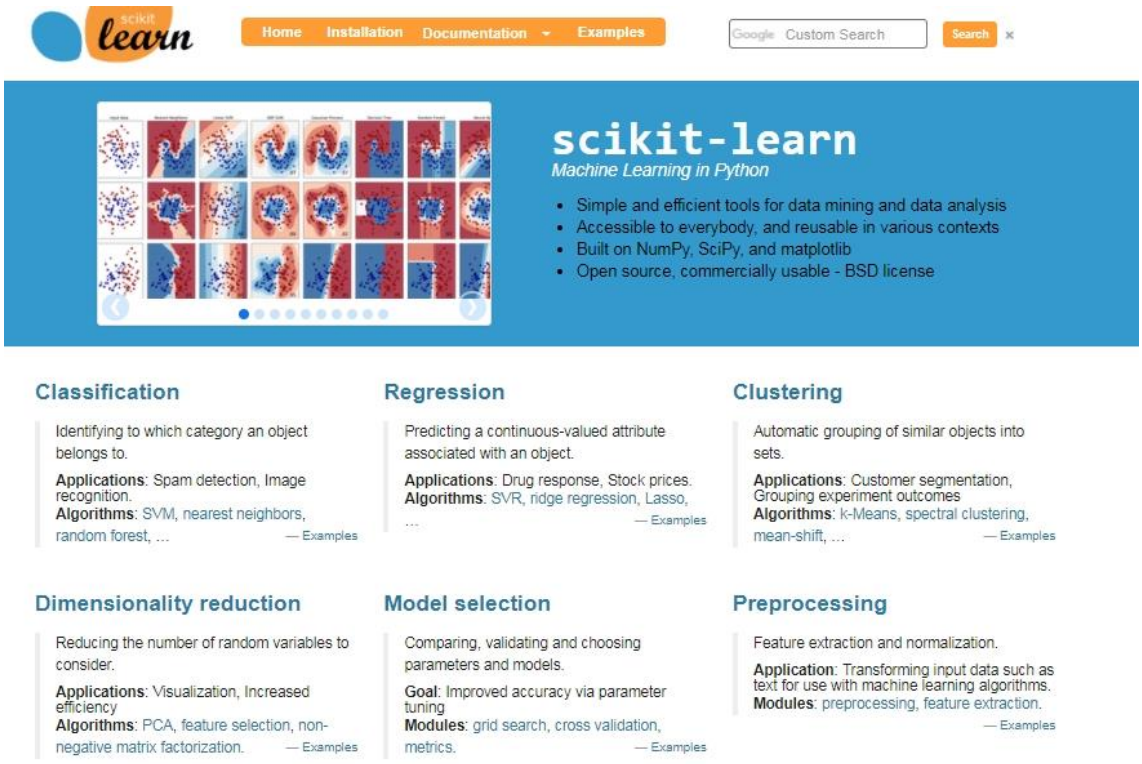


Figura 1 Página principal do módulo Scikit-Learn

O *Scikit-Learn*, rico em sua documentação, contém toda a base de informação para a realização de algoritmos de classificação, regressão, redução da dimensionalidade, pré-processamento e outros processos importantes para os desenvolvedores que querem trabalhar com algoritmos de aprendizado de máquina.

2.2 – Linguagem C#

Inicialmente alcunhada de *Cool* por Anders Hejlsberg e por outros integrantes da equipe formada pela Microsoft, a linguagem foi apresentada e renomeada nos anos 2000 no evento *Professional Developers Conference* (PDC) que reúne os desenvolvedores de software da Microsoft para apresentar as novas tecnologias do momento.

O C# é uma linguagem orientada a objeto que faz parte da plataforma .NET. Contém uma tipagem forte e é amplamente utilizada quando se trata da construção de softwares, sistemas operacionais e jogos em geral.

A IDE oficial para a elaboração de códigos em C# é a plataforma da própria Microsoft chamada Visual Studio. Mesmo possuindo suporte nativo para outras linguagens como Visual Basic (VB), C e C++ a ferramenta é completa para os desenvolvedores em C# que buscam criar serviços Desktop como também para a Web.

O serviço Web elaborado no projeto foi implementado com a linguagem C# através do Visual Studio Community 2017. O serviço foi implementado utilizando o framework Web API. Apesar de existir versões, como o Ultimate, que possui uma maior quantidade de recursos, optamos pelo Community por possuir os recursos necessários e gratuitos para o desenvolvimento da aplicação.

2.3 – Linguagem Java

Alcunhada de Oak no início de seu desenvolvimento, Java foi criado na mesma época que a linguagem Python. Desenvolvido pela Sun Microsystems e pertencido atualmente à Oracle, a linguagem nasceu com a finalidade de ser executada em diversas plataformas de hardwares diferentes [6]. Para obter esse funcionamento distribuído em sistemas operacionais distintos, a compilação de códigos em Java gera arquivos objetos

chamados de *byte-codes*, e estes podem ser executados por meio de interpretadores desenvolvidos para cada tipo de plataforma [6].

Lançada nos anos 90, Java ainda tem uma presença massiva no universo dos softwares desktops e mobiles. Segundo IEE Spectrum [7] no ano de 2017 o Java ocupa a terceira posição nas linguagens mais utilizados do Mundo.

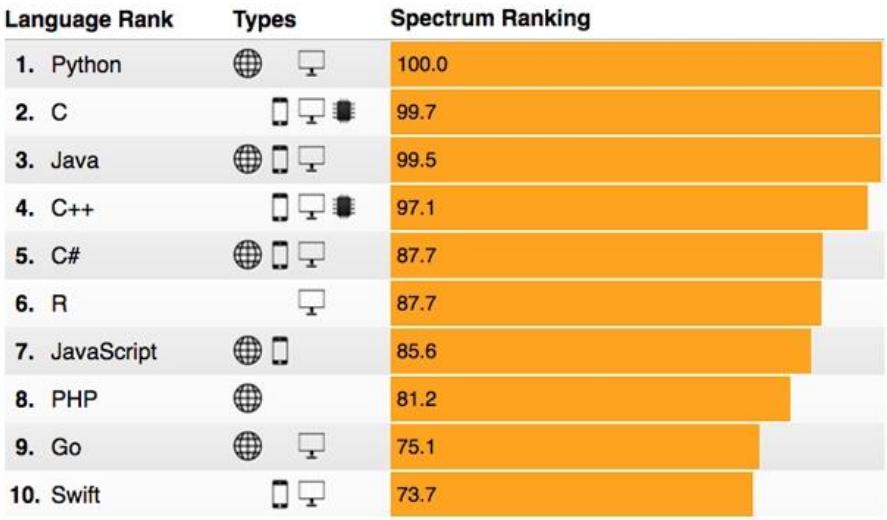


Figura 2 Ranking das linguagens de programação por IEE Spectrum em 2017

No caminho oposto da linguagem Python, a linguagem da Oracle não é *open-source*. O fato de não ser aberta ao público para desenvolver em conjunto, não impede que as bibliotecas se mantenham atualizadas e ricas em conteúdo. Java, por ter sido desenvolvida a um tempo considerável, possui bibliotecas muito bem consolidadas e uma grande quantidade de material disponível na internet para estudo.

No universo Mobile a situação não é diferente. Os aplicativos dos dispositivos móveis com o sistema operacional Android podem ser desenvolvidos utilizando outras linguagens, porém o recomendado pela própria Google, criadora do sistema Android, é em Java através da IDE oficial Android Studio. De acordo com a IDC Quarterly Mobile Phone Tracker [8], a presença de aparelhos Android no mercado é aproximadamente de 85% nos primeiros meses de 2017. A Figura 3 ilustra a diferença da presença e a sua evolução dos diferentes sistemas mobile no mercado mundial.

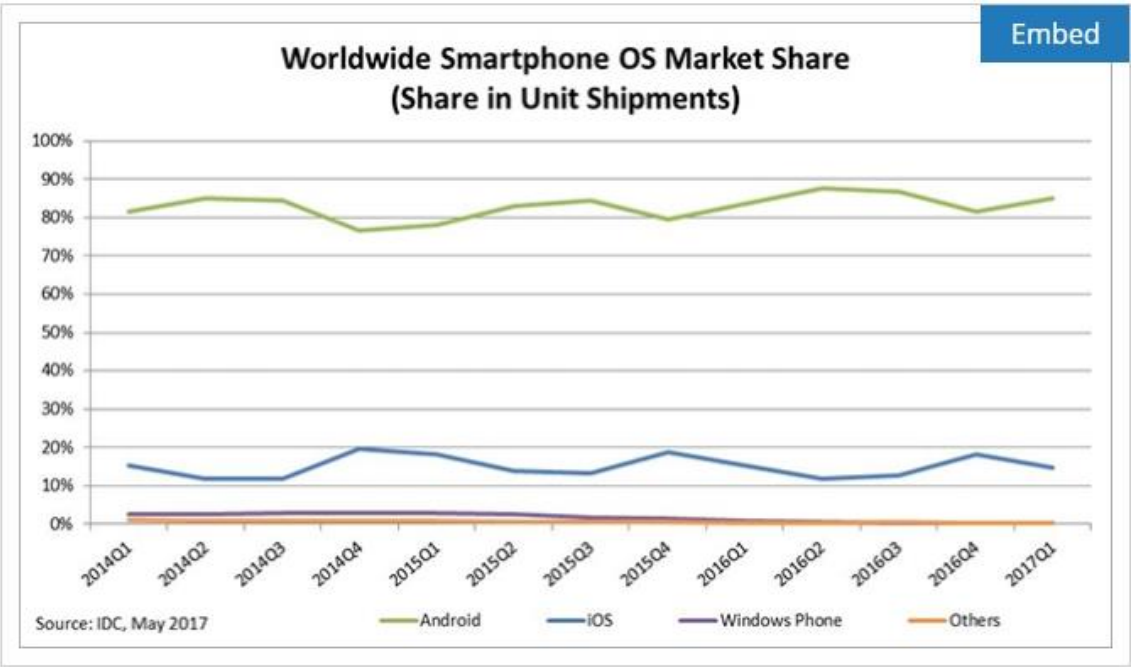


Figura 3 Participação entre os sistemas operacionais mobile no mercado em 2017

Em virtude da maior presença de dispositivos Android na sociedade, o aplicativo construído para a captura de dados do sensor acelerômetro e das coordenadas do GPS foi para smartphones Android, através da utilização da IDE oficial da Google em conjunto com a linguagem recomendada.

2.4 – Web API

A maioria das aplicações existentes utilizam serviços conectados à Internet. O motivo pode variar desde uma simples conexão com um banco de dados para consultar uma certa informação, ou mesmo pelo fato da aplicação como um todo estar hospedada na nuvem.

Hoje, consegue-se acessar um serviço qualquer por diversas plataformas e sistemas operacionais distintos. O uso de smartphones, por exemplo, vem crescendo exponencialmente e ultrapassando o próprio computador como ferramenta para acessar uma aplicação [9]. Com base nesse conhecimento, é razoável que os desenvolvedores de software levem em consideração estruturar um projeto para que o próprio funcione tanto em sistemas Windows, Mac, Linux e seus derivados.

Um dos serviços lançados pela Microsoft focados na questão de desenvolvimento para linguagens de programação e para sistemas operacionais diferentes é o framework

Web API. Fazendo uso deste framework em uma aplicação, não haverá distinção se o cliente está acessando através de um computador Linux, Mac OS X, dispositivo móvel ou até mesmo de aplicações desenvolvidas em outras linguagens de programação. O Web API independe do que está por dentro da "caixa preta" do cliente. São estabelecidas algumas regras para a conexão e para a configuração dos serviços HTTP construídos, permitindo que todas as funcionalidades do software funcionem.

Para realizar a conexão entre cliente e servidor são utilizados basicamente, como protocolo de comunicação, o SOAP e/ou REST. Na Figura 4 é apresentada uma comparação entre os dois tipos de comunicação:

SOAP	REST
Protocolo de Mensagem baseada em XML	Estilo de Arquitetura
Usa WSDL para comunicação entre cliente e serviço	Usa XML ou JSON para enviar e receber dados
Invoca serviços por método RPC	Invoca serviços através da URL
Retorno do dado em formato complexo para a leitura humana	Retorno do dado em formato simples para leitura humana
Transferência pode ser por HTTP, SMTP, FTP etc.	Transferência somente via HTTP
Performance pior em comparação ao REST	Performance superior ao SOAP em uso de CPU, leitura de código etc.

Figura 4 Diferenças entre o protocolo de comunicação SOAP e REST

O protocolo de transferência mais utilizado por desenvolvedores de softwares que necessitam realizar requisições Web em seus produtos é o HTTP. Este protocolo possui métodos bem definidos, sendo os principais listados na Figura 5.

Métodos Principais	Definição
GET	Requisita um recurso do servidor
POST	Envia um recurso para o servidor
PUT	Atualiza e/ou Insere um recurso no servidor
DELETE	Apaga um recurso do servidor
HEAD	Requisita os cabeçalhos de uma resposta
PATCH	Atualiza parcialmente um recurso do servidor

Figura 5 Definição dos métodos da requisição HTTP

Para exemplo de demonstração, será analisado um breve funcionamento de um Web Service público que, ao informar o nome do país, são retornados nomes de cidades do mesmo. O host desta aplicação aberta a qualquer usuário possui a seguinte URL: www.webservicex.net/globalweather.asmx/GetCitiesByCountry

Em requisições GET, os parâmetros devem ser passados na construção do próprio link do serviço. Para separar o que é o direcionamento para um determinado método da sua aplicação e quais são os parâmetros, é necessário o uso do símbolo "?". Ademais, os parâmetros devem ser organizados em pares de chave-valor.

Neste Web Service, o nome do parâmetro que será preenchido com o nome do país é "CountryName". Portanto, a construção final da URL escolhendo "United States" como base ficará no seguinte formato:

www.webservicex.net/globalweather.asmx/GetCitiesByCountry?CountryName=United States

Os dados podem ser retornados via XML ou JSON. Neste caso, o desenvolvedor deste serviço público web optou pela tecnologia XML. A seguir está uma parte do retorno do envio da URL acima.

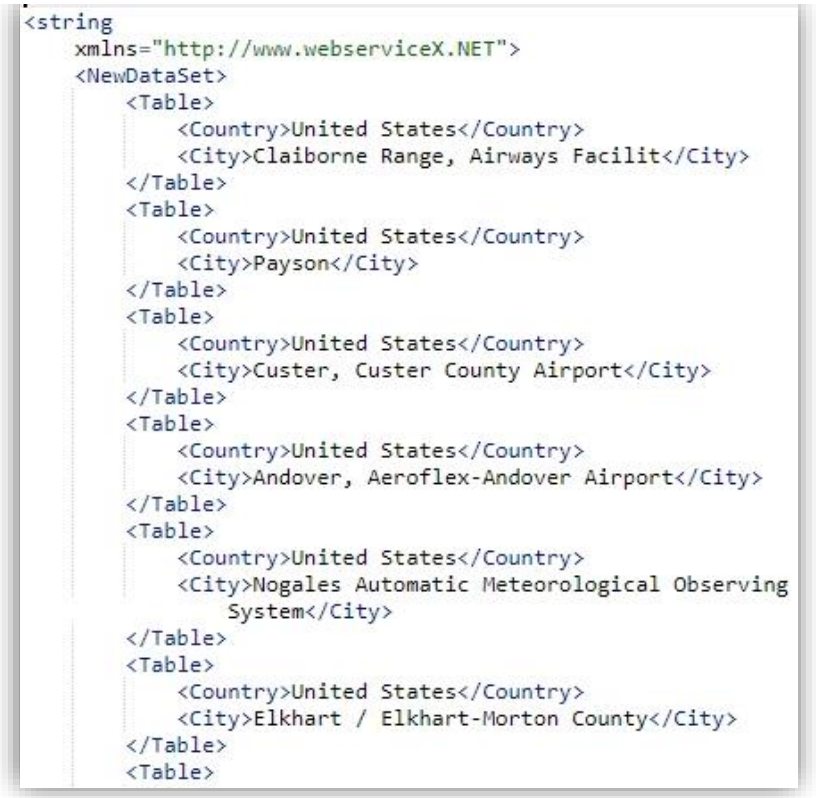


Figura 6 Exemplo do resultado da API GlobalWeather do site webservicex

Aparentemente o XML retornou uma codificação fácil de ser interpretada, mas não é comum de acontecer. Neste caso ocorreu devido a passagem de somente 1 parâmetro e que o mesmo possui uma estrutura simples.

A respeito do método POST que é amplamente utilizado, as informações não são mais visíveis e expressas pelo próprio link no navegador. Os parâmetros serão inseridos no *Body* (corpo) da requisição. A seguir consta como deve ser a construção do *Header* e *Body* da solicitação por um determinado recurso que se encontra no servidor.

Figura 7 Header e Body da requisição da API GlobalWeather

As informações contidas no retângulo de cor vermelha correspondem a construção do *Header* (cabeçalho) e a parte de fora do *Body*. Diferentemente da requisição GET que o próprio usuário utiliza a própria URL do serviço para informar os parâmetros, no POST o interessado deverá preencher a solicitação do recurso através de uma aplicação.

2.5 – Sensor Acelerômetro

O acelerômetro é um dispositivo eletrônico utilizado para medir as acelerações nos eixos x, y e z em relação à gravidade do ambiente. Todo o deslocamento de massa implica induzir uma velocidade, direção e sentido. A partir da variação da velocidade obtida, tem-se como resultado a aceleração independentemente da direção aplicada.

Uma questão relevante é saber interpretar os dados deste tipo de sensor e por isso é preciso conhecer seu sistema referencial. O posicionamento dos eixos do sensor acelerômetro em dispositivos móveis é apresentado na Figura 8.

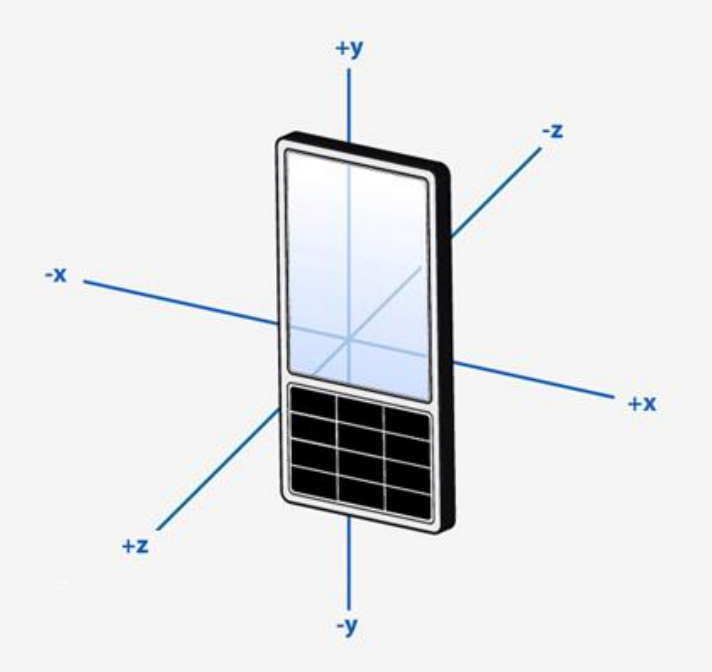


Figura 8 Eixos X, Y e Z do sensor acelerômetro para smartphones

Este sensor, presente na maioria dos smartphones comercializados, é amplamente utilizado em sistemas de posicionamento, inclinação e de vibração.

O seu princípio baseia-se na Segunda Lei de Newton que relaciona a força resultante aplicada em um corpo com a massa e a velocidade. Conhecendo o valor da força imposta no sensor, podemos obter a variação da velocidade em relação ao tempo.

Existem diversas formas de construir um acelerômetro. Cada forma parte de um princípio físico distinto para obter o mesmo resultado que é a medição da aceleração. Alguns tipos de acelerômetros mais utilizados são: Capacitivo, Piezoelétrico e Piezoresistivo [2].

O acelerômetro capacitivo baseia-se no movimento que uma massa de prova, suspensa por molas, faz entre duas placas fixas ao substrato do sensor. Quando há uma aceleração sendo aplicada, as capacitâncias entre a massa de prova e cada uma das placas variam antagonicamente. Isto é, enquanto uma capacitância possui uma variação positiva, a outra terá uma perda em valor absoluto. Na Figura 9 apresenta uma ilustração esquemática de como se comporta este tipo de sensor. Nesta figura, quando há uma aceleração sendo aplicada, a massa de prova fica mais próxima da placa fixa que consta o capacitor C2. Como a distância entre esses dois objetos será menor, o capacitor C2 terá uma maior capacitância. A variação desta é proporcional à aceleração.

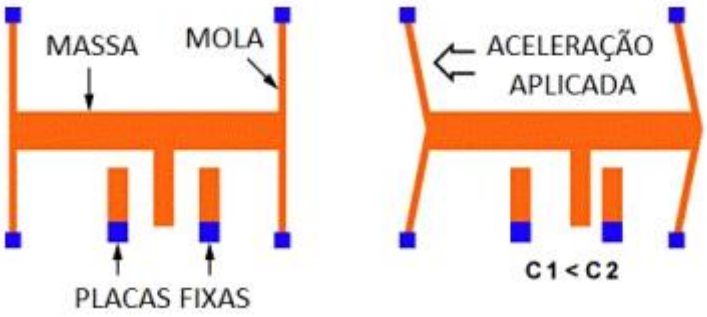


Figura 9 Estrutura do sensor acelerômetro capacitivo

Este tipo de acelerômetro também está presente nos smartphones modernos. Como apresentado nas Figuras 10 e 11, o circuito é composto de um elemento móvel, o *Seismic Mass*, que vibra sobre a base (*Housing*) em caso de movimentação do dispositivo móvel. Em cada vibração, as aletas do *Seismic Mass* se aproximam dos capacitores, alterando a capacitância. Os circuitos medem o diferencial da tensão e retornam o mesmo para a saída do acelerômetro. Cada valor é proporcional a movimentação das aletas.

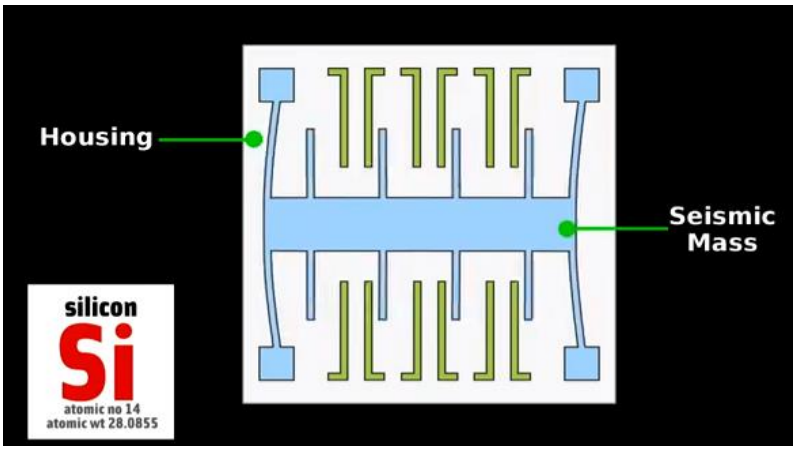


Figura 10 Estrutura do sensor acelerômetro capacitivo em smartphones

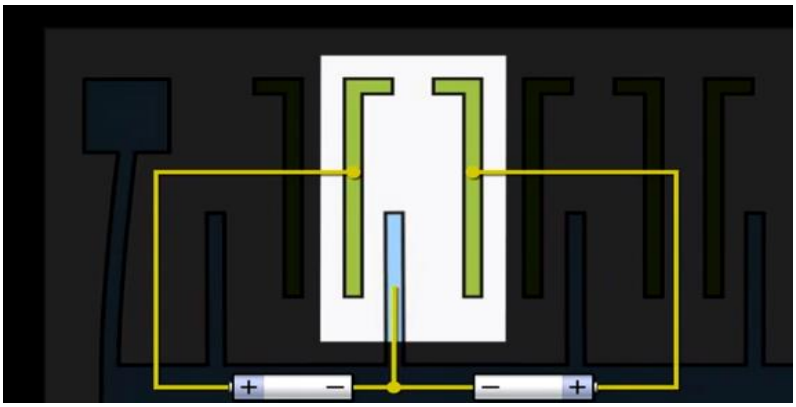


Figura 11 Detalhes externos do sensor acelerômetro do tipo capacitivo

O acelerômetro do tipo Piezoelétrico remete a capacidade de alguns cristais gerarem tensão elétrica por resposta a uma pressão mecânica. Normalmente, há uma massa presa a um cristal piezoelétrico e quando há uma aceleração no sistema (smartphone) a massa presa ao cristal acaba gerando uma deformação no cristal e este deslocamento gera um sinal elétrico proporcional à força aplicada. Análogo ao Piezoelétrico, o Piezoresistivo é relacionado à variação da resistência do circuito ao ocorrer uma deformação do cristal através da massa acoplada.

No projeto, esse dispositivo eletrônico será utilizado para capturar irregularidades nas estradas pavimentadas através dos deslocamentos no espaço impostos no veículo e, consequentemente, no smartphone também.

2.6 – Normalização

Os dados capturados pelo sensor acelerômetro assim como em outros sensores possuem ruídos vinculados. Uma técnica bastante utilizada para as pessoas que trabalham com dados sujeitos à ruídos é aplicação de técnicas de normalização.

A normalização consiste em suavizar os ruídos presentes no sinal que está sendo capturado ou em uma base de dados já existente. Para amenizar esses dados inconvenientes, aplica-se a seguinte fórmula.

$$Z = \sqrt{\frac{(x - \acute{x})^2}{\sigma x}}$$

A variável de saída “Z” é o dado normalizado, “x” o valor que se deseja suavizar o ruído, “$\overline{x}$” como a média dos dados e “σ” o desvio padrão do sinal.

Existem alguns tipos de normalização e os dois principais são: normalização completa e por janela. A completa consiste em calcular a média e desvio padrão de todo o conjunto de dados e aplicá-los na fórmula acima para encontrar o vetor de saída “Z”. Este método é apropriado quando o sinal tende a ser uniforme com a adição de um ruído ao mesmo.

A normalização por janela também conhecida como *Sliding Window,* contempla um curto intervalo de amostras, o que reflete em valores de média e desvio padrão somente para o escopo observado. O sinal de saída irá conter picos mais destacados devido a remoção ou suavização dos ruídos existentes. A amplitude dos picos dependerá do valor ponderado do desvio padrão em relação à média do sinal.

2.7 – Trabalhos Relacionados

Este trabalho dá continuidade à pesquisa de Laudelino [1] que fornece uma base de dados importante para se ter um entendimento melhor de como o órgão federal brasileiro realiza atualmente a classificação das ruas pavimentadas no Brasil. Ele cunhou o termo “coeficiente lunar” como uma variável para quantificar ou melhor, para qualificar as vias pavimentadas. Além disso, explicita todos os itens que são considerados pelo DNIT, desde fenda, afundamento, até por tipos específicos de cada item como por exemplo “Trinca do tipo Couro de Jacaré”. Além de abordá-los, também é informada uma tabela contendo todos os detalhes oficiais a respeito das classificações desses defeitos nas vias. Ademais, é comunicado como é feito a identificação das condições dos pavimentos citando o Deflectógrafo Lacroix, veículos específicos utilizados e o processamento de imagem embarcado. A monografia é recomendada para os leitores que querem aprofundar mais para entender o processo de mapeamento das ruas pavimentadas no Brasil.

O desenvolvimento do projeto Lunar também se baseou no artigo científico publicado pelos autores Christian Koch e Ioannis Brilakis [3]. Este artigo apresenta resultados dos estudos envolvendo a análise de imagens de vias pavimentadas com o auxílio de uma câmera de alta velocidade embarcada em um veículo. O conhecimento foi extraído através de 120 imagens sendo 50 para treino dos algoritmos e 70 para teste a fim de aperfeiçoar os códigos elaborados.

O estudo foi motivado ao se depararem que, no ano de 2009, o Departamento de Transporte dos Estados Unidos da América utilizava métodos manuais para detectar as fissuras, buracos, remendos e outros tipos de obstáculos nas vias pavimentadas das regiões dos EUA e do Canadá. A coleta e o armazenamento dos dados eram através do uso de câmeras acopladas em veículos do departamento, porém a sua análise era feita manualmente através de técnicos especializados. Os resultados eram influenciados pela subjetividade e pela experiência dos avaliadores.

Desde então, esforços foram dedicados tanto pelos autores como também por outros laboratórios na área de detecção automática de buracos e outros obstáculos nas vias pavimentadas. A detecção, segundo Koch e Brilakis [3], é dividida nas seguintes etapas: reconstrução do pavimento em 3D, emprego de digitalização laser, *computer stereo-vision* e utilização de acelerômetros. Enquanto as duas primeiras etapas sofrem com alto custo de equipamentos e de esforço computacional, as demais são destinadas para levantamento de mais informações para o aumento de precisão e da confiabilidade dos dados.

A respeito do uso de acelerômetros, descobriram que as abordagens baseadas exclusivamente em vibrações poderiam fornecer resultados errados, informando buracos por engano ou falsos negativos. Buracos localizados no centro de uma pista muitas vezes não são atingidos pelas rodas dos carros e, portanto, não podem ser reconhecidos pelo sensor. No entanto, estes obstáculos devem ser capturados visto que são defeitos de pavimentos e particularmente danosos aos motociclistas. Caso contrário, o mapeamento dos obstáculos das estradas não vai possuir uma precisão e uma confiabilidade significante.

O trabalho de Koch e Brilakis se baseia nas propriedades geométricas das regiões defeituosas dos formatos dos buracos utilizando técnicas de regressão e de processamento de imagem. Foram identificadas três características principais referentes à aparência visual dos *potholes*. São elas:

A. Um buraco inclui uma ou mais sombras que são mais escuras do que a área circundante.
B. A forma de um buraco é aproximadamente elíptica.
C. A textura da superfície dentro de um buraco é mais áspera e granulosa em relação a uma superfície intacta.

Com base nessas informações, o modelo proposto foi dividido na segmentação da imagem, extração da forma e comparação com superfícies próximas na mesma imagem para obter um diagnóstico final. Como resultado, obtiveram uma acurácia de 85,9%, precisão de 81,6% e sensibilidade de 86,1%.

Já Adarkwa e Attoh-Okine [4] chamam atenção sobre o uso da fatorização Tensorial para realizar a classificação dos obstáculos nos pavimentos. O artigo considerou somente as rachaduras longitudinais e transversais devido serem de um fenômeno de defeitos nas vias de maior ocorrência. Os danos causados por estes obstáculos atingem a cada de US$ 10 bilhões anuais no EUA o que explica o porquê do Departamento de Transporte dos Estados Unidos da América enfatiza o monitoramento e a manutenção das calçadas.

O estudo desenvolvido foi realizado em Matlab utilizando o Matlab Tensor Toolbox 2.5. Há quatro etapas importantes a serem citadas:

1) Pré-processamento e formação de *Training Tensor*
2) Pré-processamento de dados de teste
3) Decomposição do valor Singular da Ordem Superior do *Training Tensor*
4) Classificação

O conjunto é composto de 1830 imagens a serem pré-processadas, dividindo-se em grupo com e sem defeitos.

O pré-processamento envolve a conversão das imagens do formato RGB em *grayscale* e convertida para 100 pixels tanto de largura como altura. Para realizar o treinamento e o teste de validação, o conjunto de imagens é então dividido em dois grupos principais, são eles: rachaduras transversais e longitudinais. Em seguida, imagens de cada classe são empilhadas, separadamente, uma após a outra para formar o *Training Tensor*. Se, após a aplicação do algoritmo, a imagem for classificada contendo fissuras longitudinais, o nome do arquivo recebia a letra “l” e caso contrário a letra “t”. No entanto, destaca-se que o nível de precisão do algoritmo mudou com as mudanças no número de imagens nos conjuntos de treinamento pois é evidente que a precisão não depende apenas do número de imagens no conjunto de treinamento, mas também da variabilidade interna entre as classes.

Nitsche, et al. [5] realizaram uma abordagem diferente sobre como processar as informações das vias pavimentadas. Eles utilizaram como base para a captura, os sensores

acelerômetros e sensores que medem a velocidade e o comportamento das rodas dos carros de passeio.

Foi criado, pelos próprios autores, um índice que analisa o nível de rugosidades das vias estudadas. Apelidado de *weighted longitudinal profile* (*wLP*), eles conseguiram detectar os fenômenos de rugosidade com precisão utilizando diferentes algoritmos como *SVM*, *MLP* e *Random Florest*.

Neste artigo, a elaboração das fórmulas para obtenção dos resultados e os testes aplicados foram baseados em exatamente três tipos de fenômenos diferentes de rugosidades nas vias pavimentadas. São elas: Rugosidade Genérica, Rugosidade Pseudo-Periódica e Irregularidade Pontuais. Na Figura 12 pode-se notar a diferença de comportamento de cada variação de obstáculo que foi incorporado ao campo de pesquisa.

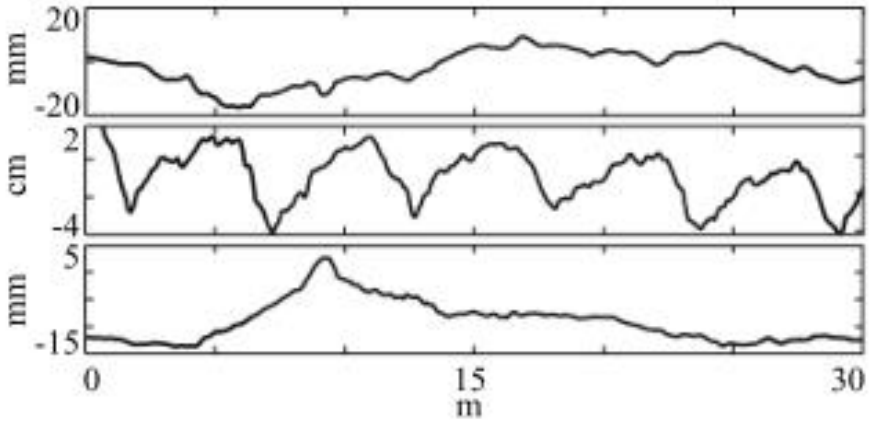


Figura 12 Comportamento em função do tempo da Rugosidade Genérica, Pseudo-Periódica e Pontual

Um outro intuito deste trabalho foi também apresentar um comparativo da eficácia de cada modelo de aprendizado de máquina aplicado, explicando as vantagens e as desvantagens de cada modelo estudado.

O método proposto pelos autores permite que o monitoramento da rede rodoviária seja alcançado por carros de passageiros convencionais e não em veículos especializados como as outras pesquisas indicam e se baseiam.

Além do parâmetro *wLP*, existe um outro indicador para a medição da rugosidade nas estradas. Criado no Brasil em 1982 através da parceria do Banco Mundial com equipes de pesquisa Brasileiros e de outros países, o Índice de Rugosidade Internacional (IRI) foi elaborado com o intuito de unificar os parâmetros utilizados nos diferentes países para determinar a rugosidade nas vias.

Sayers e Karamihas [10], em Setembro de 1998, realizaram um estudo chamado "*The Little Book of Profiling*" que fornece todas as informações básicas para o entendimento das medições e interpretações dos perfis rodoviários visto a tamanha relevância dos indicadores como o IRI para a realização do mapeamento de condições asfálticas. Iniciando desde a história da elaboração deste parâmetro até uma análise detalhada sobre o seu funcionamento, o IRI é apresentado como o primeiro índice de perfil amplamente utilizado, onde o método de análise é planejado para o trabalho com diferentes tipos de perfiladores tornando-o um índice reproduzível através de outros meios de cálculo.

Ainda nesse estudo elaborado em 1998, o IRI é reconhecido como o único parâmetro de alcance global que sumariza a qualidade asfáltica, podendo ser utilizado para o cálculo de custo operacional, cargas dinâmicas inseridas nas rodas (isto é, danos nas estradas causados por veículos pesados ou por outros motivos que colocam em risco os veículos que trafegam na região) e condições gerais da superfície. Na Figura 13 apresenta os intervalos de IRI para diferentes tipos de rodovia e velocidade.

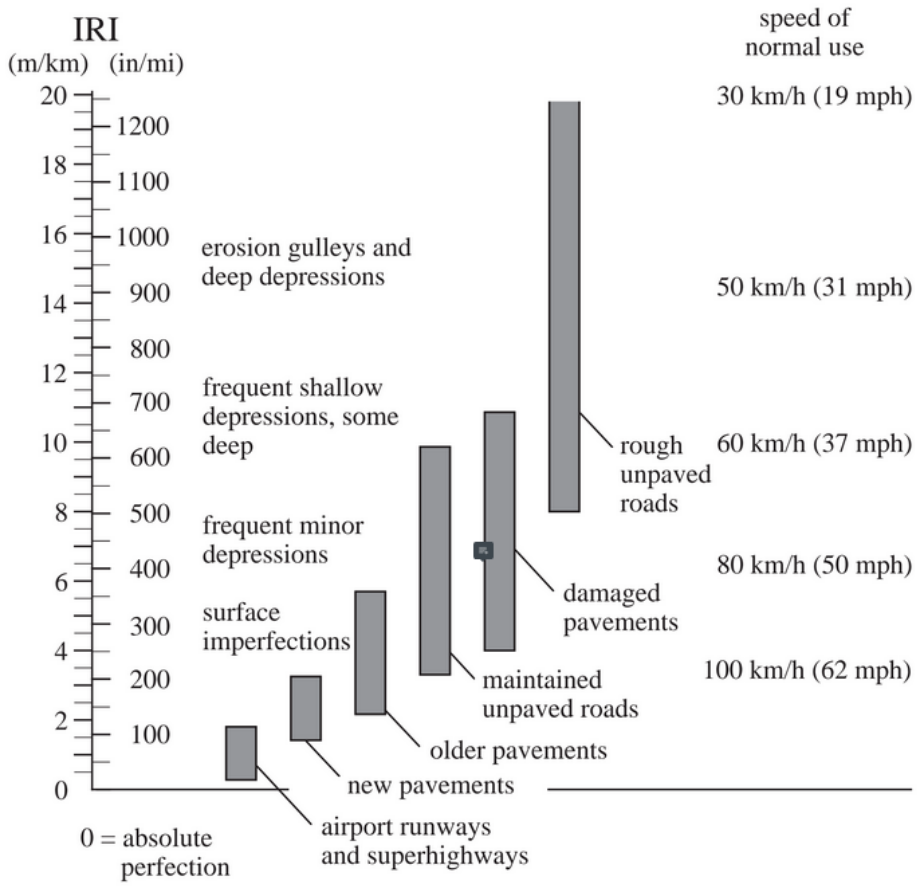


Figura 13 Intervalos de IRI em diferentes tipos de rodovias e velocidades

O modelo que o algoritmo do IRI utiliza como base é chamado de "*quarter-car"*. Esta construção teórica é exatamente o que o seu nome implica. O esquemático está apresentado na Figura 14 abaixo.

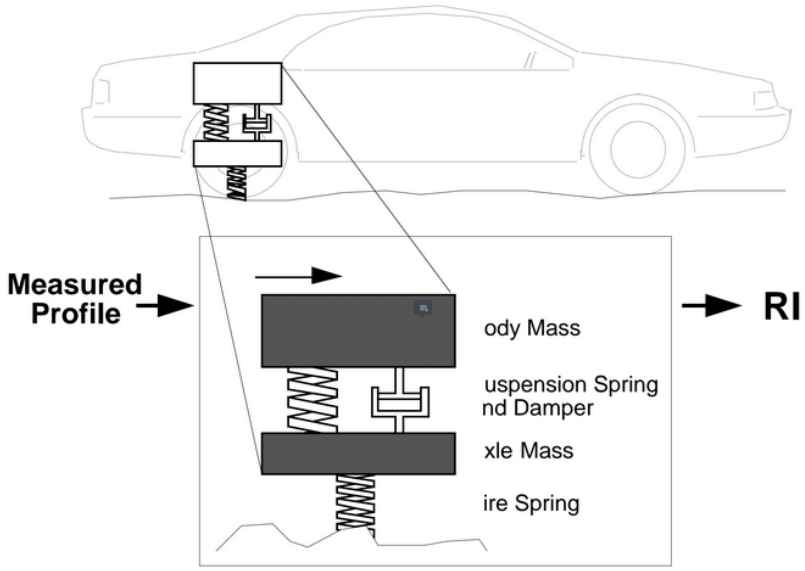


Figura 14 Funcionamento do modelo "quarter-car" utilizado pelo IRI

O filtro "*quarter-car*" calcula a simulação da deflexão da suspensão de um sistema mecânico. Mesmo tratando-se de uma simulação, os valores coincidem com as respostas físicas tanto para veículos de passeios e caminhões ao trafegarem por tipos diferentes de vias pavimentadas.

Uma outra característica importante do Índice Internacional de Rugosidade apontada por Sayers e Karamihas é o fato dele ser linearmente proporcional à rugosidade, o que significa que se ocorrer uma alteração dos valores de um perfil medido, então o IRI alterará exatamente na mesma proporção.

Vargas [29], em 2010, também elabora um estudo que apresenta os aspectos e as considerações importantes para a determinação das condições das estradas com base no cálculo do IRI. Em termos mais simples, Vargas descreve este parâmetro da seguinte forma: "O IRI é um modelo matemático que calcula o acúmulo de força aplicada na suspensão de um veículo de passeio percorrendo uma estrada em um intervalo de tempo".

Vargas expõe a relação do índice com as consequências das vias asfálticas. As variações significativas do IRI estão diretamente relacionadas com o maior custo de

manutenção para a administradora da pista como também para os motoristas que frequentam essa região.

O intervalo para as estradas pavimentadas é entre 0 a 12 m/km (metros acumulados por quilômetro percorrido) aonde o valor 0 representar uma superfície perfeitamente uniforme e 12 uma estrada intransitável. O perfil de um percurso recém-inaugurado possui um valor próximo a zero, visto que igual a zero é extremamente difícil no ponto de vista construtivo. Uma vez calculado o índice para uma determinada região, a rugosidade do pavimento tende a diminuir em função da circulação de veículos.

O autor cita alguns métodos para realizar a medição das variáveis que o IRI exige para poder ser quantificado. Estes métodos diferem na precisão e na velocidade dos resultados. Alguns citados foram o Perfilógrafo, Perfilômetro Inercial a Laser e Nível e mira Topográfica.

Como resultado do estudo, Vargas obtém que uma diminuição de 10% no valor do IRI de uma estrada aumenta a eficiência do combustível em aproximadamente 1,91 quilômetros por litro. Além disso, um acréscimo de 50% na diminuição de rugosidade equivale na elevação de 15% na vida útil do pavimento.

Diferente do meio proposto pelo projeto Lunar, os meios utilizados pelo Vargas e assim como pelos demais autores, são necessários uso de equipamentos extras para realizar a medição dos índices para fornecerem valores quantitativos a respeito da qualidade asfáltica ou até mesmo de veículos especializados como é o caso do Adarkwa e Attoh-Okine [4] e Koch e Brilakis [3].

Capítulo 3

Tecnologias na Cloud

Para uma melhor compreensão de toda a infraestrutura implementada no projeto, é necessária uma breve explicação sobre os serviços na nuvem que foram utilizados e também dos proprietários desses serviços de modo geral. Na seção 3.1, é exposto a empresa de *cloud computing* escolhida e os seus principais serviços para o escopo da aplicação desenvolvida. Adiante, nas seções 3.2 e 3.3, são abordados com mais detalhes os serviços utilizados.

3.1 – Amazon Web Service

Quando o assunto são empresas de *cloud computing*, três grandes nomes se destacam. São eles: Amazon Web Services (AWS), Microsoft Azure e Google Cloud Platform. Tratam-se de tecnologia recente, mas ao mesmo tempo, já possuem milhares de serviços robustos na nuvem.

O mais amadurecido desses serviços é a AWS. A empresa provê serviços online para websites, aplicações cliente servidor baseado nas nuvens, banco de dados, análise de dados, redes, dispositivos móveis, banco de dados, análise de dados, redes, dispositivos móveis, ferramentas de gerenciamento, IoT, armazenamento, aluguel de servidor entre outros [19]. Pode ser acessado via HTTP, protocolo REST ou SOAP.

Este provedor de serviço possui um plano gratuito (*free-tier*) que possui duração de 1 ano e que contempla benefícios mais do que suficientes para executar aplicações testes. Dentre alguns dos serviços fornecidos gratuitamente [20] estão: 750 horas por mês para alugar máquinas na Cloud, 5 GB de armazenamento de objetos, 750 horas por mês de uso de banco de dados e 1.000.000 de solicitações ao Amazon Simple Queue Service (SQS).

A maioria dos serviços presentes na AWS possuem um plano de pagamento chamado *pay-as-you-go*, isto é, pagamento conforme o uso. O custo é dado pelos serviços individuais que estão sendo utilizados, pelo instante que os utilizar, e sem necessidade de contratos para usufruir por um determinado período de tempo.

Figura 15 Distribuição das regiões disponíveis da AWS

A Amazon opera datacenters espalhados ao redor do mundo. Os locais são compostos por regiões e zonas de disponibilidade. Cada região é uma área geográfica separada, totalmente independente, e contém vários locais isolados conhecidos como zonas de disponibilidade. O fato das regiões serem completamente isoladas é o que proporciona a maior tolerância a falhas e estabilidade possível. Além desses benefícios, uma outra qualidade não menos importante é a possibilidade de distribuir uma aplicação entre várias zonas de disponibilidade, permitindo assim uma maior disponibilidade dos processos. Caso ocorra algum problema técnico em uma determinada região, haverá outras cópias da aplicação para manter as tarefas em andamento.

Ao fazer referência ao termo “escalabilidade”, um dos serviços fornecidos pela AWS que o projeto Lunar irá utilizar é o Amazon Simple Queue Service (SQS). Este serviço que, permite escalar um processo, será explicado com maior gama de detalhes na seção 3.3. Um outro recurso a ser utilizado pelo Lunar é o Amazon Elastic Compute Cloud (EC2). Esse, permite que os usuários aluguem computadores virtuais para rodar os processos de interesse. Os computadores virtuais, denominados de instâncias, podem ser criados, lançados, parados e terminados conforme necessário, pagando por hora pelas máquinas que estão em uso.

3.2 – Amazon Elastic Compute Cloud (EC2)

O EC2 [21] é o segmento da plataforma AWS aonde os usuários irão alugar instâncias (máquinas virtuais) para a hospedagem dos processos. Para atender todos os perfis de usuário, a plataforma tem disponível três tipos distintos de instâncias e várias configurações diferentes para as máquinas em cada grupo.

As instâncias são divididas em Spot, Reservada e Dedicada. Cada uma delas possui públicos alvos diferentes. As instâncias Spot do Amazon EC2 [22] representam a capacidade computacional excedente da Nuvem AWS. Elas possuem um valor até 90% mais baixo em relação às Dedicadas. Em contrapartida, as máquinas podem ser interrompidas a qualquer momento com uma notificação prévia de dois minutos, caso algum cliente que usa o EC2 estiver precisando desse poder de processamento. Como consequência, as instâncias Spot são recomendadas para aplicativos que têm períodos de início e de término flexíveis, aplicativos que são viáveis somente por preços computacionais muito baixos e usuários com necessidades computacionais pontuais.

A respeito das instâncias reservadas [23], elas podem possuir um valor até 75% mais barato em relação às Dedicadas. Ao contrário das máquinas do tipo Spot, as reservadas não podem ser terminadas quando algum cliente do EC2 precisar do poder computacional alocado pelas mesmas. Elas não instâncias físicas, mas sim um desconto aplicado ao uso de instância dedicada. O usuário pode reservar determinados atributos de uma instância física e operar durante um determinado prazo de tempo. Após a expiração do tempo, caso haja necessidade de continuar utilizando, o preço a ser pago será de uma instância dedicada.

As instâncias Dedicadas referem-se a uma instância física totalmente dedicada a um único cliente. É recomendada para aplicações que não podem ser interrompidas, as Dedicadas são as mais caras, mas que trazem uma maior confiabilidade no quesito de estabilidade. As cobranças das máquinas, assim como na Spot e na Reservada, são feitas por segundo de uso. Vale ressaltar que há uma taxa extra de 2 (dois) dólares por hora caso tenha uma instância dedicada em uma das regiões disponíveis pela AWS.

As máquinas, além de possuírem diferentes tipos de alugueis, também possuem diversas famílias, aonde cada família é voltada para um uso específico de uso. Existem 5 famílias atualmente [24]. São elas: Uso Geral, Otimizadas para Computação, Otimizadas para memória, Computação acelerada, Otimizada para Armazenamento. As instâncias, da

mesma família, também podem possuir tamanhos diferentes que variam de nano, micro, small, medium, large, xlarge, 2xlarge etc.

A Figura 16 apresenta uma imagem retirada do próprio site da AWS referente a informações das máquinas do grupo “Otimizadas para memória”.

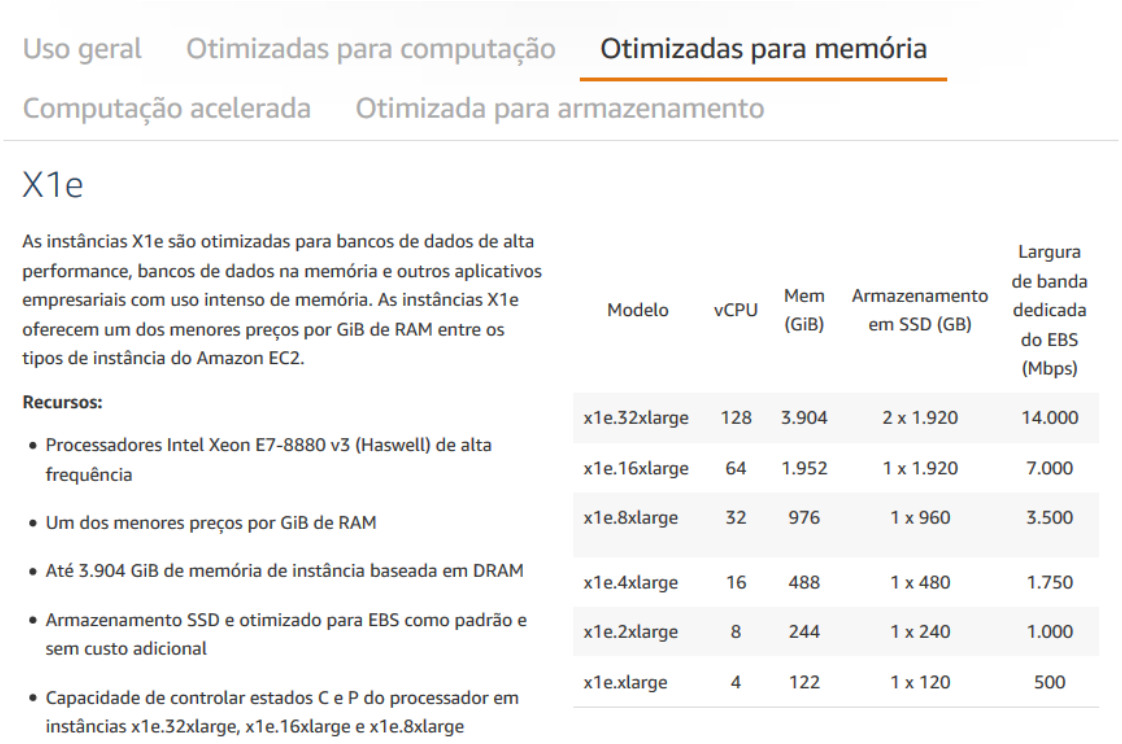


Figura 16 Instâncias da família “Otimizadas para memória”

A respeito do processo de aprovisionamento de uma máquina pelo serviço EC2, ao criar um cadastro em um plano gratuito (*free-tier*) ou pago no console da AWS, e entrar no painel do serviço EC2, tem-se acesso a algumas informações como as instâncias que estão rodando, os volumes extras adquiridos e outros dados interessantes como mostrado na Figura 17.

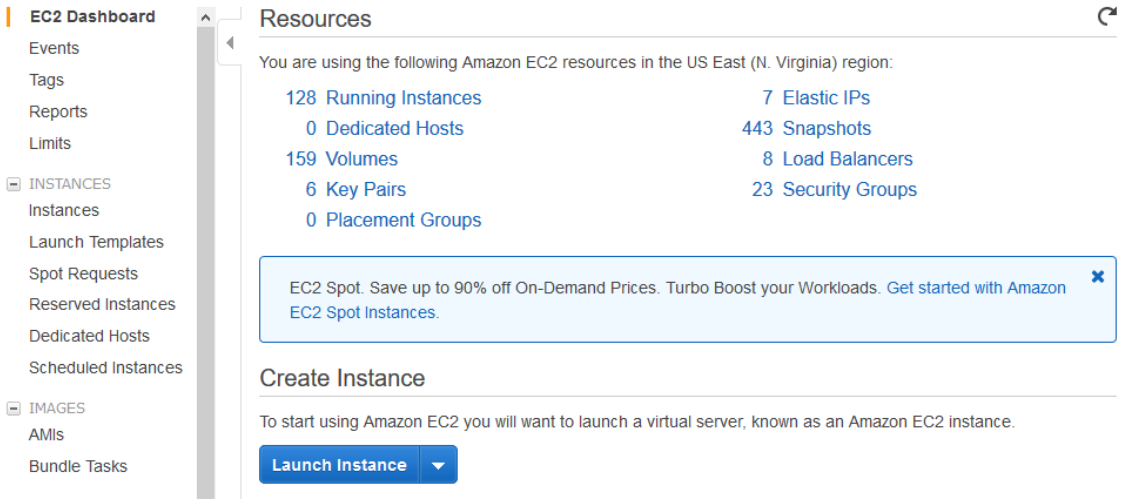


Figura 17 Painel EC2 da AWS

Ao clicar em "Launch Instance", nas próximas janelas será possível manipular o sistema operacional da máquina, o volume *root*, configurações de acesso, bloquear ou permitir portas de conexão entre outros fatores. Além disso, um serviço que AWS disponibiliza para acelerar a produtividade é o lançamento de uma instância já com uma pré-configuração definida, isto é, softwares instalados e já configurados. Para isso, durante as etapas para iniciar uma nova máquina, haverá uma opção para informar se deseja inserir uma *Amazon Machine Image* (*AMI*). Existem tanto imagens gratuitas quanto pagas. É aconselhável ir nas seções "AWS Marketplace" e "Community AMIs" e procurar por palavras chaves contendo nome dos softwares que pretende utilizar após o aluguel da máquina.

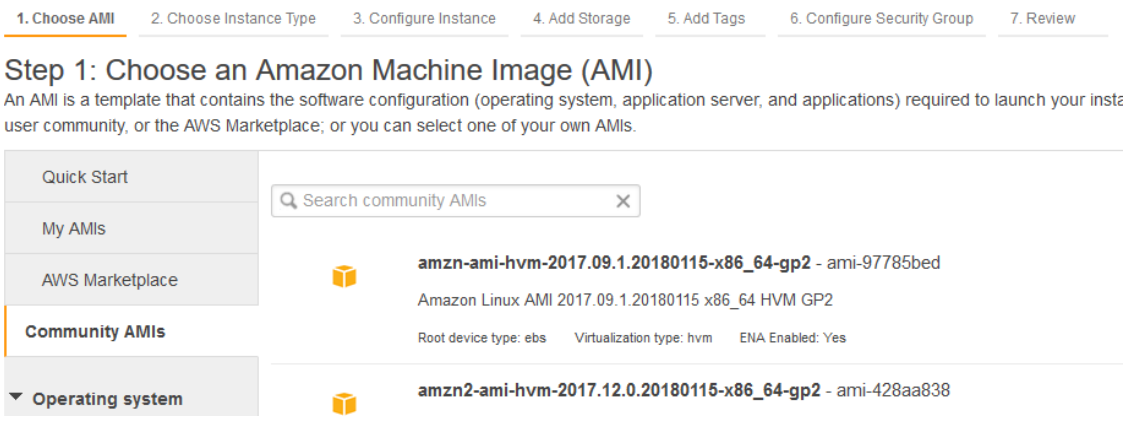


Figura 18 Painel de AMIs criadas pela comunidade

3.3 – Amazon Simple Queue Service (SQS)

O serviço SQS da AWS oferece hospedagem de filas altamente escalável para armazenamento de mensagens, com tamanho máximo de 256 kb cada, na medida em que trafegam entre aplicativos ou micro serviços. Movimenta dados entre os componentes distribuídos do aplicativo e ajuda a desacoplá-los. Este é um papel fundamental para permitir escalonar serviços menores que compõe um projeto grande e de alta complexidade.

O serviço possui dois tipos de filas. A fila FIFO (*First-In-First-Out*) e a *Standard* (Padrão). Há duas diferenças entre estes tipos de filas que são: Ordenamento da Entrega e a Duplicidade de Processamento.

Na FIFO, a ordem que as mensagens são enviadas e recebidas é preservada. Ademais, garante que uma mensagem seja entregue uma única vez, permanecendo disponível até que o consumidor a processe e a exclua. Não existem duplicações introduzidas neste tipo de fila.

A respeito do tipo Standard, as mensagens poderão ser entregues em uma ordem diferente da qual elas foram enviadas. Segundo a questão de duplicidade, podem existir um número infinito de cópias de mensagens dentro da mesma fila, atribuindo a responsabilidade do usuário de criar uma lógica dentro de sua aplicação para realizar o tratamento dessas mensagens repetidas.

O custo para a criação de fila é gratuito. O valor do serviço SQS é cobrado por número de solicitações (primeiro 1 milhão é gratuito) enviadas e também por gigabytes de transferência de dados caso seja para uma instância EC2 de uma região distinta da qual a fila foi criada. O tipo *first-in-first-out*, por possuir os benefícios de sem duplicidade e a entrega de forma ordenada, possui o dobro do custo em relação a fila padrão.

Capítulo 4

Machine Learning

4.1 - Introdução

O *Machine Learning* (Aprendizado de Máquina) é uma subárea da Inteligência Artificial com o foco em elaborar modelos que possam aprender através da bagagem adquirida ao longo de toda a experiência de treinamento. O aprendizado ocorre por meio da utilização de algoritmos dedutivos que, fundamentados em conceitos matemáticos, extraem padrões que possam existir em uma massa de dados.

Apesar de aparentar ser um assunto recente da área da ciência da computação, as primeiras menções a este subcampo e também da Inteligência Artificial ocorreram na década de 1950, sendo Arthur Samuel [28] o primeiro que temos conhecimento de escrever um programa de ordenação com a habilidade de aprendizagem.

A tecnologia, assim como todas as outras, passou por momentos de crescimento exponencial e também de fases aonde os investimentos nesta área foram reduzidos o que diminuiu todo o prestígio e glamour presente naquela época. Mais tarde, com a descoberta de novos métodos de aprendizado de máquina e de meios para contornar problemas que antes não eram solucionáveis, a área da Inteligência Artificial como um todo obteve o prestígio de volta.

Nos primeiros anos do século 21 até os dias atuais, com o avanço da popularização da Internet entre as pessoas de todas as camadas sociais, muito se falam a respeito das subáreas da Inteligência Artificial. Muitas pessoas conectadas, muitos dados sendo trafegados, consequentemente uma grande demanda é requisitada para atender problemas na área de privacidade, entretenimento e todas as outras.

Hoje, podemos não perceber, mas existem algoritmos de aprendizado de máquina sendo aplicados a todo momento na maioria das tarefas relacionadas ao uso de dispositivos tecnológicos. Ao entrar na caixa de entrada do endereço eletrônico, há *Machine Learning* sendo aplicado para categorizar algumas mensagens como spam, promoções, fóruns etc. Quando visita uma rede social, todo o feed de notícias foi

processado por algoritmos que procuram identificar quais são as melhores notícias para serem exibidas. Os algoritmos estão em constante aprendizado a fim de entregar o melhor resultado possível em qualquer intervalo de tempo.

A empresa de consultoria *Gartner Group*, além de desenvolver tecnologias relacionadas a introspecção necessária para seus clientes tomarem suas decisões todos os dias, produz anualmente uma apresentação gráfica bastante conceituada chamada "*Gartner Hype Cycle for Emerging Technologies"* [11]. Trata-se de um gráfico que representa a maturidade, adoção e aplicação social de tecnologias emergentes para um determinado ano.

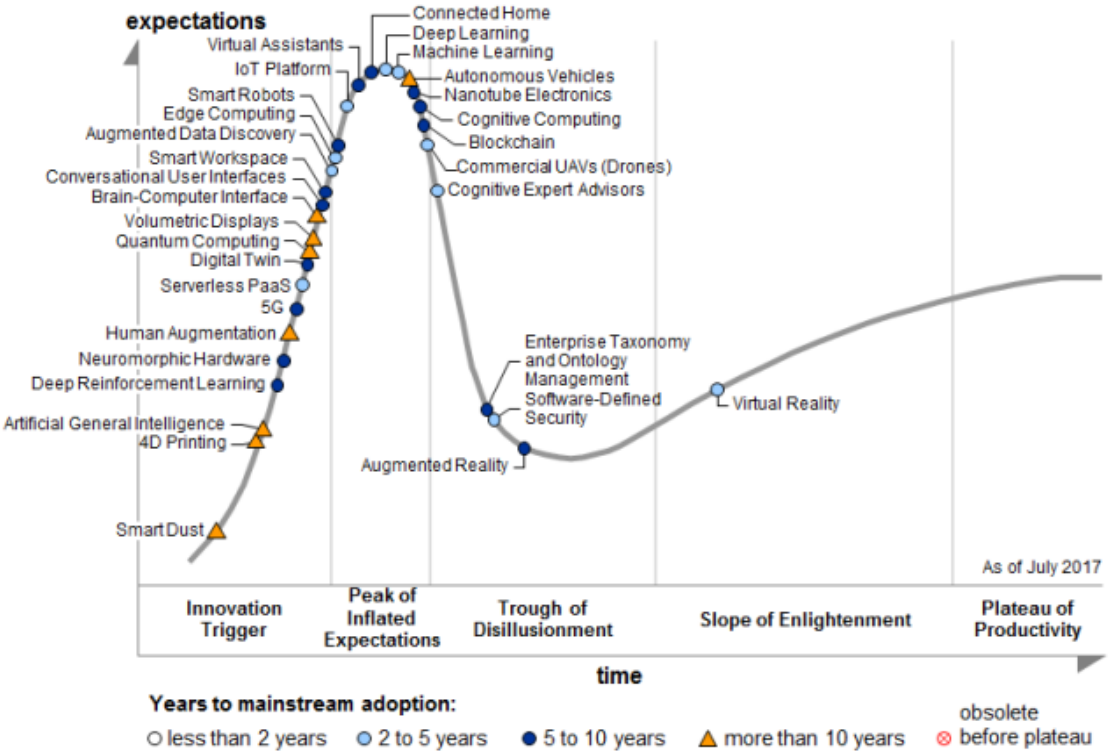


Figura 19 Gráfico de 2017 do Gartner Hype Cycle for Emerging Technologies

A Figura 19 apresenta que em 2017 foi o ápice de temas relacionados a Aprendizado de Máquina, desde da sua adoção ao mercado à divulgação de estudos relevantes para atrair o interesse das empresas e das pessoas de forma geral.

4.2 - Long Short-Term Memory

Os modelos computacionais inspirados pelo funcionamento do sistema nervoso central são conhecidos como Redes Neurais. Um dos tipos de redes neurais que existe é a *Long Short-Term Memory*, mais conhecida como LSTM. É um tipo de rede neural recorrente que pode aprender a depender da ordem dos itens em uma determinada sequência. O fato de ser "recorrente" significa que este modelo utiliza a própria saída de um determinado período como entrada para uma próxima sequência de amostras, ou seja, possui habilidade de armazenar memória.

A LSTM é bastante utilizada quando o desafio está relacionado com aprendizagem do passado para prever o presente e o futuro. Alguns exemplos bem comuns são: a previsão da cifra de uma determinada ação na bolsa de valores, previsão de chuva dada diversas *features*, previsão da próxima palavra ao digitar um conjunto de termos etc.

O modelo padrão de uma simples rede neural recorrente é apresentado na Figura 20:

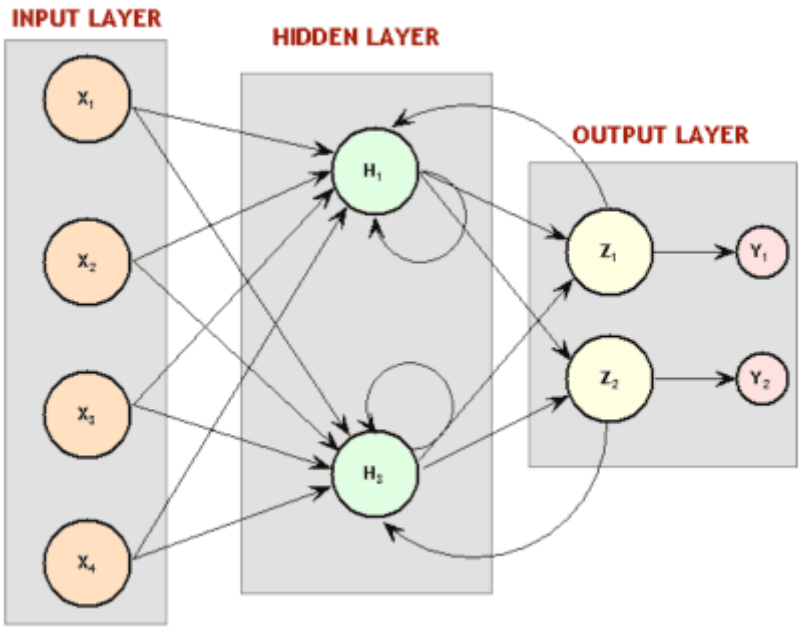


Figura 20 Modelo padrão de uma rede neural recorrente [17]

Este tipo de modelo não é *Feed-Forward*, o que significa que os sinais não trafegam em uma única direção, ou seja, da entrada para a saída. Essa rede possui sinais em ambos os sentidos, introduzindo a lógica de loop (memória) para realizar o

aprendizado e aumentar a precisão dado os cálculos produzidos por uma etapa em um determinado instante de tempo.

O ponto que possui o maior destaque em redes neurais recorrentes (RNN) é de ser capaz de conectar informações anteriores à tarefa atual, porém não é sempre uma tarefa eficaz. Às vezes, o desafio está em apenar observar os recentes dados para executar um serviço atual. Para ficar mais claro, podemos imaginar o ato de prever a última palavra dada uma frase. Como não há necessidade de um contexto, e o espaço entre a informação relevante e a palavra a ser prevista é pequeno, as RNNs podem aprender com uma precisão bastante satisfatória.

Mas também há casos em que se precisa de todo um contexto. Imagine um texto aonde no seu início é informado que a pessoa nasceu no Brasil. É relatado momentos da infância, adolescência, experiências profissionais e no final de toda essa apresentação a seguinte sentença é abordada: "Eu falo português com fluência". O ser humano lendo todo o texto, possui claro entendimento a respeito da pessoa saber, com fluência, o idioma português por ser nativo do país aonde nasceu. Mas a questão é como a RNN poderá adquirir este tipo de habilidade.

À medida que o espaço da informação relevante aumenta, as RNNs não conseguem aprender a conectar a informação, neste caso, da pessoa nascer no Brasil e saber falar o idioma português com fluência. A fim de suprir este problema encontrado no modelo padrão de RNN é que foi desenvolvida a LSTM, em português, as redes de memória de longo prazo.

A LSTM é um tipo especial de RNN capaz de aprender dependências de longo prazo. Elas foram introduzidas por Hochreiter (1991) [12] e Bengio, et al. (1994) [13], funcionando muito bem em uma grande variedade de problemas, e agora são amplamente utilizadas.

Estas redes foram projetadas para saberem lidar com os problemas que as RNNs padrões não conseguem. O módulo que está diretamente relacionado com o loop existentes em modelos que não são *Feed-Forwards*, tem uma estrutura diferente e mais complexa.

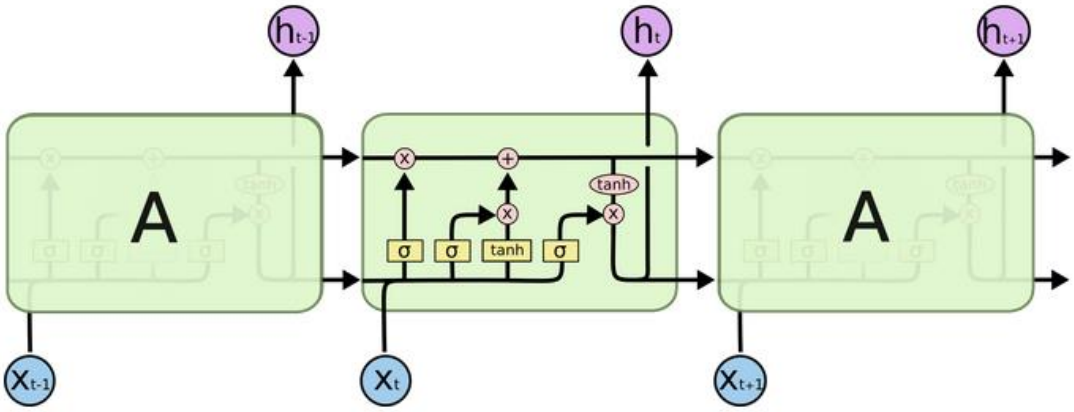


Figura 21 Modelo padrão de uma rede LSTM [18]

Na figura acima, cada linha carrega um vetor inteiro de informação, desde a saída de um nó até as entradas dos demais. Os círculos de coloração rosada representam operações pontuais que são executadas e, em relação as caixas de cor amarela, são camadas de redes neurais aprendidas. As linhas de fusão denotam concatenação, enquanto uma linha de bifurcação denote seu conteúdo sendo copiado e as cópias indo para diferentes locais.

O *core*, por trás das redes LSTMs, trata-se do estado de cada célula que é apresentado pela linha horizontal que passa pela parte superior do diagrama da Figura 22.

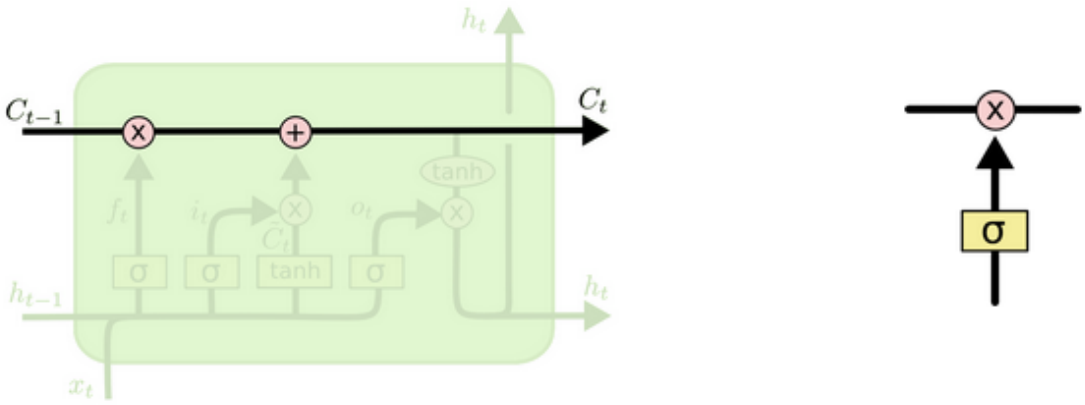


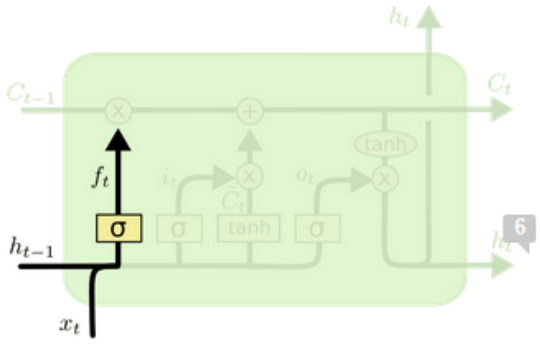
Figura 22 Estrutura que armazena o estado da célula [18]

O estado da célula percorre todo o trajeto com apenas algumas pequenas interações lineares tornando a fluidez da informação praticamente inalterada. A estrutura responsável por remover ou adicionar informações ao estado da célula é chamada de *Gates*. Os *Gates* são compostos de uma camada de rede neural sigmoide e de uma

operação de multiplicação. Ao todo são três *gates* para proteger e controlar os estados das células. São eles: *input gate, output gate e forget gate*.

A camada sigmoide, representada pela estrutura da direita na Figura 22, resulta um número entre 0 e 1 descrevendo quanto de cada componente deve ser barrado ou passado para o próximo estado. O valor 1 significa permitir a passagem de todas as informações de uma *feature* em específico.

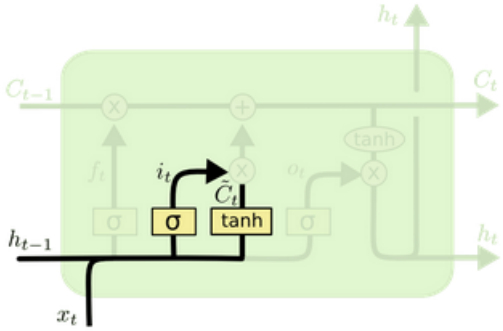
O primeiro passo da LSTM, assim como qualquer projeto que está sendo projetado, é decidir quais informações devem seguir adiante e quais devem ser afastados do estado da célula. Esta escolha é executada pela camada sigmoide denominada *forget gate layer*.



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

Figura 23 Primeiro passo de uma rede LSTM [18]

Tem como entrada h_{t-1} e x_t obtendo como saída novamente um valor entre 0 e 1, aonde 1 significa para manter todas as informações ali presentes. A camada *input gate* decide quais valores devem ser atualizados. Em seguida, a camada tangente hiperbólica (tanh) cria um vetor de novos possíveis valores, $\tilde{C}_t$ que podem ser adicionados ao estado. A combinação desses dois vetores irá gerar uma nova atualização para o estado.

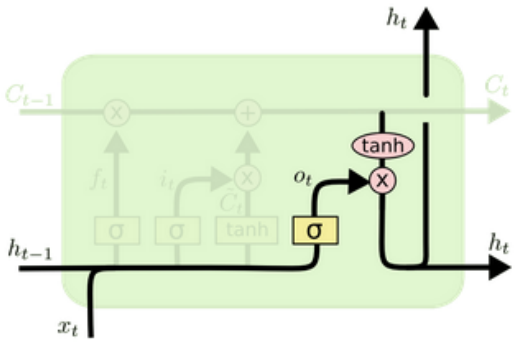


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 24 Segundo passo de uma rede LSTM [18]

Após a decisão de quais informações devem ser trabalhadas, o segundo passo possui a função de executar a atualização do estado da célula anterior C_{t-1} para C_t. O estado antigo é multiplicado por f_t descrito na Figura 23, esquecendo as informações que foram decididas anteriormente. Então, são somados ao produto vetorial i_t e $\tilde{C}_t$. Os novos valores obtidos são candidatos, escalados o quanto necessário para atualizar cada valor do estado.

Após decidir quais novas informações vão ser armazenadas e feito atualização do estado da célula, chega o momento de decidir o que será lançado. A saída é baseada em uma versão filtrada do estado da célula.



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

Figura 25 Terceiro passo de uma rede LSTM [18]

Os valores são colocados entre -1 e 1 e multiplicados pela saída do portão sigmoide, de modo que apenas contenha as partes que foram decididas pelo algoritmo.

O modelo LSTM apresentado nesta seção é simples quanto comparado a outros existentes cujo, o número de operações matemáticas é maior para melhorar ainda mais a

sua eficácia e poder de processamento. Cientistas como Kalchbrenner, et al. (2015) [14], Bayer & Osendorfer (2015) [15] criaram modelos de LSTM diferentes daquele apresentado nesta seção.

Capítulo 5

Projeto Lunar

O projeto foi implementado visando torná-lo escalável. As principais etapas foram divididas em pequenos processos distintos para facilitar a manutenção, além de diminuir a complexidade do código como um todo. A Figura 26 mostra como está estruturado a aplicação Lunar no quesito do envio de dados pelo usuário até a apresentação dos registros do banco de dados através da Web API.

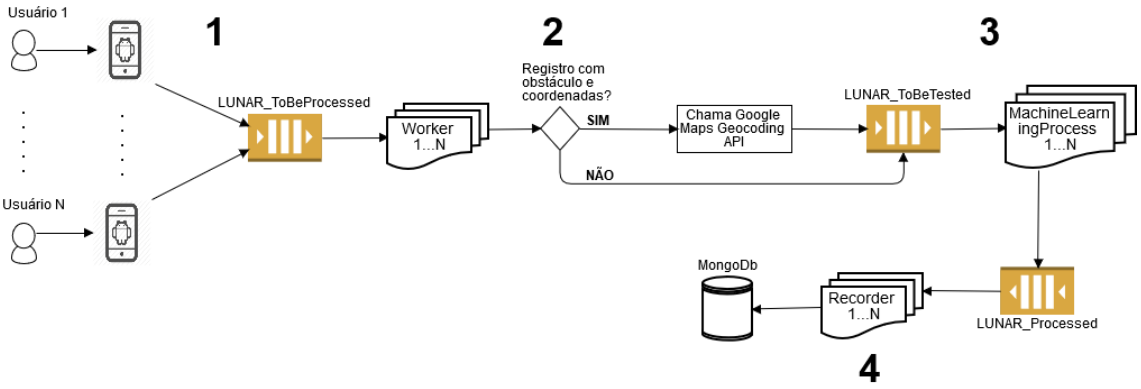


Figura 26 Estrutura global do projeto Lunar

A estruturação do projeto é fracionada em quatro principais etapas. A primeira delas refere-se à comunicação do aplicativo Android com uma *standard queue* chamada "LUNAR_ToBeProcessed". É nesta fase que os dados capturados pelo smartphone serão enviados para a fila aonde serão consumidos para a realização de validações e filtros. A transmissão de dados entre essas duas ferramentas está apresentada na figura a seguir.

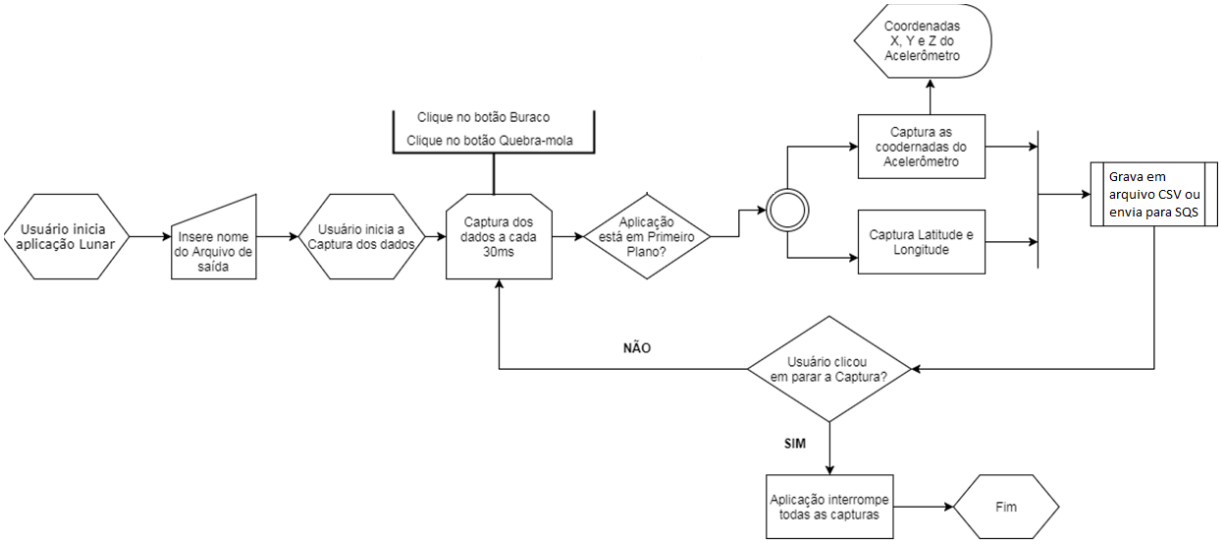


Figura 27 Fluxograma do funcionamento do aplicativo Android

O processo relacionado a captura dos dados dos sensores ocorre a cada 30 milissegundos, isto é, a cada 50 centímetros percorridos pelo veículo caso a velocidade instantânea seja de aproximadamente 60 km/h. As informações cruas são enviadas em formato JSON para a fila “LUNAR_ToBeProcessed”.

A segunda etapa é o estágio responsável pelo processamento tanto no quesito de extrair mais informações a partir das coordenadas do GPS como também para realizar as validações com o objetivo de remover os dados caracterizados como inválidos. Como esta parte do projeto depende exclusivamente da quantidade de dados armazenada em uma fila da AWS, ela foi estruturada de modo que tornasse viável a escalabilidade, podendo, portanto, executar a quantidade de processos o quanto for necessário para atender a demanda da entrada. Após o término do processamento, os registros são inseridos na fila “LUNAR_ToBeTested”.

A terceira etapa é a fase aonde há o uso do algoritmo de aprendizado de máquina. Cada mensagem contida na fila será lida, aplicada no modelo de identificação automatizada, deletada da fila de entrada e inserida na “LUNAR_Processed”. Este componente da aplicação Lunar também é escalável. Caso seja notado um acúmulo na “LUNAR_ToBeTested”, será executado mais instâncias do processo “MachineLearningProcess”.

O sistema referente à quarta etapa é hospedado em uma máquina distinta daquela usufruída pelos processos responsáveis do tratamento e da aquisição de mais dados. É o

segmento do projeto aonde ocorre somente a leitura das mensagens hospedadas na fila “LUNAR_Processed” e o envio das mesmas para uma coleção do banco de dados não relacional MongoDb.

Após a informação percorrer as quatro etapas, ela estará disponível para ser requisitada pelo usuário através de uma requisição HTTP do tipo GET. A construção de uma Web API foi desenvolvida para facilitar a comunicação com os registros presentes na base de dados.

5.1 – Tipos de Dados

Anterior a etapa da aquisição dos dados, é importante entender quais tipos de dados são necessários para que torne plausível o objetivo principal e, se possível, também o encaminhamento das metas a serem cumpridas a longo prazo que foram descritas na seção “Trabalhos Futuros”. Após ter todos os objetivos bem definidos, torna-se fácil a tarefa de identificar quais dados devem ser capturados.

A respeito da detecção de obstáculos nas vias pavimentadas, foi elaborada uma análise prévia para verificar se o sensor Acelerômetro presente em dispositivos mobile, seria sensível o suficiente para registrar as diferenças sutis das oscilações provocadas pela via no automóvel. Observada a viabilidade, a conclusão foi de capturar os dados gerados pelo sensor Acelerômetro.

Em relação ao mapeamento dos dados, é viável se e somente se forem capturadas informações relacionadas ao GPS. Para isso, no código desenvolvido para o aplicativo Android, foi desenvolvida a lógica para salvar as informações referentes a latitude e a longitude em tempo real.

Além dos dados já mencionados, outros, não menos importantes, são capturados como: a inclinação em graus do dispositivo no momento da aquisição; tempo em milissegundos desde que foi iniciada a aplicação; data e hora corrente. A informação referente à inclinação é para possibilitar novos estudos diferentes daqueles realizados neste projeto. Já sobre o cronômetro e hora que está marcando no aparelho, servirá como auxílio para a realização do processamento e exposição dos dados.

5.2 – Método de Captura

A construção de uma aplicação teste para smartphone Android foi o primeiro passo para tornar possível a captura dos dados. Antes de realizar a aquisição, foi estudado a necessidade de incluir mais equipamentos tecnológicos para poder realizar o treinamento do modelo de *machine learning*.

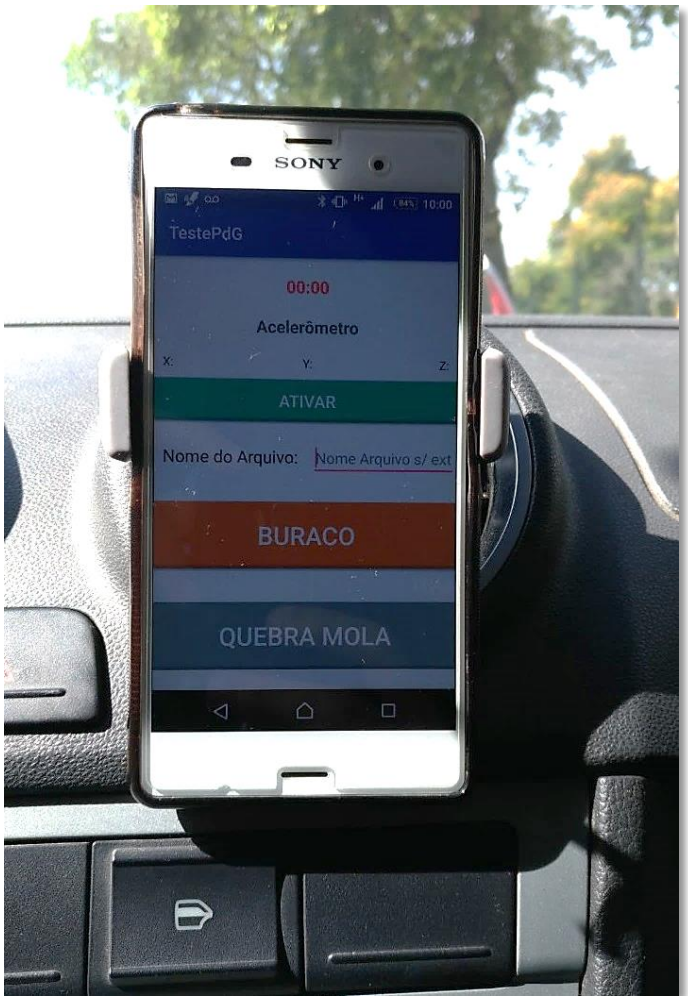


Figura 28 Aplicação teste já posicionada para iniciar a captura

A aplicação teste além de contar com o botão “Ativar”, também possui os botões de “BURACO” e “QUEBRA MOLA” para registrar os eventos dos obstáculos de forma manual. Um suporte de celular foi utilizado para diminuir a variação da inclinação do dispositivo e também para manter um padrão a ser seguido na aquisição dos dados.

Um outro auxílio muito importante para realizar a validação dos eventos registrados foi a utilização de uma câmera no capô do automóvel para produzir a gravação de todo o trajeto percorrido a fim de possibilitar a validação dos dados.

Figura 29 Uso de uma filmadora no capô do veículo

A colaboração conjunta de uma câmera filmando o trajeto e os eventos serem inseridos de forma manual, possibilitou uma análise visual para checar a viabilidade de desenvolver um algoritmo de aprendizagem de máquina que detecte, de forma automatizada, os obstáculos nas vias pavimentadas.

A respeito da base de dados utilizada para os testes, o automóvel percorreu vias de diferentes pontos da cidade e do estado do Rio de Janeiro, como a Cidade Universitária (UFRJ), Aterro, Botafogo, Humaitá, Glória e Lagoa. Foram capturados quebra-molas de diferentes tipos e buracos de baixa, média e alta profundidade.

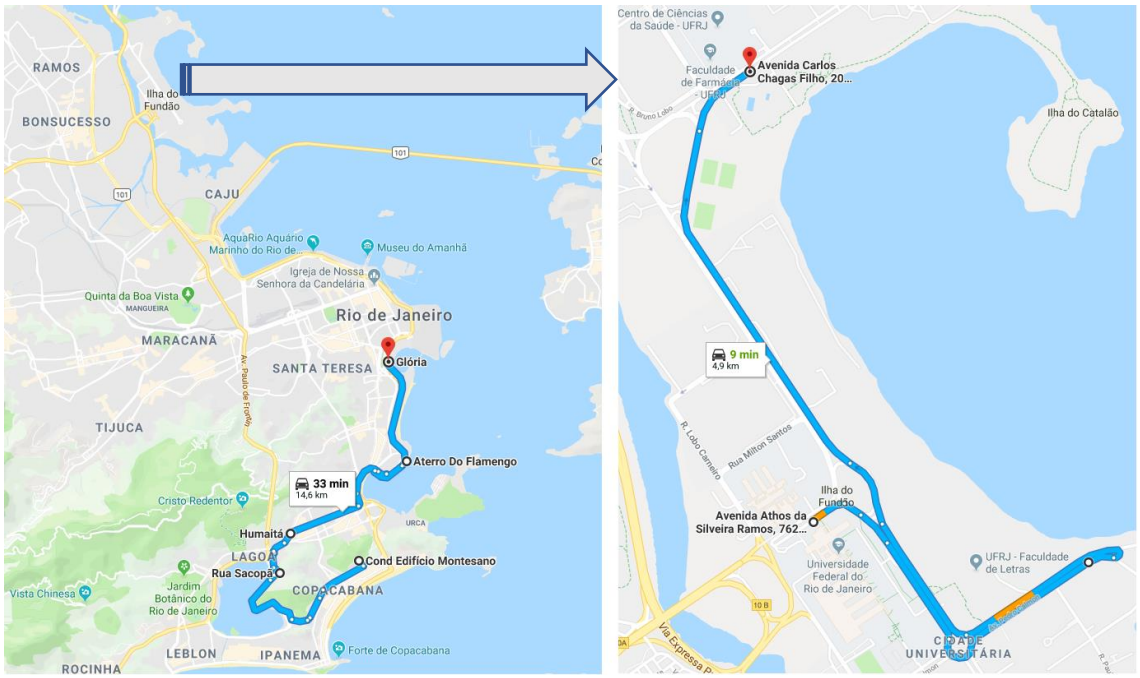


Figura 30 Trajeto percorrido na cidade do Rio de Janeiro

Um outro percurso realizado para aumentar a quantidade de obstáculos na base dados foi no distrito de Bacaxá pertencente a cidade de Saquarema/RJ. A Figura 31 ilustra o caminho percorrido.

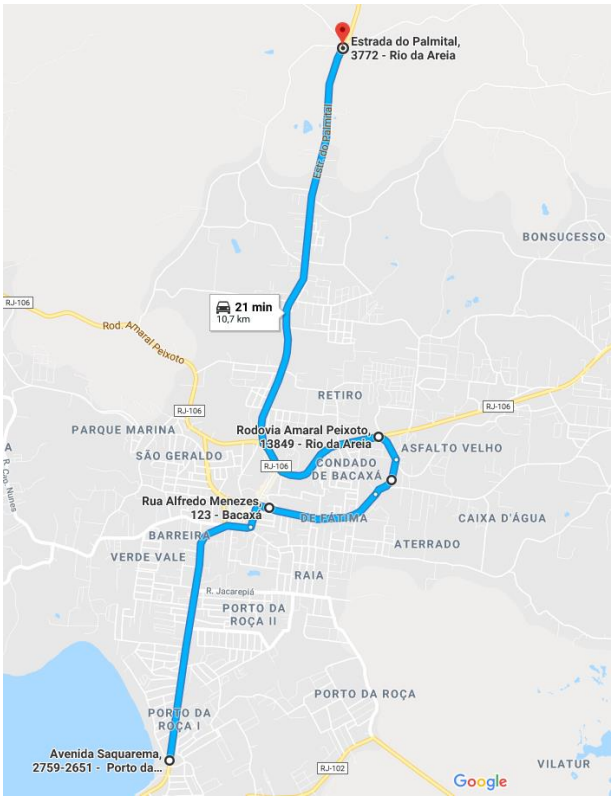


Figura 31 Trajeto percorrido no distrito de Bacaxá, Saquarema/RJ

Após a realização desses trajetos, foi efetuado um experimento como todos os dados compilados em uma única base de informações. Como os obstáculos variam consideravelmente entre percursos diferentes, foram aplicados testes individuais com o objetivo de selecionar um em específico para servir como base de estudo para os demais filtros e novos experimentos. A trajetória escolhida foi da Cidade Universitária – UFRJ, em razão do conhecimento da localização de cada obstáculo e das imperfeições ao longo do trajeto.

5.3 – Análise Diagnóstica

Após realizada a captura dos dados, inicia-se a elaboração de uma análise diagnóstica a partir dos gráficos gerados pelos valores dos três eixos do sensor acelerômetro em cada instante do tempo. É realizado uma busca para identificar as relações de causa e consequência percebidas ao longo das amostras para entender o porquê de ter ocorrido picos e depressões em certos momentos da coleta.

Anterior a geração dos gráficos, foi realizada uma normalização em todo o conjunto das amostras utilizadas com o intuito de reduzir o ruído e de deixar mais nítido as consequências geradas após o automóvel sofrer o impacto de um certo tipo de obstáculo.

As imagens a seguir apresentam o comportamento do sensor acelerômetro no eixo x, y e z em um trajeto percorrido na Cidade Universitária do Rio de Janeiro, bem como as marcações de buraco e quebra-molas inseridas no momento da captura.

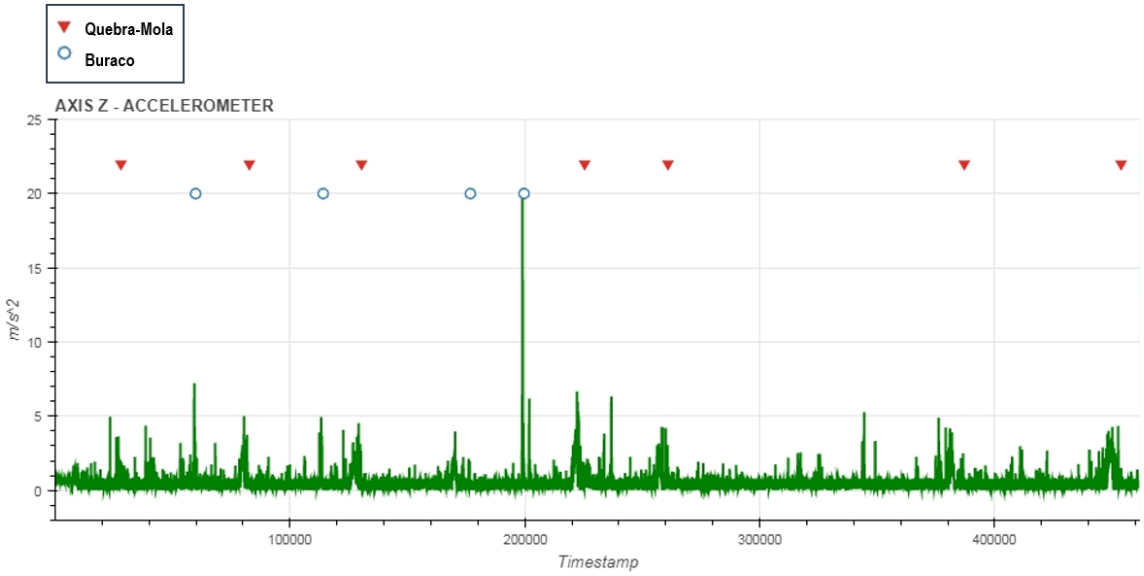


Figura 32 Comportamento do eixo Z do sensor acelerômetro

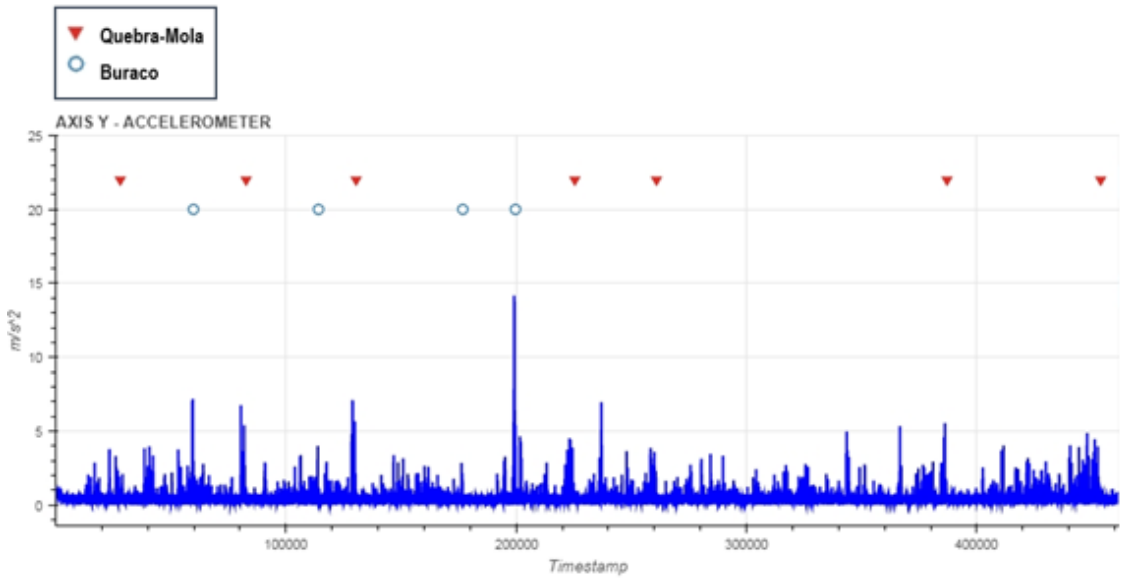


Figura 33 Comportamento do eixo Y do sensor acelerômetro

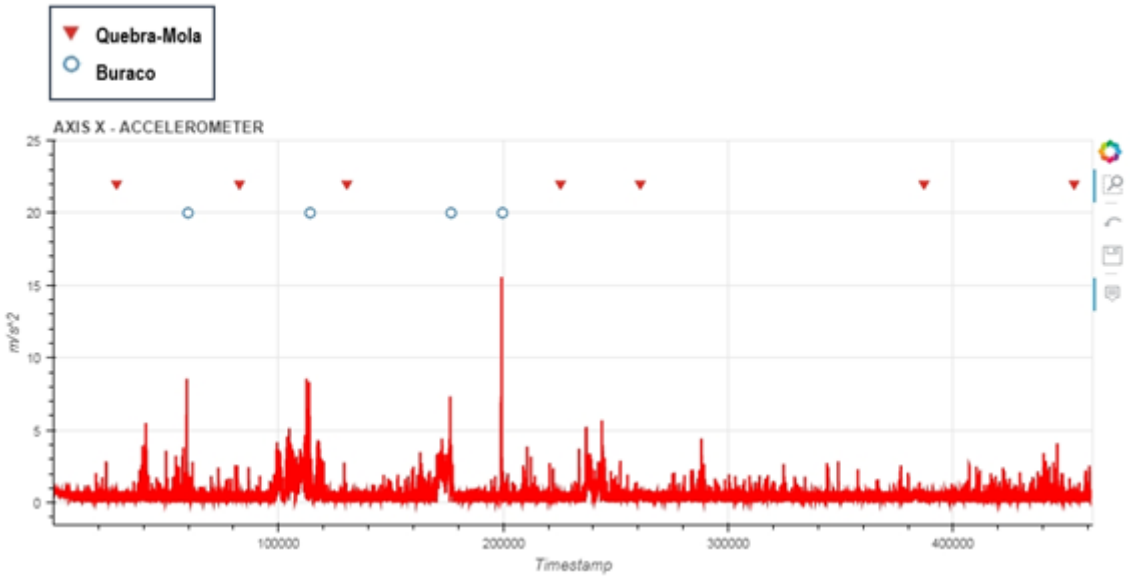


Figura 34 Comportamento do eixo X do sensor acelerômetro

As consequências geradas pelos eventos se tornam nítidas no conjunto de dados normalizados. Os picos com as intensidades diferentes estão diretamente relacionados com a força do impacto que cada evento foi capaz de gerar no automóvel. Isto vale também para as reações distintas entre cada eixo do sensor. Dependendo de como o automóvel passou por cima de um obstáculo, as assinaturas em cada eixo tendem a apresentar valores diferentes. Os picos menores, mas que não podem ser ignorados, foram provocados por imperfeições existentes na via percorrida.

O algoritmo de aprendizagem de máquina deve ser capaz de saber diferençar os picos ocasionados por eventos de interesse e aqueles ocorridos por ondulações da pista. Uma das técnicas possíveis é a inserção de um *threshold* para realizar essa distinção.

A Figura 35 apresenta um caso que é utilizado a imagem fornecida pela câmera presa ao capô do carro e a assinatura provocada pelo evento destacado em um dos gráficos gerados.

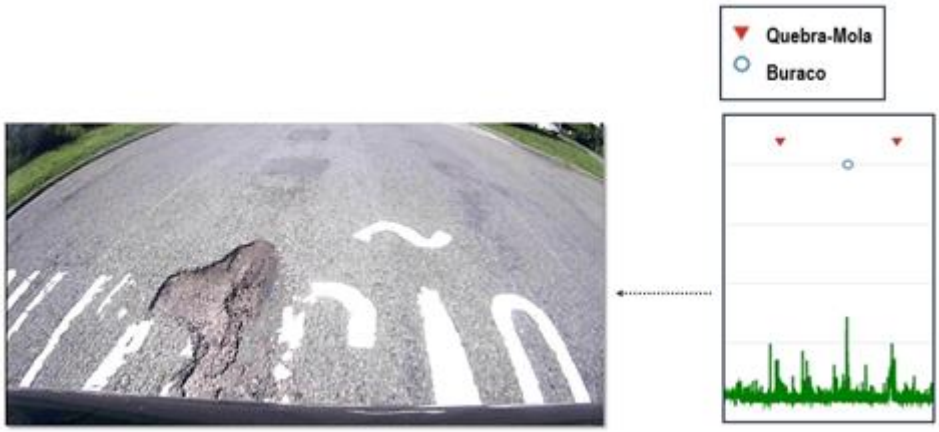


Figura 35 Relação de causa-consequência de um buraco captado pela filmadora

O ícone triangular significa um evento de quebra-mola, já o círculo de um buraco. O pico de maior intensidade foi gerado pelo buraco presente na imagem da esquerda na Figura 36. Entre o primeiro quebra-mola e buraco, foi constatado um pico significativo. Este surto está relacionado com uma faixa em alto relevo existente assim que o veículo entra em uma rotatória, momentos antes da imagem capturada e apresentada acima. A faixa contém desgastes, desníveis e esses detalhes foram capturados com sucesso pelo sensor presente no dispositivo móvel utilizado.

5.4 - Processamento dos Dados

O processamento dos dados foi dividido em duas partes. A primeira parte é por processos denominados *Worker* desenvolvidos em C#. Já a segunda foi através de algoritmos em Python para uso exclusivo do código de aprendizado de máquina. As seções a seguir abordarão como foram feitas as mudanças nos dados crus.

5.4.1 – Processo Worker

O Worker é um dos projetos desenvolvidos em C# para executar o processamento dos dados antes de serem enviados para o algoritmo de aprendizado de máquina. Este processo está hospedado na nuvem em uma máquina de modelo “t2.micro”.

Os dados crus gerados pelo aplicativo mobile são enviados para uma fila na AWS chamada "LUNAR_ToBeProcessed”. O Worker captura, de tempos em tempos, as mensagens disponíveis e aplica uma série de filtros.

O primeiro filtro aplicado é relacionado ao problema da captura das coordenadas do GPS. É de se esperar que existem muitas áreas com sinais precários ou até mesmo inexistentes. Em virtude desse acontecimento, alguns dados são enviados para a fila contendo “0.0” como latitude e longitude. Como um dos objetivos principais da aplicação é possibilitar o mapeamento dos obstáculos nas vias pavimentadas, sem as coordenadas do GPS não é possível tal feito. Portanto, dados que não registram corretamente informações do GPS são excluídos pelo filtro inicial.

Além da importância em realizar a aquisição correta das coordenadas, um outro fator importante para termos uma detecção automática satisfatória é o comportamento do dispositivo móvel ao longo do percurso percorrido. Os atos de mexer no celular, alteram a posição e a inclinação, o que influenciam e muito no resultado final. Diante disso, há a necessidade de aplicar um filtro que delete o registro caso não siga alguns comportamentos esperados. Como a captura dos dados vista na seção 5.2 foi realizada com o celular na vertical, o limite seguro estabelecido foi de 40 graus com o eixo horizontal.

Caso o dado que está sendo processado no momento possuir uma *Output* diferente de 0, isto é, algum evento de obstáculo, é feita uma requisição a um Web Service chamado Geocoding API [16] pertencente à empresa Google. Esta API, em específica, fornece diversos atributos relacionados ao local oriundo de uma determinada latitude e longitude.

A URL a ser utilizada para esta biblioteca é do seguinte formato:

https://maps.googleapis.com/maps/api/geocode/json?latlng={0},{1}&key={2}&language=pt-BR

Os campos {0} e {1} são preenchidos com as coordenadas da latitude e longitude com a vírgula de delimitador. O último campo obrigatório é o {2} aonde é inserida a chave de acesso que foi adquirida ao fazer uma conta no site *Google Cloud Platform*. Uma pequena parte do resultado de uma amostra para a latitude -22.921493 e -43.235691 pode ser vista na Figura 36.

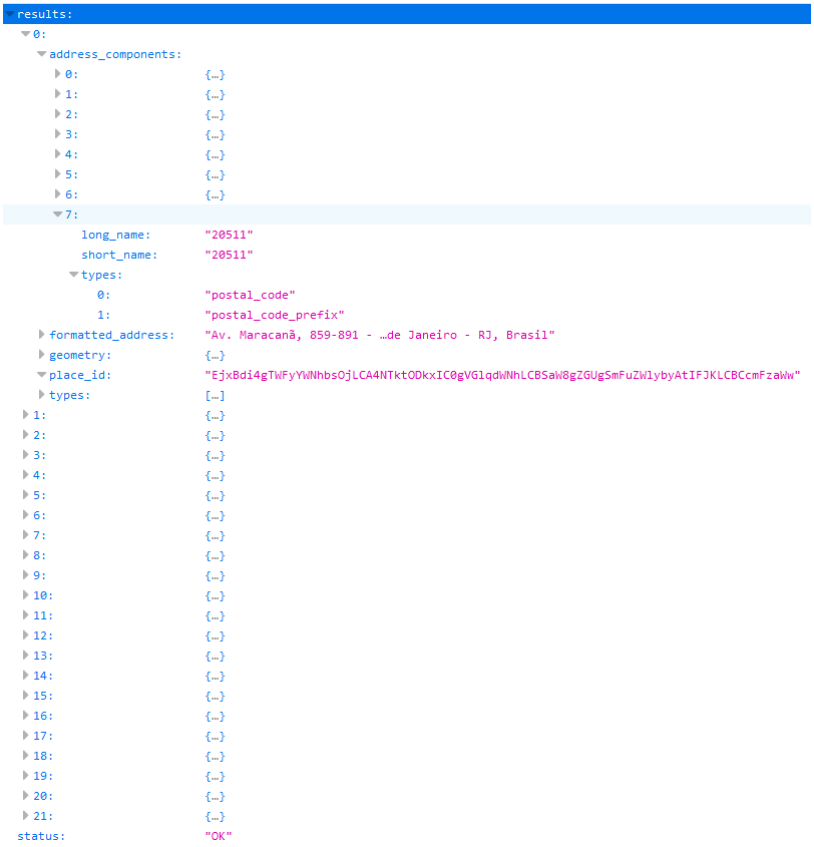


Figura 36 Exemplo da resposta da Geocoding API

A API retorna diversos dados relacionados com as coordenadas informadas. Os dados estão apresentados na forma decrescente em relevância. Isto significa que as informações contidas no primeiro item da lista são as mais precisas.

O campo "*address_components"* apresenta uma lista começando desde das informações mais específicas referentes àquele endereço até dados genéricos inerentes ao estado que pertence.

A fim de registrar somente os dados de maior relevância, a aplicação captura somente dados do primeiro item da resposta, ignorando as demais informações. Dados como rua, cidade, estado serão inseridos no objeto a ser enviado para a fila "LUNAR_ ToBeTested" para ser persistido até o momento de o registro ser adicionado ao banco de dados.

5.4.2 – Processo Machine Learning

5.4.2.1 – Fase de Teste

Um outro tratamento totalmente diferente é inerente ao pré-processamento dos dados para obter um melhor resultado no algoritmo de *machine learning*.

Durante a fase de uso da aplicação teste, os dados são salvos em três arquivos diferentes de extensão csv. O primeiro é um arquivo genérico possuindo registros a cada 30 milissegundos de todo o trajeto que o automóvel percorreu, contendo informações do sensor acelerômetro, latitude, longitude, inclinação do dispositivo, tempo do cronômetro e tempo real. O segundo e terceiro arquivo possuem estrutura semelhante, porém somente de registros de eventos que foram inseridos de forma manual pelo usuário. Segue abaixo parte das amostras geradas pelo arquivo principal cujo *output* vale zero.

accelerometer_X	accelerometer_Y	accelerometer_Z	latitude	longitude	tilt	timestamp	datetime
1.4174107	5.523592	7.3839436	-22.9236388	-43.233107	38	91	2018-03-15T10:27:11.907Z
-0.32562137	5.2410674	7.2546525	-22.9236388	-43.233107	36	209	2018-03-15T10:27:12.025Z
0.6153286	7.0295906	7.1660647	-22.9236388	-43.233107	45	389	2018-03-15T10:27:12.205Z
0.8284191	7.745479	7.755056	-22.9236388	-43.233107	45	576	2018-03-15T10:27:12.393Z
0.46927786	6.0599093	8.595447	-22.9236388	-43.233107	35	753	2018-03-15T10:27:12.570Z
0.1699935	5.5786605	6.9505796	-22.9236388	-43.233107	39	936	2018-03-15T10:27:12.752Z
0.44054657	7.8460383	8.152506	-22.9236388	-43.233107	44	1110	2018-03-15T10:27:12.926Z
1.1516461	5.5020437	8.2339115	-22.8589141	-43.2308922	34	1288	2018-03-15T10:27:13.104Z
1.0917894	5.9808984	6.3807425	-22.8589141	-43.2308922	44	1475	2018-03-15T10:27:13.292Z
0.102953814	8.051947	7.075082	-22.8589141	-43.2308922	49	1655	2018-03-15T10:27:13.471Z
-0.0670397	6.0670924	7.355212	-22.8589141	-43.2308922	40	1835	2018-03-15T10:27:13.651Z
0.9624985	7.501263	8.877971	-22.8589141	-43.2308922	40	2016	2018-03-15T10:27:13.832Z
1.0462981	7.1109962	6.6201696	-22.8589141	-43.2308922	47	2195	2018-03-15T10:27:14.012Z
0.019154198	6.7997403	6.715941	-22.8589141	-43.2308922	45	2380	2018-03-15T10:27:14.196Z
0.7637737	7.1133904	7.1947956	-22.8589141	-43.2308922	45	2554	2018-03-15T10:27:14.370Z
0.047885496	7.692805	7.3815494	-22.8589141	-43.2308922	46	2735	2018-03-15T10:27:14.551Z
0.27055305	6.811712	6.6153812	-22.8589141	-43.2308922	46	2911	2018-03-15T10:27:14.728Z
-0.6847626	5.3919067	8.571504	-22.8589141	-43.2308922	32	3091	2018-03-15T10:27:14.907Z
0.92658436	6.708758	6.3137026	-22.8589141	-43.2308922	47	3274	2018-03-15T10:27:15.091Z
-0.035914123	6.8452315	6.55313	-22.8589141	-43.2308922	46	3454	2018-03-15T10:27:15.271Z
1.297697	7.285778	6.3759537	-22.8589141	-43.2308922	49	3634	2018-03-15T10:27:15.451Z
0.41420954	7.0966306	8.605023	-22.8589141	-43.2308922	40	3810	2018-03-15T10:27:15.626Z
1.9752767	7.4964743	6.818895	-22.8589141	-43.2308922	49	3994	2018-03-15T10:27:15.811Z
-0.12450229	5.1859994	5.152479	-22.8589141	-43.2308922	45	4173	2018-03-15T10:27:15.990Z
-2.5810282	8.006455	4.9609375	-22.8589141	-43.2308922	59	4355	2018-03-15T10:27:16.171Z
-0.76137936	7.7526617	7.091842	-22.8589141	-43.2308922	48	4534	2018-03-15T10:27:16.350Z
0.18435916	5.355993	6.299337	-22.8589141	-43.2308922	40	4713	2018-03-15T10:27:16.529Z
-0.29689008	6.0527267	6.0886407	-22.8589141	-43.2308922	45	4893	2018-03-15T10:27:16.710Z

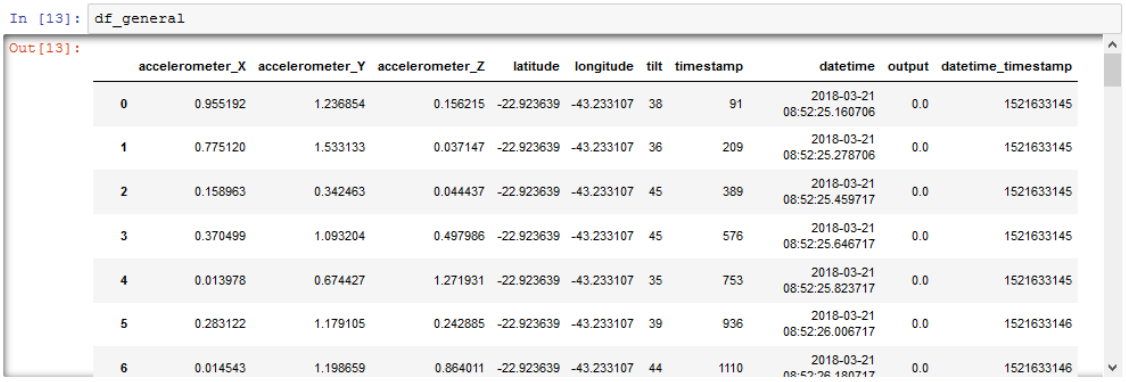
Figura 37 Amostras do arquivo CSV genérico

As colunas *timestamp* e *realtime* servem exclusivamente para auxiliar na identificação dos eventos registrados no arquivo genérico. Em virtude disso, a primeira lógica aplicada é de comparar os valores do atributo *timestamp* do arquivo de eventos com o arquivo geral. Cria-se, portanto, uma nova coluna no maior arquivo chamada de *output* informando o valor de 0, 1 (Buraco) ou 2 (Quebra-Mola).

É criado uma coluna com valores do tipo *datetime* utilizando as informações referentes ao tempo para ser utilizada como índice e termos um *dataframe* com dados temporais. A fim de possuir uma maior representatividade da quantidade de eventos em

relação a todos os dados capturados, foi desenvolvido um método que possibilita replicar os eventos com base em um intervalo de tempo. Como existe um possível *delay* do usuário informar que um determinado evento ocorreu, o seguinte modelo de negócio foi estabelecido: se há um evento em um instante "t", todos os registros 1 segundo antes e depois também conterão o mesmo valor de *output*.

Além dessas correções mencionadas, também foi realizada uma normalização em janela para manter os dados com a mesma dimensionalidade. Um outro algoritmo importante que foi desenvolvido se trata de um método que transforma os registros temporais em problema de aprendizagem supervisionada.

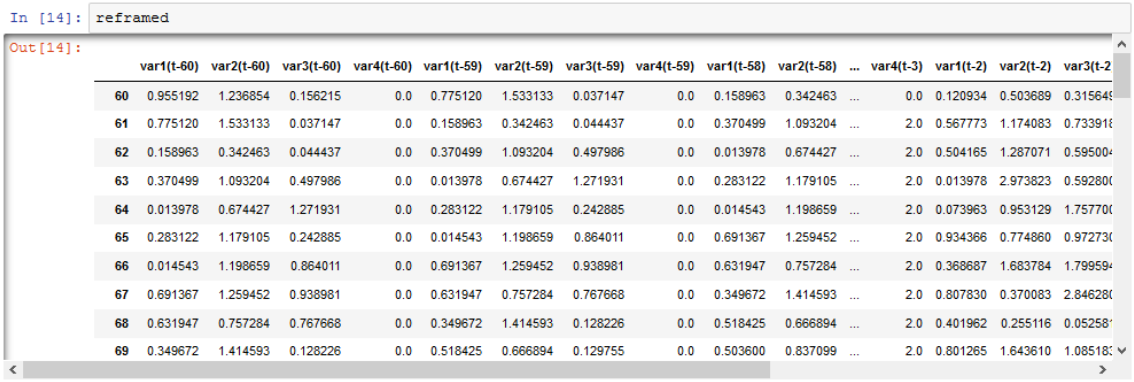


```
In [13]: df_general
```

Out[13]:

	accelerometer_X	accelerometer_Y	accelerometer_Z	latitude	longitude	tilt	timestamp	datetime	output	datetime_timestamp
0	0.955192	1.236854	0.156215	-22.923639	-43.233107	38	91	2018-03-21 08:52:25.160706	0.0	1521633145
1	0.775120	1.533133	0.037147	-22.923639	-43.233107	36	209	2018-03-21 08:52:25.278706	0.0	1521633145
2	0.158963	0.342463	0.044437	-22.923639	-43.233107	45	389	2018-03-21 08:52:25.459717	0.0	1521633145
3	0.370499	1.093204	0.497986	-22.923639	-43.233107	45	576	2018-03-21 08:52:25.646717	0.0	1521633145
4	0.013978	0.674427	1.271931	-22.923639	-43.233107	35	753	2018-03-21 08:52:25.823717	0.0	1521633145
5	0.283122	1.179105	0.242885	-22.923639	-43.233107	39	936	2018-03-21 08:52:26.006717	0.0	1521633146
6	0.014543	1.198659	0.864011	-22.923639	-43.233107	44	1110	2018-03-21 08:52:26.180717	0.0	1521633146

Figura 38 Dataframe genérico com registros temporais



```
In [14]: reframed
```

Out[14]:

	var1(t-60)	var2(t-60)	var3(t-60)	var4(t-60)	var1(t-59)	var2(t-59)	var3(t-59)	var4(t-59)	var1(t-58)	var2(t-58)	...	var4(t-3)	var1(t-2)	var2(t-2)	var3(t-2
60	0.955192	1.236854	0.156215	0.0	0.775120	1.533133	0.037147	0.0	0.158963	0.342463	...	0.0	0.120934	0.503689	0.31564
61	0.775120	1.533133	0.037147	0.0	0.158963	0.342463	0.044437	0.0	0.370499	1.093204	...	2.0	0.567773	1.174083	0.73391
62	0.158963	0.342463	0.044437	0.0	0.370499	1.093204	0.497986	0.0	0.013978	0.674427	...	2.0	0.504165	1.287071	0.59500
63	0.370499	1.093204	0.497986	0.0	0.013978	0.674427	1.271931	0.0	0.283122	1.179105	...	2.0	0.013978	2.973823	0.59280
64	0.013978	0.674427	1.271931	0.0	0.283122	1.179105	0.242885	0.0	0.014543	1.198659	...	2.0	0.073963	0.953129	1.75770
65	0.283122	1.179105	0.242885	0.0	0.014543	1.198659	0.864011	0.0	0.691367	1.259452	...	2.0	0.934366	0.774860	0.97273
66	0.014543	1.198659	0.864011	0.0	0.691367	1.259452	0.938981	0.0	0.631947	0.757284	...	2.0	0.368687	1.683784	1.79959
67	0.691367	1.259452	0.938981	0.0	0.631947	0.757284	0.767668	0.0	0.349672	1.414593	...	2.0	0.807830	0.370083	2.84628
68	0.631947	0.757284	0.767668	0.0	0.349672	1.414593	0.128226	0.0	0.518425	0.666894	...	2.0	0.401962	0.255116	0.05258
69	0.349672	1.414593	0.128226	0.0	0.518425	0.666894	0.129755	0.0	0.503600	0.837099	...	2.0	0.801265	1.643610	1.08518

Figura 39 Dataframe em formato para realizar aprendizagem supervisionada

Após ter realizado essas mudanças, os dados são separados em 80% para treino e 20% para teste. Antes de desenvolver o modelo da rede neural, as saídas dos registros que variavam entre 0, 1 e 2 foram convertidas para dados categóricos. Isto é, foram transformadas em um vetor 1x3, sendo [0 0 0] nenhum evento detectado (default), [0 1 0] buraco e [0 0 1] quebra-mola.

5.4.2.2 – Fase de Produção

Os tratamentos dos dados, os métodos de processamento aplicados não são diferentes daqueles mencionados na seção 5.4.2.1.

O contraste está presente na entrada e saída dos dados. Enquanto o *input* dos dados na fase de teste é através de arquivos em formato csv, na fase de produção os dados são lidos por meio da fila chamada “LUNAR_ToBeTested”. Ao passo que os dados, durante os testes, não são salvos em nenhum local, em produção os registros são enviados para a fila “LUNAR_Processed” com o objetivo de serem lidos pelo processo Recorder e salvos na base de dados.

5.4.3 – Processo Recorder

O processo, escalável, denominado “Recorder”, possui duas tarefas bem definidas. A primeira trata-se de realizar a leitura da mensagem contida na fila “LUNAR_Processed”. A segunda e última função é relacionada a inserção do conteúdo no banco de dados não relacional MongoDb.

Nesse ponto do projeto, não são executados validações e tratamentos nos dados de entrada. O Recorder, assim como o processo Worker, estão hospedados em máquinas separadas de família T2 e tamanho micro.

5.5 – Aplicação de Machine Learning

O algoritmo de aprendizado de máquina da aplicação foi estruturado através do modelo de rede neural LSTM. Existem bibliotecas, como Keras, para a linguagem de programação Python que já possui toda a construção do LSTM bem definida. O Keras é uma API de redes neurais de alto nível, possuindo diversas funções para desafios que exigem *Deep Learning*. É através dessa API que parte do processamento dos dados e de toda a fundamentação da rede neural foi baseada.

Design Network

```
In [5]: # design network
        model = Sequential()
        model.add(LSTM(120,input_shape=(x_train.shape[1],x_train.shape[2])))
        model.add(Dropout(0.2))
        model.add(Dense(3, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figura 40 Estrutura da Rede Neural aplicada

O projeto Lunar utiliza o *Sequential Model* do Keras adicionando camadas como a própria *LSTM*, *Dropout* e *Dense*. A design da rede está com 120 neurônios na camada *LSTM*, *Dropout* de 20% para reduzir a probabilidade de ocorrer *overfitting* e por fim, camada *Dense* com 3 neurônios utilizando método de ativação "*softmax*" que informa a probabilidade de os dados serem de uma determinada classe. Para realizar a compilação, foi utilizado o método de perda "*categorical_crossentropy*" com otimizador "*adam*".

Os dados de entrada para a rede LSTM foram estruturados no seguinte formato:

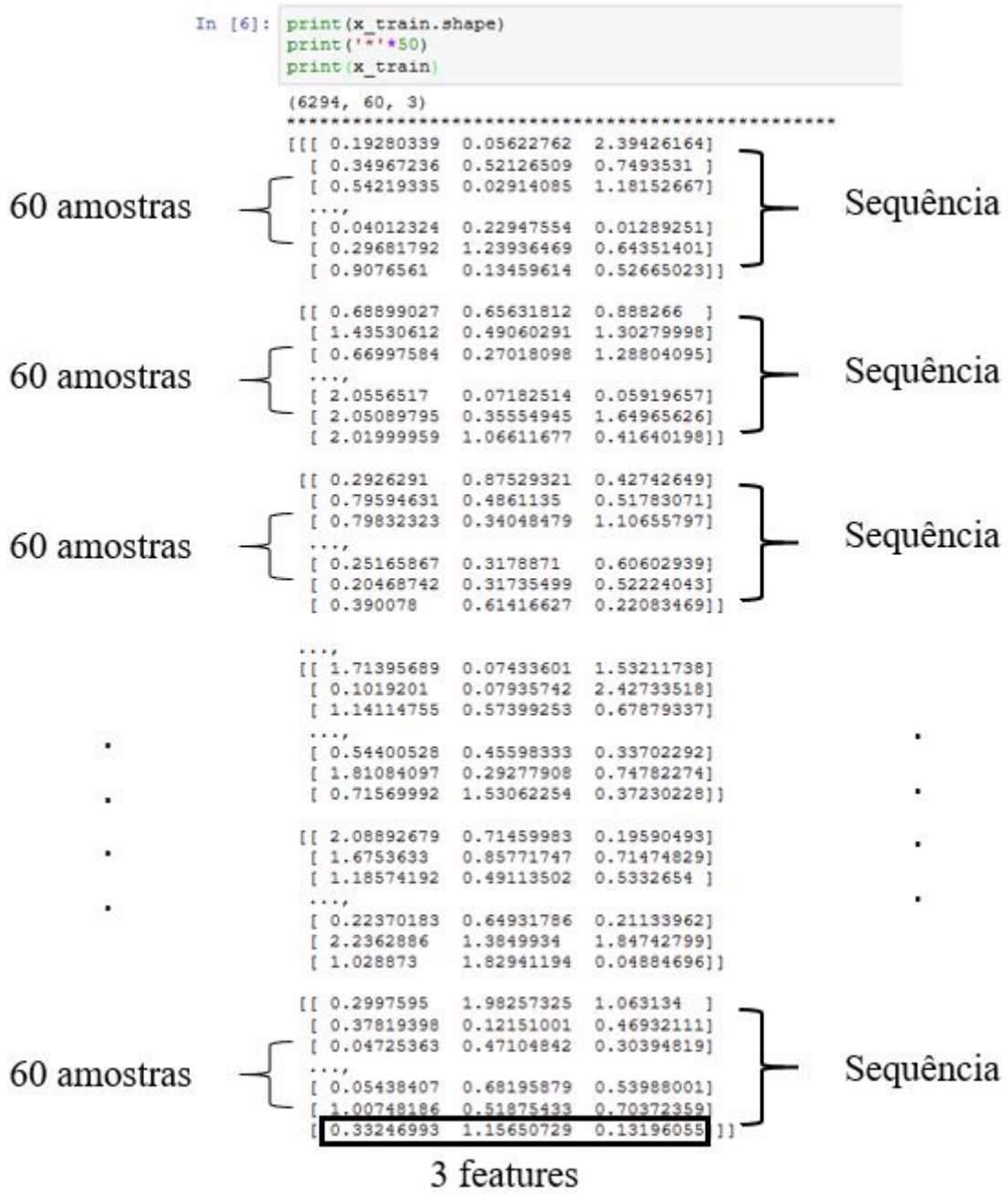


Figura 41 Estrutura dos dados de entrada já normalizados

O vetor da entrada é composto por N sequências de M amostras contendo 3 (três) *features* que são informações relacionadas ao eixo x, y e z do sensor acelerômetro. A quantidade de sequências (N) está diretamente relacionada com a quantidade de registros armazenadas no *dataset* que está sendo processado. A variável M representa a quantidade de amostras que cada sequência possui, ou seja, o quanto de informação temporal está presente em uma determinada amostra. O valor estabelecido foi de 60.

As amostras foram capturadas em um intervalo de 30 milissegundos pelo aplicativo Android, isto significa que, M igual a 60 contém amostras equivalentes a 1.8 segundos. Utilizando como base um carro percorrendo um trajeto a 60 km/h (16,7 m/s), a cada 50 centímetros de distância são salvos os dados referentes ao acelerômetro e as coordenadas de latitude e longitude.

Cada sequência dentro do vetor de entrada conterá, no total, informações de aproximadamente dois segundos anteriores aos eventos detectados.

Por fim, a respeito da estrutura dos dados de saída, a Figura 42 apresenta como é a sua representação, cuja explicação encontra-se em seguida.

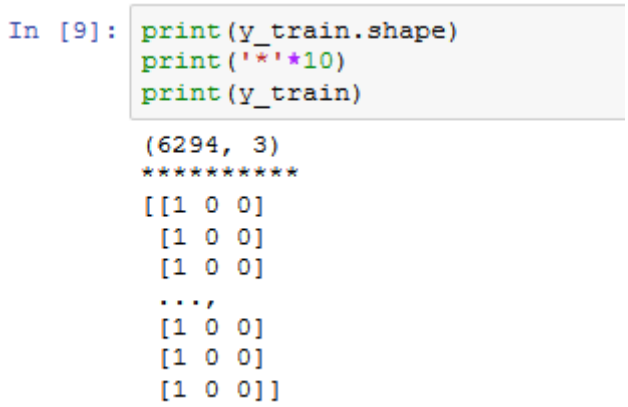


Figura 42 Estrutura dos dados de saída

A variável de saída é um vetor aonde cada item corresponde a saída esperada pela sequência correspondente. Cada item trata-se de um vetor 1x3 aonde [0 0 0] corresponde a nenhum evento, [0 1 0] evento de buraco e [0 0 1] evento de quebra-mola. A matriz inteira possui dimensões Nx3, onde N representa o número de sequências existentes dentro do conjunto de dados que está sendo tratado naquele momento.

5.6 – Implementação

Nesta seção será apresentado o diagrama de blocos que apresenta todos os processamentos executados e a explicação detalhada de cada um dos blocos. Todos os processos desse fluxograma foram elaborados através da linguagem de programação Python com o uso da aplicação web Jupyter Notebook. Os procedimentos desenvolvidos

durante a fase de teste do modelo de *machine learning* treinado são apresentados na figura a seguir.

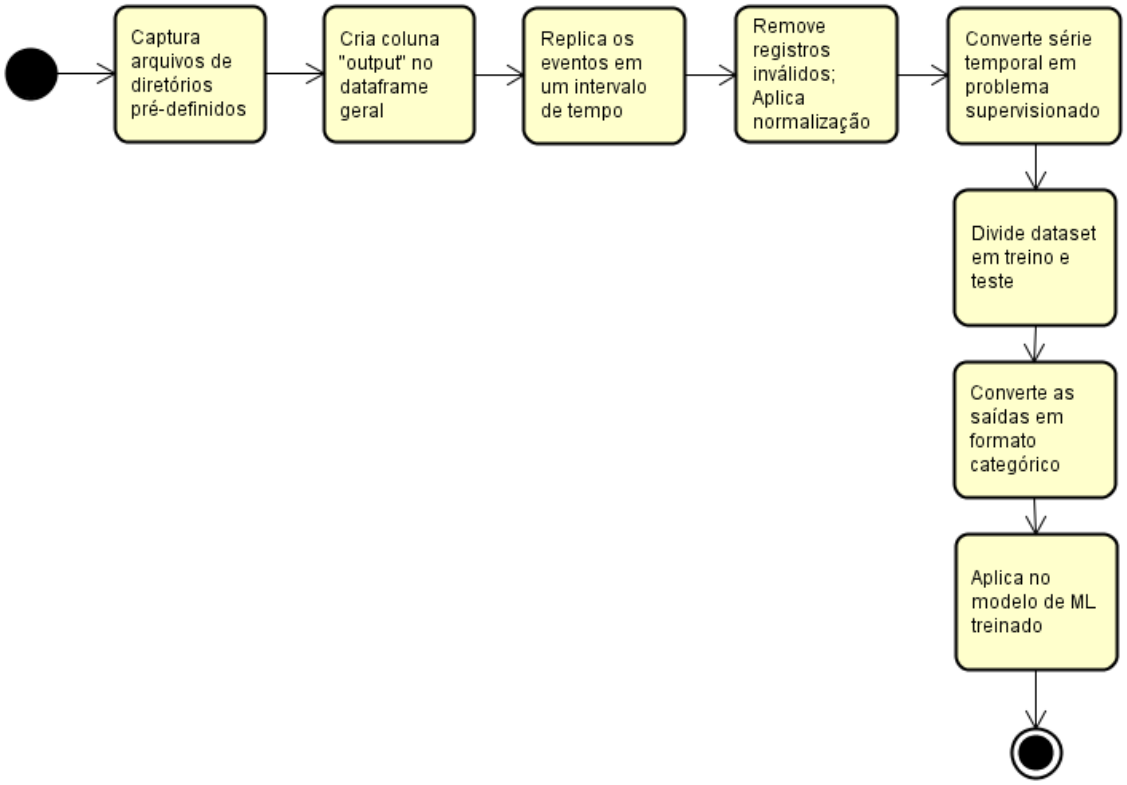


Figura 43 Procedimentos realizados durante a fase de testes

Durante a fase de teste do modelo de aprendizagem de máquina, os arquivos de *input* são aqueles gerados pelo aplicativo teste para dispositivos Android. Esses arquivos são inseridos em um diretório pré-definido para serem processados. A busca por esses documentos é feita através de uma padronização inserida na nomenclatura. Arquivos contendo o prefixo “SpeedBump_” são referentes aos eventos de quebra-molas. “Pothole_”, obstáculos de buraco. Caso o arquivo não possua nenhum desses dois identificadores, é considerado como arquivo geral. Isto é, um documento que não possui evento detectado, porém contém as informações das coordenadas do GPS, inclinação do dispositivo e outros recursos.

A leitura dos arquivos é feita através do método *read*_csv da biblioteca Pandas que retorna um objeto do tipo *dataframe*. Uma das principais diferenças entre o *dataframe* do arquivo genérico e dos eventos é que o primeiro não possui uma coluna *output*. Para

essa coluna ser inserida neste arquivo, é realizada uma comparação dos valores da coluna auxiliar chamada *timestamp*.

Um importante empecilho a ser solucionado é a disparidade entre a quantidade de registros existentes no *dataframe* genérico e dos eventos. Como as informações dos sensores são capturadas aproximadamente a cada 30ms, é intuitivo concluir que a quantidade de registros sem eventos será superior à do quebra-molas e do buraco. A solução encontrada para resolver este impasse foi a elaboração de um método que replica a quantidade de eventos com base no *timestamp* daquele instante e do tempo em segundos informado na chamada da função a seguir. O método apelidado de *insert_repeated_values_on_dataframe* possui basicamente os seguintes parâmetros: *dataframe*; nome da coluna que sofrerá a ação; os índices do *dataframe* a serem replicados; o tempo em segundos; *flag* booleana para informar se deseja replicar tanto nos registros do passado como do futuro. Enviando como *true* o valor para variável booleana, tempo de 2 segundos, um índice do *dataframe* que possua o instante 2018-04-04T12:06:10.000Z e saída X de um determinado evento, o resultado dessa chamada irá modificar todas as saídas dos registros entre 2018-04-04T12:06:08.000Z e 2018-04-04T12:06:12.000Z pelo valor X.

Após ter sido criada a lógica para diminuir a disparidade dos registros, é executado um filtro com base no *tilt* do dispositivo. Como, em todas as fases de teste, o celular manteve-se mais próximo da vertical (90 graus) do que na horizontal para realizar a captura dos dados, os registros com uma inclinação inferior a 30 graus são descartados. Posteriormente, é aplicado uma normalização completa com o objetivo de manter todos os dados na mesma ordem de grandeza.

Aplicar as técnicas de processamento em uma amostra de um determinado instante de tempo não produzirá nenhum resultado coerente. A percepção de como o veículo se comporta ao passar por um determinado obstáculo na via pavimentada é dada pela análise de dados de um conjunto significativo de amostras. O efeito provocado pelo evento no veículo sucede instantes de tempo antes e depois do ocorrido. Com base nessa problemática, é necessária uma lógica que trabalhe com séries temporais. Aprendizado de máquina supervisionado e não supervisionado trabalham com um valor único de entrada, isto é, para um determinado X a saída é Y. No caso das séries temporais, a entrada não é um único valor, mas sim uma lista de números que possuem relações entre si. Já a saída, continua não sendo representada por um vetor. Porém, é necessário um algoritmo que decida qual é o valor de saída que representará o *input* passado. Para isso, verifica

nesse espaço amostral qual o valor mais frequente. O valor que possui a maior frequência representará a saída daquele grupamento de informação. O algoritmo utilizado para converter séries temporais em problema de aprendizagem supervisionada foi criado pelo Ph.D. Jason Brownlee [25].

Feita a conversão que permite a viabilidade do uso de algoritmos de aprendizagem supervisionada, os dados foram separados, através do método *train_test_split* do pacote *sklearn.model_selection*, em 80% treino e 20% teste com o objetivo de diminuir a chance do modelo possuir *overfitting*.

Tanto o vetor de saída de treino como o de teste possuem uma única dimensão que se refere a saída de um conjunto de amostras. Para manter no formato esperado pelo modelo *Sequential* do Keras, foi executado um algoritmo que realiza a categorização. O evento de quebra-mola, antes identificado pelo número 2, contém o formato [0 0 1]. Buraco, [0 1 0]. Registros que não foram detectados eventos, [1 0 0]. Após realizado essa alteração nos vetores de saída, os dados são passados para o modelo de aprendizado de máquina já treinado.

5.7 – Análise dos Dados

Uma das etapas principais e que requer uma quantidade expressiva de validações é a etapa da análise dos dados após a aplicação dos filtros e dos ajustes. É nesse estágio que define se, todo o algoritmo desenvolvido para efetuar as alterações dos dados, estão correspondendo de modo satisfatório.

No projeto Lunar são utilizados dois recursos distintos para elaborar um diagnóstico que contenha a real concepção do quanto o algoritmo de aprendizagem de máquina está classificando com êxito.

Os projetos que envolvem algoritmos de *machine learning* costumam utilizar uma quantidade significativa de parâmetros para realizar o treinamento e o teste dos dados. No projeto Lunar, alguns destes parâmetros são: número de neurônios na camada LSTM, número de *epochs*, *batchsize* e porcentagem da camada d*ropout*. Cada tipo de projeto possui uma configuração mais adequada dos parâmetros para obter o resultado mais satisfatório. Existem bibliotecas, para a linguagem Python, desenvolvidas especialmente para reduzir o trabalho manual do programador de realizar as alterações nos parâmetros esperando uma possível resposta melhor do seu modelo.

A biblioteca *Pyswarm* [26] permite que seja passado o valor mínimo e máximo a ser testado para cada parâmetro de interesse e também uma função a ser minimizada para que seja encontrado o melhor conjunto das variáveis fornecidas.

Em vista disso, a primeira etapa da análise de dados foi a utilização do PSO (*Particle Swarm Optimization*) através da *Pyswarm library* para identificar os melhores valores dos parâmetros do modelo de aprendizado de máquina desenvolvido no projeto Lunar. Identificado esses números, é desenvolvida uma Matriz de Confusão [27] com novos registros, isto é, que não foram utilizados em treino e em teste. Por fim, um outro método aplicado foi de realizar novamente alguns trajetos já percorridos, porém sem efetuar cliques manuais nos botões do aplicativo mobile.

5.7.1 – PSO (Particle Swarm Optimization)

O algoritmo de otimização PSO é utilizado como a principal ferramenta para detectar as melhores combinações dos parâmetros informados no intuito de obter o menor erro possível dado a função a ser minimizada.

As variáveis testadas pelo PSO e seus respectivos valores mínimos e máximos são:

- Número de neurônios na camada LSTM: 100 a 160
- Porcentagem da camada *Dropout*: 10% a 40%
- Número de épocas do método *fit*: 50 a 200
- Tamanho do batch do método *fit*: 100 a 400

As 20 melhores precisões dos obstáculos estão apresentadas na Tabela 1.

Tabela 1 Vinte melhores resultados dos testes com obstáculos diferenciados

Épocas	Tamanho do Batch	LSTM (neurônios)	Dropout	Precisão Geral	Precisão dos Eventos
70	250	120	20%	88,1%	56,30%
140	300	120	10%	91,36%	49,63%
150	350	100	10%	89,67%	47,92%
100	300	120	10%	91,49%	46,77%
170	400	100	10%	89,19%	44,66%

150	300	120	10%	91,55%	44,54%
130	150	120	10%	91,24%	43,95%
80	300	100	10%	89,79%	43,95%
80	300	120	20%	90,46%	43,82%
160	400	120	10%	91,24%	43,14%
80	350	120	20%	91,49%	42,88%
80	100	120	20%	89,98%	42,67%
100	200	160	30%	90,52%	42,65%
100	150	100	10%	90,1%	42,65%
200	400	140	40%	90,04%	42,41%
90	250	120	20%	90,1%	42,37%
100	200	120	20%	91,06%	42,28%
60	300	100	20%	91,3%	41,84%
150	350	120	10%	91,36%	41,76%
60	300	120	20%	90,76%	41,62%

O conjunto de parâmetros que gerou a melhor precisão geral, isto é, considerando os eventos *default*, *speedbump* e *pothole* foi:

- Número de neurônios na camada LSTM: 120
- Porcentagem de Dropout: 20%
- Número de Épocas: 70
- Batch_size: 250

Uma outra avaliação realizada foi diminuindo a complexidade do modelo de rede neural. Em outras palavras, a saída da rede ter somente dois valores possíveis. São eles: evento sem obstáculo e com obstáculo.

Neste caso, as variáveis testadas pelo PSO e seus respectivos valores mínimos e máximos são:

- Número de neurônios na camada LSTM: 100 a 120
- Porcentagem da camada *Dropout*: 10% a 20%
- Número de épocas do método *fit*: 50 a 200
- Tamanho do batch do método *fit*: 100 a 400

Os testes, com essa composição, estão fornecidos na Tabela 2.

Tabela 2 Resultados dos testes do PSO com a saída da rede limitada

Épocas	Tamanho do Batch	LSTM (neurônios)	Dropout	Precisão Geral	Precisão dos Eventos
50	100	100	10%	82,37%	41%
70	350	100	10%	82,61%	40%
50	150	100	10%	82,67%	40%
70	200	100	10%	84,24%	38%
200	200	120	10%	84,9%	38%
90	350	100	10%	81,58%	38%
60	200	100	10%	84,0%	38%
70	150	100	10%	82,73%	38%
60	250	100	10%	83,82%	37%
160	200	100	10%	83,39%	37%
60	400	100	10%	83,76%	37%
80	350	100	10%	83,51%	37%
130	300	100	10%	83,15%	36%
60	350	100	10%	82,73%	36%
60	300	100	10%	83,45%	36%
170	400	100	10%	83,15%	35%
160	250	100	10%	82,13%	35%
80	400	100	10%	83,45%	35%
130	250	100	10%	84,18%	35%
170	100	100	10%	83,33%	34%

Para essa nova validação, o conjunto de parâmetros que gerou a melhor precisão geral foi:

- Número de neurônios na camada LSTM: 100
- Porcentagem de Dropout: 10%
- Número de Épocas: 50
- Batch_size: 100

5.7.2 – Matriz de Confusão

A Matriz de Confusão é um tipo de tabela que permite a visualização direta do desempenho de um algoritmo de aprendizado supervisionado. Cada coluna da matriz representa as classes previstas pelo algoritmo, enquanto as linhas representam as classes reais. Os números presentes na diagonal principal nos dizem a quantidade de dados que

foram classificados como o previsto. Em relação aos demais valores, significam o quanto de falha ocorreu no processo de catalogação. A Figura 44 e 45 apresentam a matriz de confusão a partir da melhor combinação de parâmetros de acordo com as Tabelas 1 e 2, respectivamente.

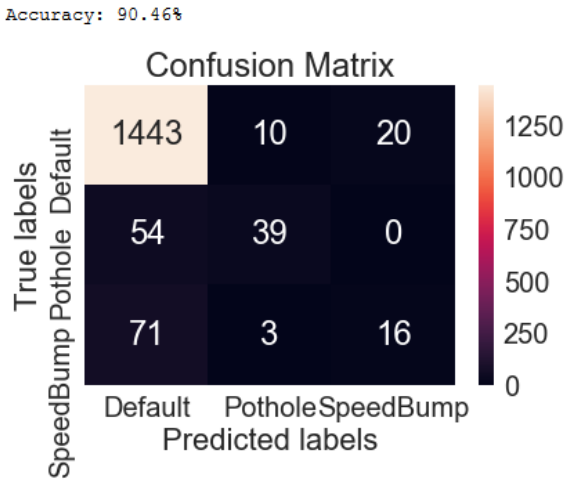


Figura 44 Matriz de Confusão para obstáculos diferenciados

A Figura 42 diz que 1443 eventos *Default*, não caracterizados como nenhum tipo de obstáculo, foram classificados corretamente pelo algoritmo desenvolvido. Já em relação ao evento de *Pothole* (buraco), o número foi de 39. Para o *SpeedBump* (quebra-mola), 16. De acordo com os dados testados para gerar essa Matriz de Confusão, 3 foram classificados como buracos, mas na verdade eram quebra-molas. O caso inverso, não houve um evento de buraco sendo catalogado como quebra-mola.

Existem fatores importantes que estão diretamente relacionados à quantidade de falsos positivos e falsos negativos presentes na matriz de confusão. As assinaturas, tanto com e sem obstáculos, são sensíveis ao modo em que o veículo passa, à velocidade instantânea e à qualidade da via pavimentada. O automóvel pode atravessar no centro, ou na borda do obstáculo, e como consequência, registrará um comportamento com intensidade e formato diferente.

Uma outra condição que impacta diretamente é o ambiente utilizado para a construção do modelo. A via pavimentada operada apresenta imperfeições como ondulações e outros tipos de imperfeições ao longo de todo o trajeto.

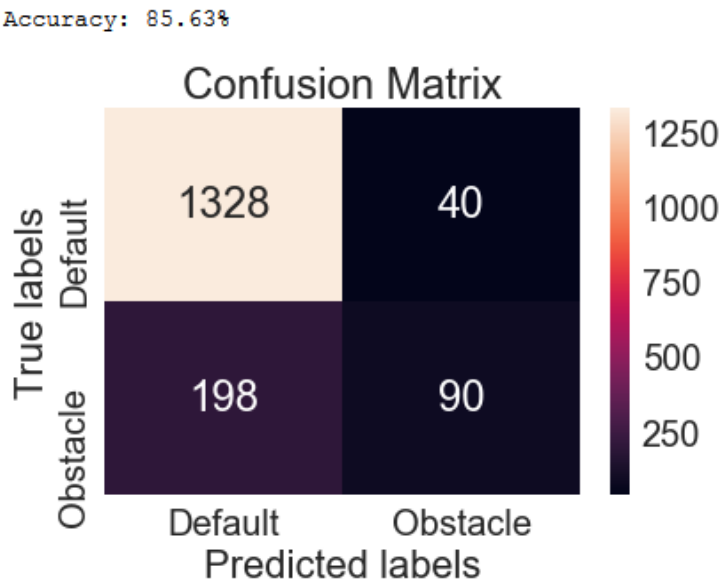


Figura 45 Matriz de Confusão para obstáculos com o mesmo valor de saída

Já a Figura 45 está relacionada ao experimento com a saída da rede limitada em sem e com obstáculo. Informa que 1328 eventos *Default*, não caracterizados como nenhum tipo de obstáculo, foram classificados corretamente pelo algoritmo desenvolvido. Já em relação a algum obstáculo, o número foi 90.

O total de obstáculos categorizados incorretamente foi 238, sendo 198 de eventos confundidos com assinaturas comuns do trajeto percorrido. A presença de imperfeições na pista, como ondulações e desnivelamento, pode ter contribuído na geração de um valor considerável de erros na detecção dos obstáculos.

5.8 – Disponibilização dos Dados

Dados capturados, analisados e processados. Todas as etapas necessárias para poder enfim, compartilhar com os usuários comuns e agentes de instituições governamentais informações relacionadas à qualidade das vias pavimentadas de uma determinada região.

O intermédio entre o usuário e os registros presentes no banco de dados se dá através uma Web API. O serviço web está disponibilizado em uma instância da AWS, podendo ser usufruída através do emprego de parâmetros na própria URL de consulta. O usuário poderá acessar os dados, em formato JSON, de acordo com o interesse. Os

parâmetros disponíveis e que podem ser combinados são: *output, latitude, longitude, city, state, numberofdays* e *limit.*

O parâmetro *output* é do tipo inteiro e aceita 0, 1 ou 2 como valor. É o atributo responsável por apresentar informações referentes a nenhum evento detectado (0), eventos de buraco (1) e eventos de quebra-molas (2).

A *latitude* e *longitude*, que são do tipo double, permitem aos usuários a realizarem filtros em busca de registros de um local específico. Aliada as coordenadas do GPS, o parâmetro *city* e *state* permitem a possibilidade de aplicar filtros mais genéricos como cidade ou estado de modo geral.

Os campos *numberofdays* e *limit* são designados a fornecerem um limite quantitativo e de informações recentes aos usuários, permitindo assim uma maior customização e leque de possibilidades de obter os dados.

A Figura 46 apresenta o objeto utilizado no mapeamento dos parâmetros informados pela requisição à Web API.

```
[BsonIgnoreExtraElements]
public class ApiQueryObject
{
    private int _NumberOfDays;
    private int _Limit;

    [JsonProperty(PropertyName = "output")]
    public int      Output          { get; set; }

    [JsonProperty(PropertyName = "latitude")]
    public double   Latitude        { get; set; }

    [JsonProperty(PropertyName = "longitude")]
    public double   Longitude       { get; set; }

    [JsonProperty(PropertyName = "city")]
    public string   City            { get; set; }

    [JsonProperty(PropertyName = "state")]
    public string   State           { get; set; }

    [JsonProperty(PropertyName = "numberofdays")]
    public int      NumberOfDays    { get { return _NumberOfDays; } set {
                                                                        if (value > 30)
                                                                        {
                                                                            _NumberOfDays = 30;
                                                                        }
                                                                        else if (value > 0 && value <= 30)
                                                                            _NumberOfDays = value;
                                                                    }}

    [JsonProperty(PropertyName = "limit")]
    public int      Limit           { get { return _Limit; }        set {
                                                                        if (value > 1000)
                                                                        {
                                                                            _Limit = 100;
                                                                        }
                                                                        else if (value > 0 && value <= 100)
                                                                            _Limit = value;
                                                                    }}
```

Figura 46 Objeto a ser mapeado em cada requisição na Web API

Toda requisição mobile à Web API é processada pelo controlador MobileController. A estrutura parcial deste *controller* é apresentada na Figura 47.

```
[RoutePrefix("api/v1/public/lunar")]
public class MobileController : ApiController
{
    Private Attributes

    [HttpGet]
    [Route("")]
    public HttpResponseMessage GetMobileRecords([FromUri] ApiQueryObject queryObject)
    {
        Validations

        List<string> results = new List<string>();
        try
        {
            bool success;

            // Build query from GET parameters
            IMongoQuery query = BuildQuery(queryObject, out success);
```

Figura 47 Estrutura parcial do MobileController – *Parte 1*

As requisições do tipo GET e, que não possuem uma rota definida exceto àquela expressa pelo prefixo do controlador, vão ser consumidas pelo método *GetMobileRecords*. Este método recebe como parâmetro um objeto do tipo *ApiQueryObject* e tem como função de resgatar os registros no *database* dado alguns filtros na requisição.

A primeira função chamada dentro deste método é a *BuildQuery*. Ela é composta por dezenas de estruturas condicionais a fim de construir exatamente a query necessária para retornar as informações desejadas pelo usuário.

A etapa seguinte à construção da query é de realizar uma busca na base de dados para extrair as informações de interesse. Com os dados em mãos, eles são convertidos em formato JSON e inserido em uma coleção a ser retornada pelo método. A Figura 48 apresenta como é feita a busca, conversão e a inserção na lista de retorno.

```
MongoCursor<MobileRecordObject> cursor = Collection.FindAs<MobileRecordObject>(query).SetLimit(queryObject.Limit);

// Iterate over all records on collection
foreach (MobileRecordObject rec in cursor)
{
    results.Add(JsonConvert.SerializeObject(rec));
}
```

Figura 48 Estrutura parcial do MobileController - Parte 2

Capítulo 6

Conclusão

6.1 – Conclusões

Como palavras finais de conclusão deste trabalho, ainda que o produto desenvolvido não tenha sido, de fato, preparado para ser usufruído pelo mercado, procurou-se estruturar o projeto de forma que atenda quaisquer quantidades de demandas e que fosse de manutenível.

Neste projeto, aplicou-se estudos sobre algoritmos de aprendizado de máquina, testes com o sensor acelerômetro, técnicas de otimização de hiperparâmetros, construção de um serviço web, serviços da Cloud e pesquisas sobre métodos utilizados por instituições governamentais. Como resultado desse esforço, antes mesmo de se investir tempo para realizar a codificação necessária para a implementação das funcionalidades, tinha-se claro em mente quais atributos deveriam ser capturados, quais tecnologias a serem utilizadas e como seriam feitas as validações iniciais para analisar a viabilidade do projeto.

Além dessa preparação para dar início de fato ao projeto, uma preocupação que permeou em todo o processo de construção do trabalho foi em organizar, de maneira escalável, cada etapa desenvolvida. Dividir as funcionalidades em pequenos processos bem definidos foi o caminho trilhado para tornar, o aplicativo como um todo, escalável, fácil manutenção e flexível.

Conforme se revelou no capítulo de Análise dos Dados, a aplicação Lunar possui limitações claras no que tange à classificação automática, seja pela existência de assinaturas diferentes para cada obstáculo, como também da falta de aplicação de outros filtros de pré-processamento. Um outro fator impactante foi o ambiente de estudo inadequado. As imperfeições na via pavimentada foram frequentes ao longo do todo trajeto percorrido para realizar o treinamento e teste do modelo. Apesar disso, o resultado que se tem ao fim deste trabalho corresponde às expectativas postas, de modo que é possível concluir que a sua realização foi satisfatória.

6.2 – Trabalhos Futuros

Como trabalho futuro, é possível conceber a elaboração de um modelo genérico que seja treinado pelo algoritmo de aprendizado de máquina deixando de visar um determinado percurso. Há grande espaço para realizar uma quantidade maior de captura das informações com o objetivo de realizar mais testes e treinos a fim de construir um modelo global ou até mesmo diversos modelos menores que sejam preparados para certos tipos de condições das vias pavimentadas. Também é possível conceber um ambiente virtual que incentive uma maior participação dos usuários com o projeto, tornando-o mais robusto e confiável. Assim como a execução de uma quantidade maior de filtros de pré-processamento e o uso de outros modelos de aprendizado de máquina. Ainda, será útil a construção de um sistema de segurança para evitar possíveis envios má intencionados à base de dados e também o estudo e a implementação da captura de diferentes obstáculos presentes nas vias pavimentadas.

Bibliografia

[0] SUÍÇA. WORLD HEALTH ORGANIZATION. (Org.). Global status report on road safety 2015. Geneva: World Health Organization, 2015. 340 p. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/>. Acesso em: 16 jan. 2018.

[1] LIMA, L.A.O. "Sistema de detecção das condições asfálticas das ruas através do uso passivo de smartphones": Monografia do MBA em Tecnologia da Informação - Executivo, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2016.

[2] _____, Wikipedia, a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Acelerômetro>. Acesso em: 24 de Agosto de 2017.

[3] KOCH, Christian; BRILAKIS, Ioanis. "Pothole detection in asphalt pavement images*"*. Advanced Engineering Informatics, v.25, n.3, pp.507-515, 2011. doi: 10.1016/j.aei.2011.01.002

[4] ADARKWA, Amanor; Attoh-Okine, Nii. "Pavement crack classification based on tensor factorization". Construction and Bulding Materials, v.48, pp.853-857, 2013. doi: 10.1016/j.conbuildmat.2013.07.091

[5] NITSCHE, Philippe; STUTZ, Rainer; KAMMER, Michael; MAURER, Peter. "Comparison of Machine Learning Methods for Evaluating Pavement Roughness Based on Vehicle Response", Journal of Computing in Civil Engineering, v.28, n.4, 2014. doi: 10.10161/(ASCE)CP.1943-5487.0000285

[6] FEDELI, Peres. "Introdução à ciência da computação", 2ª Edição. São Paulo: Cengage Learning, 2010.

[7] CASS, Stephen. "The 2017 Top Programming Languages". Disponível em: <http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>. Acesso em: 29 de Agosto de 2017

[8] _____, "Smartphone OS Market Share, 2017 Q1", http://www.idc.com/promo/smartphone-market-share/os, Acesso em: 28 de Agosto de 2017)

[9] _____, "Zenith forecasts 75% of internet use will be mobile in 2017". Disponível em: < https://www.zenithmedia.com/mobile-forecasts-75-internet-use-will-mobile-2017/ >. Acesso em: 28 de Agosto de 2017

[10] SAVERS, Michael W.; KARAMIHAS, Steven M. "The Little Book of Profiling. Basic Information about Measuring and Interpreting Road Profilers", The Regent of the University of Michigan, pp.45-52. Setembro de 1998.

[11] _____, Wikipedia, a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Hype_cycle>. Acesso em: 15 de Março de 2018.

[12] HOCHREITER, Josef. "Untersuchungen zu dynamischen neuronalen Netzen". 1991. 74 pág, Instituto da Ciência da Computação, Universidade Técnica de Munique, Munique, 1991. Disponível em: <http://people.idsia.ch/~juergen/SeppHochreiter1991ThesisAdvisorSchmidhuber.pdf>. Acesso em: 15 de Março de 2018.

[13] BENGIO, Yoshua; SIMARD, Patrice; FRASCONI, Paolo. "Learning Long-Term Dependencies with Gradient Descent is Difficult". IEEE Transactions on Neural Networks, v.5, n.2, pp.157-166 1994. doi: 10.1109/72.279181 Disponível em: < https://ieeexplore.ieee.org/document/279181/ >. Acesso em: 15 de Março de 2018.

[14] KALCHBRENNER, Nal; DANIHELKA, Ivo; GRAVES, Alex. "Grid Long Short-Term Memory", 3ed., 2015. Disponível em: <https://arxiv.org/abs/1507.01526>. Acesso em: 16 de Março de 2018.

[15] BAYER, Justin; OSENDORFER, Christian. "LEARNING STOCHASTIC RECURRENT NETWORKS", 3 ed., 2015. Disponível em: <https://arxiv.org/pdf/1411.7610v3.pdf>. Acesso em: 16 de Março de 2018.

[16]_____, Google. Disponível em: <https://developers.google.com/maps/documentation/geocoding/start>. Acesso em: 07 de Janeiro de 2018.

[17] MOOCARME, Matthew. "COUNTRY LYRICS CREATED WITH RECURRENT NEURAL NETWORKS". Disponível em: <http://www.mattmoocar.me/blog/RNNCountryLyrics>. Acesso em: 14 de Março de 2018.

[18] OLAH, Christopher. "Understanding LSTM Networks", 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 15 de Março de 2018.

[19]_____, Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/products>. Acesso em: 02 de Fevereiro de 2018.

[20]_____, Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/free/#legal>. Acesso em: 02 de Fevereiro de 2018.

[21]_____, Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/ec2/pricing>. Acesso em: 02 de Fevereiro de 2018.

[22]_____, Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/ec2/spot>. Acesso em: 02 de Fevereiro de 2018.

[23]_____, Amazon Web Services. Disponível em: <https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/ec2-reserved-instances.html>. Acesso em: 02 de Fevereiro de 2018.

[24]_____, Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types>. Acesso em: 02 de Fevereiro de 2018.

[25] BROWNLEE, Jason. “How to Convert a Time Series to a Supervised Learning Problem in Python”, 2017. Disponível em: <https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/>. Acesso em: 20 de Fevereiro de 2018.

[26]_____, Python. Disponível em: <https://pypi.python.org/pypi/pyswarm>. Acesso em: 10 de Abril de 2018.

[27]_____, Wikipedia, a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Confusion_matrix>. Acesso em: 17 de Abril de 2018.

[28] SAMUEL, Arthur Lee. “Some Studies in Machine Learning Using the Game of Checkers”, 1959. Cap. 4. p. 535-554. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.368.2254&rep=rep1&type=pdf>. Acesso em: 28 de Março de 2018.

[29] VARGAS, Gustavo Badilla. “Determinación de la regularidad superficial del pavimento, mediante el cálculo del Índice de Regularidad Internacional (IRI)”. Disponível em: < https://revistas.ucr.ac.cr/index.php/vial/article/view/2016 >. Acesso em: 15 de Maio de 2018.