Iran Freitas Ribeiro

MobDeep: um arcabouço para geração de dados de mobilidade urbana utilizando aprendizado profundo

Vitória, ES

Setembro, 2021

Iran Freitas Ribeiro

MobDeep: um arcabouço para geração de dados de mobilidade urbana utilizando aprendizado profundo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Vinícius F. S. Mota

Vitória, ES

Setembro, 2021

Iran Freitas Ribeiro

MobDeep: um arcabouço para geração de dados de mobilidade urbana utilizando aprendizado profundo/ Iran Freitas Ribeiro. – Vitória, ES, Setembro, 2021-

137 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Vinícius F. S. Mota

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, Setembro, 2021.

1. Palavra-chave1. 2. Palavra-chave2. I. Mota, Vinícius F. S. II. Universidade Federal do Espírito Santo. IV. MobDeep: um arcabouço para geração de dados de mobilidade urbana utilizando aprendizado profundo

CDU 02:141:005.7

Iran Freitas Ribeiro

MobDeep: um arcabouço para geração de dados de mobilidade urbana utilizando aprendizado profundo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 03 de setembro de 2021:

Prof. Vinícius F. S. Mota
Orientador

Prof. Rodolfo da Silva Villaça
Examinador Interno

Prof. Antonio Augusto de Aragão Rocha
Examinador Externo

Vitória, ES
Setembro, 2021

À meu pai, Edson Lima Ribeiro que, mesmo com todas as dificuldades, garantiu que eu tivesse acesso à melhor educação possível.

Agradecimentos

Agradeço primeiramente à meus pais, Edson Lima Ribeiro e Marizete Freitas Ribeiro, pelo amor, apoio e incentivo que me fizeram chegar até aqui. Agradeço às minhas irmãs, Sueide e Alice, pelo companheirismo e apoio durante todos esses anos. Agradeço à Flávia A. Conceição, pelo apoio, companheirismo, paciência e conversas que me ajudaram a esquecer da vida acadêmica quando era preciso. Agradeço ao meu amigo e primo, Valdir Carvalho Ribeiro, pela amizade e conversas sobre a vida acadêmica. Agradeço ao meu orientador Dr. Vinícius F. S. Mota pela confiança, conhecimento compartilhando e a orientação durante o período do mestrado. Agradeço aos meus amigos da Universidade Estadual do Sudoeste da Bahia que, mesmo distantes, me fazem sorrir. Agradeço aos integrantes do Laboratório NERDS, pelo espaço disponibilizado para estudo e conhecimento compartilhando durante o mestrado. Agradeço à Universidade Federal do Espírito Santo (UFES) e ao Programa de Pós-Graduação em Informática (PPGI) pela possibilidade de cursar o mestrado. Agradeço, também, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa concedida durante o mestrado. Por fim, agradeço à Prefeitura de Vitória-ES, que disponibilizou os dados do WAZE utilizados nesta dissertação.

Resumo

Entender a mobilidade dos componentes de redes móveis é fundamental para diversos tipos de redes, como redes sem fio, redes veiculares ou ad hoc. Nesse sentido, o estudo e acesso a dados de mobilidade urbana se mostram imprescindíveis para avaliar o desempenho dessas redes. O impacto da mobilidade é investigado por meio de modelos sintéticos ou dados de mobilidade reais. Embora modelos sintéticos tentem reproduzir características reais de mobilidade, podem não refletir o realismo do cenário estudado. O acesso e divulgação de dados de mobilidade urbana é limitado por desafios na coleta dos dados, tratamento de informações faltantes e garantias de privacidade. Uma alternativa a esse problema é a geração de dados sintéticos, a partir de dados reais, que possam preservar as características dos dados enquanto mantém a sua privacidade. Considerando que dados de mobilidade urbana são altamente dependentes do tempo, é possível tratá-lo como séries temporais. Assim, neste trabalho propomos o MobDeep, um arcabouço baseado em aprendizado profundo para geração e avaliação de modelos de séries temporais de mobilidade urbana. Neste trabalho, utilizamos um modelo estatístico (ARIMA) e três modelos baseados em aprendizado profundo (GANs) para simulação das séries temporais. Para validar a solução proposta, os modelos são treinados com duas bases de dados: uma base aberta, com informações sobre locações de bicicletas em cidades dos Estados Unidos; e; uma base privada que possui informações sobre o trânsito da cidade Vitória-ES, que são reportadas pelos usuários do aplicativo WAZE. Os modelos treinados são usados para geração de bases sintéticas que, posteriormente, têm seu desempenho avaliado por meio de análises qualitativas e quantitativas. Os resultados das avaliações mostram que a solução proposta, utilizando os modelos baseados em aprendizado profundo, consegue gerar dados sintéticos com as mesmas características dos dados reais. Desta forma, os modelos podem ser compartilhados, permitindo a geração de dados sintéticos, preservando a privacidade dos dados originais.

Palavras-chave: Mobilidade urbana. Séries temporais. Redes Generativas Adversárias.

Abstract

Understanding the mobility among mobile network devices is critical for different types of networks, such as wireless networks, vehicular networks, or ad hoc networks. In this sense, the availability of urban mobility data is essential to evaluate these networks' performance. One can investigate the impact of mobility using synthetic models or realistic mobility data. Although synthetic models attempt to reproduce real mobility characteristics, they may not reflect the realism of the studied scenario. Furthermore, the gathering and disseminating of urban mobility data face challenges such as data collection, handling of missing information, and privacy protection. An alternative to tackle this problem is the generation of synthetic data based on the original data, preserving its characteristics while maintaining its privacy. Considering that urban mobility data are highly time-dependent, they may be represented as time series. Thus, in this work, we propose MobDeep, a deep learning-based framework to generate and evaluate urban mobility time series models. In this work, we use a statistical model (ARIMA) and three deep learning-based models (GANs) to simulate time series. We validate the proposed solution using an open dataset that contains information about bicycle rentals in US cities and a private dataset that contains information about the urban traffic in Vitória-ES, reported by the WAZE users. The evaluation results show that the proposed solution using deep learning-based models can generate synthetic data with the same characteristics as the real ones. With this approach, the models can be shared, allowing the generation of synthetic data and preserving the privacy of the original dataset.

Keywords: Urban mobility. Time series. Generative Adversarial Networks.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CDR	*Call Detail Records*
CEP	Código de Endereçamento Postal
CGAN	*Conditional Generative Adversarial Networks*
CNN	*Convolutional Neural Network*
DCGAN	*Deep Convolutional Generative Adversarial Networks*
GAN	*Generative Adversarial Networks*
GCN	*Graph Convolutional Networks*
GPS	*Global Positioning System*
GPU	*Graphics Process Unit*
GRU	*Gated Recorrent Unit*
IA	Inteligência Artificial
LGPD	Lei Geral de Proteção de Dados
LSTM	*Long Short-Term Memory*
SMS	*Short Message Service*
UFES	Universidade Federal do Espírito Santo
RNN	*Recurrent Neural Networks*

Sumário

1 Introdução

O desempenho de novas tecnologias e protocolos em redes móveis, como redes móveis sem fio, redes veiculares e redes *ad hoc*, está diretamente relacionado à mobilidade dos participantes que a compõem. A mobilidade pode ser avaliada por meio de modelos de mobilidade sintéticos e/ou dados de mobilidade reais (*traces*) (MOTA et al., 2014). Embora modelos sintéticos permitam uma maior escalabilidade e repetibilidade, o seu uso possui restrições em relação ao realismo dos dados utilizados. Por outro lado, *traces* contêm informações sobre a mobilidade de um conjunto de participantes da rede durante um período de tempo. Um *trace* pode conter a trajetória dos participantes, seja por meio de uso de posicionamento via satélite (GPS) ou transição entre estações bases de uma rede sem fio, a cada período (ZHENG et al., 2009; Malandrino; Chiasserini; Kirkpatrick, 2018), ou pontos de chegada e partida.

Contudo, apesar de *traces* representarem comportamentos mais realistas de mobilidade, a sua utilização é dificultada por alguns fatores, como a dificuldade em se obter esses dados, erros e dados faltantes durante a coleta, e a privacidade dos dados. Dados de mobilidade podem ser gerados por diversas fontes como censos demográficos, telefonia de móvel e de serviços baseados em geolocalização.

Os dados de telefonia móveis ou *Call Detail Record* (CDR, em Inglês) são gerados quando os usuários quando realizam alguma chamada ou enviam uma mensagem. Apenas as operadoras de telefonia têm acesso a esses dados e podem disponibilizá-los, por exemplo, para estudos. Contudo, podem limitar quais informações serão disponibilizadas com base em políticas de privacidade da própria operadora ou leis de proteção de privacidade (BLONDEL; DECUYPER; KRINGS, 2015), como a Lei Geral de Proteção de Dados (LGPD) (BRASIL, 2018) do Brasil.

Por outro lado, serviços baseados em geolocalização, como o *Twitter*[1], *Forsquare*[2] e o Waze[3] disponibilizam APIs com acesso à dados que contém, entre outras informações, a localização dos seus usuários ou eventos em determinados pontos. O Twitter e Foursquare oferecem acesso sem custo aos serviços, contudo, fica limitado uma quantidade reduzida dos dados e tipo de informações que serão acessadas. Já o Waze oferece serviços específicos de acesso aos dados por meio de acordos. A comunidade científica tem feito um esforço na disponibilização de bases públicas de mobilidade em repositórios, como o *Crawdad*[4], e

[1] www.twitter.com
[2] www.forsquare.com
[3] www.waze.com
[4] https://www.crawdad.org/

com conferências especificas, como a *Netmob*[5].

Nos dados de mobilidade de urbana disponíveis publicamente, um outro fator que dificulta a sua utilização são erros e dados faltantes. Erros podem ser, por exemplo, mal funcionamento de algum sensor na coleta dos dados ou informações que não seguem o padrão que era esperado (um registro que deveria ser do tipo numérico foi salvo como texto, por exemplo). Dados faltantes representam porções dos dados em que simplesmente não há registros para serem analisados. Dependendo da quantidade de dados faltantes, pode se tornar uma situação difícil de contornar, mesmo com técnicas interpolação.

As questões relativas à privacidade dos dados é o terceiro fator que dificulta a divulgação dos dados. Tanto dados fornecidos pelas redes de telefonia móvel quanto dos serviços com informações de geolocalização possuem restrições na forma como os dados podem ser divulgados. Segundo Machado e Neto (2021), existem 3 diferentes categorias de métodos que podem ser usados para fornecer privacidade à dados de mobilidade urbana: anonimização, ofuscação e privacidade diferencial. O desafio, ao se utilizar esses métodos, é prover proteção à privacidade dos dados sem prejudicar a sua usabilidade. Dados com informações de geolocalização, por exemplo, são essenciais para serviços que oferecem valor aos usuários como serviços de navegação, serviços de recomendação e aplicações de tempo (MACHADO; NETO, 2021).

Como alternativa para possibilitar que dados de mobilidade reais possam ser divulgados, uma abordagem é modelá-los de forma que dados sintéticos possam ser gerados a partir de bases de dados de mobilidade reais. Para isso, este trabalho investigará o uso de algoritmos de inteligência artificial, em especial aprendizado profundo.

Entre os modelos de Aprendizado Profundo mais conhecidos estão o Autoenconder (RUMELHART; HINTON; WILLIAMS, 1985), a Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997), as Máquinas Restritas de Boltzman (RBM, em Inglês) (HINTON, 2002), Redes Neurais Convolucionais (CNN, em Inglês) (LECUN; BENGIO et al., 1995), e as Redes Generativas Adversárias (GANs) (GOODFELLOW et al., 2014a). As GANs, especificamente, são um arcabouço desenvolvido por Goodfellow et al. (2014a) para a otimização de modelos generativos e têm sido utilizadas larga e eficientemente no campo de visão computacional (ISOLA et al., 2017; LEDIG et al., 2017; ZHU et al., 2017; BROCK; DONAHUE; SIMONYAN, 2018). Alguns exemplos de aplicações diretas das GANs incluem a geração de objetos 3D (YU et al., 2020b; MA et al., 2020), medicina (BAEK; KIM; KIM, 2020; TERAMOTO et al., 2020), processamento de imagem (ZHOU et al., 2020; GO et al., 2020), detecção de rostos (ZHAO et al., 2018; MOKHAYERI; KAMALI; GRANGER, 2020) controle de tráfego (XU et al., 2020; BEERY et al., 2020), entre outros.

[5] https://netmob.org/

Recentemente, especificamente no campo de mobilidade urbana, as GANs tem sido usadas na geração de trajetórias de pessoas (GUPTA et al., 2018; SONG; BAEK; SUNG, 2019) e veículos (ZHANG et al., 2020), para geração de bases de dados de pedidos de carona (JAUHRI et al., 2020) e para bases de dados em que representam algum tipo de rede (contatos entre pessoas ou dispositivos móveis, por exemplo) (LEI et al., 2019). Nesse sentido, Qu et al. (2020) demostram que as GANs são eficientes em gerar dados de mobilidade que garantem a privacidade dos dados utilizados.

1.1 Problema

A principal problemática abordada por este trabalho trata de como gerar dados de mobilidade a partir de uma base de dados real, preservando suas características, mas com variabilidade. Formalmente, considerando que os dados de mobilidade urbana podem ser analisados como séries temporais, o desafio é encontrar um modelo capaz de simular a série temporal original. Para tratar esta problemática, serão estudados os desafios discutidos nas próximas subseções.

1.1.1 Privacidade e qualidade dos dados

O principal fator que limita a utilização de dados de mobilidade urbana é a privacidade dos usuários. Uma empresa que armazena algum tipo de dados de mobilidade, como operadoras de telefonia, pode ter suas próprias políticas referentes à privacidade dos dados (BLONDEL; DECUYPER; KRINGS, 2015). A disponibilização desses dados por parte da empresa pode ser limitada por leis da região onde ela opera como, por exemplo, a LGPD (BRASIL, 2018) do Brasil.

Além disso, as informações presentes em uma base de dados podem permitir que pessoas mal intencionadas obtenham conhecimentos sensíveis dos usuários, sendo imprescindível, durante a geração de bases sintéticas, que a privacidade seja mantida. Por exemplo, com o conhecimento prévio de alguma informação presente na base de dados (nome do usuário) uma pessoa mal intencionada poderia (MACHADO; NETO, 2021):

- Cruzar informações com outras bases de dados disponíveis e obter conhecimento sobre novos atributos do usuário;
- Identificar o grupo à qual o usuário faz parte, por exemplo, com base no Código de Endereçamento Postal (CEP), inferir seu estado ou cidade;
- Verificar se um usuário está presente ou não em uma base de dados que foi publicada;
- Com base em informações analisadas de uma base de dados, inferir as características de algum usuário (a probabilidade de ele ser do sexo masculino, por exemplo)

Por outro lado, além de ter a privacidade mantida, os dados gerados precisam continuar úteis, no sentido de possuir as características principais dos dados reais. Por exemplo, considerando fins acadêmicos, deve ser possível realizar estudos a partir dos dados gerados da mesma forma que seria feito com a base de dados original. Da mesma maneira, para fins práticos, como para algum tipo de serviço que é baseado na mobilidade dos usuários, os dados sintéticos precisam estar condizentes com a realidade, permitindo que o serviço opere normalmente. O desafio, nesse cenário, é encontrar maneiras de se gerar dados de forma a manter o equilíbrio entre privacidade e utilidade dos dados (MACHADO; NETO, 2021).

1.1.2 Poder de generalização das soluções atuais

No campo da visão computacional, principalmente em relação às imagens, as GANs se destacaram pela eficiência demonstrada na geração de novos dados sintéticos de qualidade. Tendo em vista a eficiência de tais modelos, algumas pesquisas buscaram a aplicação de GANs para o problema de geração de dados de mobilidade urbana (SONG; BAEK; SUNG, 2019; JAUHRI et al., 2020; ZHANG et al., 2020). Muito embora as propostas tenham apresentado sucesso, são limitadas em relação às possibilidades de serem generalizadas para outros tipos de problemas, por exemplo, uma base de dados com características distintas das bases usadas nas propostas apresentadas.

Como estão diretamente relacionados à rotina das pessoas, dados de mobilidade urbana serão influenciados por essa rotina. De fato, Song et al. (2010), analisando trajetórias feitas por indivíduos, mostraram que a mobilidade humana é extremamente previsível e estável, e são dependentes do tempo (BROCKWELL; DAVIS, 2009). A vantagem de se utilizar séries temporais, nesse contexto, é que diferentes cenários de mobilidade podem ser tratados como observações ao longo do tempo, como mobilidade e conectividade de veículos (ALJERI; BOUKERCHE, 2020), a mobilidade de indivíduos em período de pandemia (VANNONI et al., 2020), a relação entre relações sociais e a previsibilidade do movimento de pessoas (DOMENICO; LIMA; MUSOLESI, 2013) anomalias em ruas (THOMÉ et al., 2020), entre outros. Dessa forma, encontrar modelos adequados para a modelagem de séries temporais que possam gerar dados de mobilidade urbana é fundamental.

1.1.3 Avaliação dos modelos

Saber como um determinado modelo se saiu durante a realização de uma determinada tarefa é essencial para se identificar quais modificações devem ser feitas para se chegar ao resultado desejado. Boa parte dos modelos de aprendizado profundo utilizam alguma métrica numérica, por exemplo acurácia, que representa o seu desempenho durante a realização de uma tarefa (CHOLLET et al., 2018).

Para o problema de geração de dados, dois requisitos principais surgem durante a avaliação: os dados sintéticos precisam ter as mesmas características dos dados reais e precisam possuir variabilidade. Para o primeiro requisito, por exemplo, uma GAN que foi treinada à partir de uma base de dados composta por rostos de pessoas, precisa gerar formas que lembrem minimamente um rosto.

No segundo, é esperado que cada nova base sintética gerada seja diferente entre si. Isso garante que o modelo utilizado não sofreu um sobre-ajuste (do Inglês, *overfitting*), do contrário os dados sintéticos gerados seriam sempre os mesmos. Ademais, considerando a questão de privacidade anteriormente mencionada, cada base sintética gerada precisa ser diferente dos dados presentes na base real. Isso garante que o modelo não está reproduzindo uma cópia de uma amostra dos dados e, sim, que aprendeu as suas principais propriedades.

Considerando os requisitos mencionados, podemos formular a questão: como avaliar a eficiência dos modelos que serão utilizados para a geração de dados de mobilidade urbana?

1.2 Motivação

Dados de mobilidade urbana podem ser obtidos de diversas fontes, como censos demográficos e pesquisas, rastreio de notas monetárias, registros de telefonia móvel, dados de GPS e serviços online, como redes (BARBOSA et al., 2018). Com o rápido aumento da urbanização aumenta-se também a proporção da população vivendo em cidades (BARBOSA et al., 2018). Da mesma maneira, serviços que geram dados com maior precisão em relação a localização das pessoas tornam-se cada vez mais comuns, produzindo um volume maior de dados de mobilidade.

De acordo com o relatório anual de internet da Cisco (CISCO, 2018), em 2023 mais de 70% da população mundial estará conectada à internet através de dispositivos móveis, totalizando ≈ 5.7 bilhões de pessoas. Em 2018, esse número era de 66%. Ainda segundo o mesmo relatório, de todos os dispositivos conectados à internet em 2023 (13.1 bilhões), 1.4 bilhões (10%) terão suporte à tecnologia 5G. O aumento do número de dispositivos móveis implica em um também crescente volume de dados de mobilidade humana sendo gerado. Dessa forma, compreender a mobilidade das pessoas é fundamental para atender esse crescente número de dispositivos conectados, tanto em relação à infraestrutura de redes que se fará necessária para as conexões quanto para os protocolos de comunicação utilizados nos dispositivos.

Além disso, o acesso e estudo de dados da mobilidade urbana possibilita estudos em diferentes cenários e com finalidades. Na área de redes, por exemplo, estudos analisam a formação de comunidades de participantes com base em interesses em eventos científicos (RIBEIRO et al., 2020) e avaliam o desempenho de algoritmos de roteamento

oportunístico em uma rede formada por contatos em eventos científicos (RIBEIRO et al., 2021). Considerando a movimentação de indivíduos, alguns trabalhos utilizam dados de mobilidade urbana para identificar melhorias na infraestrutura da cidade (HILLIER et al., 2009; KITAMURA et al., 2000), analisar o comportamento das pessoas durante fugas em prédios com pouca visibilidade (HELBING; FARKAS; VICSEK, 2000), avaliar como desastres naturais podem ser mitigados através do controle de multidões (HELBING; JOHANSSON; AL-ABIDEEN, 2007; JOHANSSON et al., 2008), analisar a propagação de doenças como SARS (BAUCH et al., 2005), malária (HUANG; TATEM, 2013), Ebola (JONES et al., 2008; GOMES et al., 2014) e Covid-19 (KRAEMER et al., 2020).

Apesar da grande e crescente disponibilidade de dados de mobilidade e sua evidente importância para diversas áreas da vida urbana e infraestrutura das cidades, a utilização desses dados, até mesmo para pesquisas com fins acadêmicos, é limitada devido às informações privadas presentes nos dados. Uma alternativa, nesse cenário, é utilizar métodos capazes de gerar dados sintéticos realistas, com base nos dados reais, que garantam a privacidade dos dados utilizados. Nesse sentido, uma possibilidade é a utilização de modelos generativos baseados em aprendizado profundo, como as GANs, que são eficientes na geração de dados de qualidade e com privacidade. Esses modelos geralmente possuem um alto custo de treinamento e necessitam de métodos eficientes de avaliação do desempenho. Dessa forma, faz-se necessário a busca por técnicas que facilitem a utilização e avaliação de modelos para a geração de dados de mobilidade urbana.

1.3 Objetivos

O objetivo principal deste trabalho é propor métodos para avaliar modelos generativos e dados sintéticos de mobilidade urbana baseados em aprendizado profundo. Especificamente, para alcançar o objetivo principal, os métodos propostos abordam os seguintes problemas:

- **Poder de generalização das soluções atuais**: as técnicas utilizadas para gerar dados sintéticos precisam ser capazes de abordar situações diferentes de mobilidade.
- **Privacidade e qualidade dos dados**: é fundamental que as técnicas usadas sejam capazes de gerar dados com as mesmas características dos dados reais e tenha capacidade de prover privacidade aos dados gerados.
- **Avaliação dos modelos**: considerando o item anterior, é necessário avaliar os modelos em relação à sua capacidade de gerar dados sintéticos. Nesse sentido, a avaliação deve verificar se os modelos capturaram as características dos reais e se os dados gerados possuem variabilidade.

1.4 Contribuições

As principais contribuições desta dissertação são sumarizadas a seguir:

- Analisamos a eficácia de modelos estocásticos clássicos (nesta dissertação, o ARIMA) na modelagem de séries temporais para bases de dados de mobilidade. Estes modelos podem exigir um menor poder computacional para modelar os dados.
- Propomos a utilização de modelos baseados em aprendizado profundo para a geração de séries temporais de mobilidade urbana. Nesta dissertação, utilizamos modelos de GANs, que enquanto eficientes para aprender as principais características dos dados reais, fornecem privacidade aos dados gerados.
- Discutimos como quantificar o desempenho dos modelos treinados, por meio de avaliações qualitativas e quantitativas dos dados sintéticos gerados.
- Considerando que não existe um modelo universal que apresenta melhor resultado em todos os cenários, propomos o MobDeep, um arcabouço que para treinamento de modelos para geração de dados de mobilidade e avaliação dos modelos treinados.

1.4.1 Lista de publicações

Os resultados parciais desta dissertação foi publicado em:

- RIBEIRO, I. et al. Uma Abordagem para Geração de Séries Temporais de Mobilidade Urbana Utilizando Aprendizado Profundo. *Anais do V Workshop de Computação Urbana*. Uberlândia, MG, Brasil: SBC, 2021

Adicionalmente, por meio de investigações de dados de mobilidade, foi realizado um estudo da geração de dados de mobilidade baseado em tópicos de interesse e uma avaliação de redes adhoc móveis utilizando estes dados em roteamento oportunístico. Nesse sentido, destacamos à seguir, as publicações que serviram como base para os estudos iniciais desta dissertação.

1. RIBEIRO, I. et al. Caracterização de Mobilidade e Detecção de Comunidades baseadas em Tópicos de Interesse. *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2020. p. 603-616. ISSN 2177-9384.
2. RIBEIRO, I. et al. Mobility and community detection based on topics of interest. *IEEE Annual Consumer Communications & Networking Conference* (CCNC). [S.l.], 2021. p. 1–6.

1.5 Organização do Texto

O restante deste texto está organizado como segue. O Capítulo 2 discute os principais abordados e o Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 descreve a solução proposta seguido do Capítulo 5 que apresenta as bases de dados utilizados e suas principais propriedades. A avaliação da solução proposta é explicitada no Capítulo 6. Por fim, o Capítulo 7 conclui a dissertação e discute os trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos abordados nesta dissertação. Primeiramente conceituamos as séries temporais, por serem fundamentais para a realização dos experimentos e entendimento dos resultados, bem como de suas representações estatísticas. Em seguida apresentamos os principais conceitos de Inteligência Artificial, Redes Neurais, Aprendizado Profundo e Redes Generativas Adversárias (GANs), que são a base da solução proposta neste trabalho. Finalmente, introduzimos o conceito de modelagem de séries temporais considerando as técnicas estatísticas e com base em aprendizado profundo.

2.1 Séries Temporais

As séries temporais são categorizadas são um conjunto de observações x_t registradas em um tempo t específico (BROCKWELL; DAVIS, 2009). Uma série temporal pode representar vários tipos de situações reais e sua aplicação pode ser encontrada em diversos campos científicos, como economia (GRANGER; NEWBOLD, 2014), ciência sociais (SALEHYAN; GLEDITSCH, 2006), epidemiologia (BHASKARAN et al., 2013) e medicina (PINCUS; GOLDBERGER, 1994).

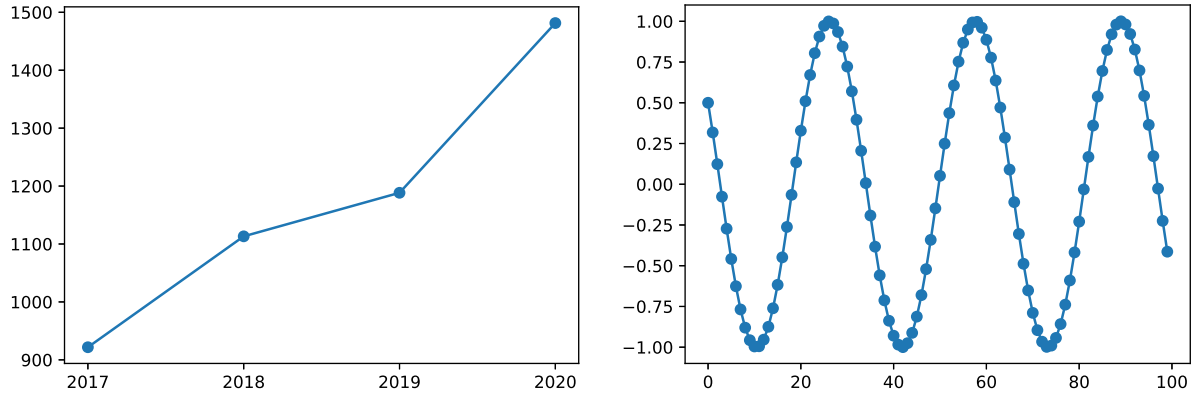
Nesse sentido, uma série temporal pode ser discreta, quando o conjunto T_0 de intervalos em que as observações acontecem são fixos, ou contínua, quando as observações são registradas continuamente em um intervalo de tempo variável, por exemplo, $T_0 = [0, 1]$. A Figura 1a é um exemplo do primeiro caso de série temporal, onde é possível ver a média da cotação, em dólares, das ações da Google[1] entre 2017 e 2020. A Figura 1b, que mostra a corrente que passa por um resistor, é um exemplo de série temporal contínua.

Uma série temporal pode possuir quatro propriedades principais, que Chatfield (2003) define como variações, que auxiliam no seu entendimento. Essas variações estão diretamente relacionadas à situação real que gera a série temporal, seja por efeito natural, humano ou ambos. As quatro propriedades são brevemente descritos a seguir:

- **Sazonalidade**: são variações nas observações dos dados que ocorrem em um certo período de tempo (ou temporadas). Uma série temporal contendo a temperatura de um país é um exemplo de série temporal com sazonalidade, com variação anual (o inverno tende à ser mais frio e verão mais quente).

- **Ciclos**: é um tipo de variação similar à sazonalidade, que geralmente é causada por processos externos à natureza da série temporal. Diferente da sazonalidade, os ciclos

[1] Disponível em https://finance.yahoo.com/quote/GOOG/history?p=GOOG



(a) Discreta (b) Contínua

Figura 1 – Exemplos de séries temporais discreta (a), mostrando o preço das ações da Google, em dólares, e contínua (b), mostrando a corrente que passa por um resistor. A figura (b) é adaptada de Brockwell e Davis (2009).

podem ocorrer por tempos variados e em intervalos de tempo distintos. As variações de temperatura durante um dia são um exemplo de variação cíclica.

- **Tendência**: a tendência pode ser definida como uma mudança na média da série temporal ocorrendo por um longo período de tempo. Essa variação pode fazer com que a média da série temporal sofra um aumento (tendência positiva), uma diminuição (tendência negativa) ou nula (tendência horizontal).

- **Irregularidades**: são alterações aleatórias nos dados, sem nenhum padrão específico. Geralmente são variações de curta duração.

A série da Figura 1a, representando o valor das ações da Google, apresenta um exemplo de uma série temporal com tendência linear, indicando que houve uma valorização das ações. A Figura 2 mostra o volume de vendas, em milhões de dólares, no varejo dos EUA, entre fevereiro 1992 e maio de 2021[2]. Este é um exemplo em que há uma combinação de duas propriedades: a tendência de crescimento e a sazonalidade, que corresponde à uma similaridade nas observações em um mesmo intervalo de tempo. No caso da Figura 2, os picos observados representam o mês de dezembro.

2.2 Modelos Estatísticos

De um ponto de vista estatístico, uma série temporal pode ser entendida como uma sequência de observações, que assim como muitos outros processos do mundo real, são aleatórios, e podem possuir propriedades probabilísticas. Nas próximas seções serão apresentados alguns desses processos estatísticos que definem boa parte das séries temporais.

[2] Disponível em https://fred.stlouisfed.org/series/RSXFSN

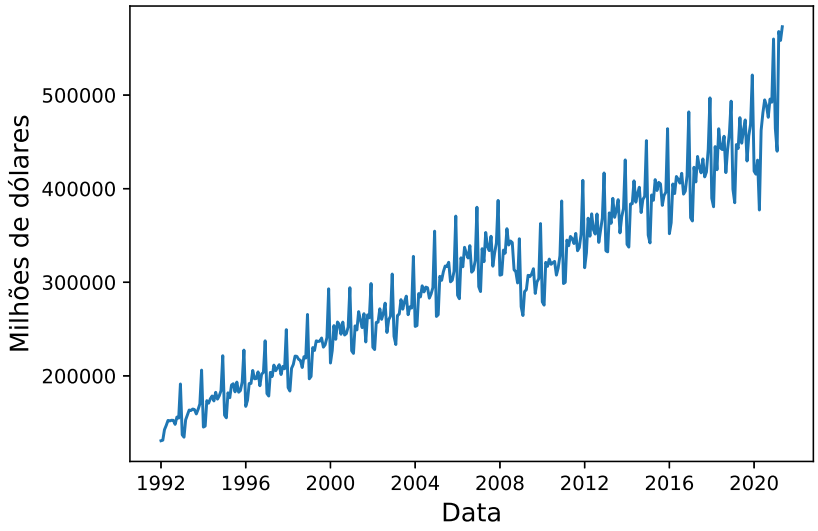


Figura 2 – Vendas no varejo nos EUA, em milhões de dólares, entre fevereiro de 1992 e maio de 2021

2.2.1 Série Temporal Como Processos Estocásticos

Um processo estocástico é um conjunto de variáveis aleatórias $\{X_t\}$ indexadas por t onde t, geralmente, é um valor discreto que varia nos intervalos dos números inteiros $(t = 0, \pm1, \pm2, ...)$ (SHUMWAY; STOFFER, 2000). É importante ressaltar a notação utilizada por vários autores, em que uma variável aleatória observada no intervalo t será $X(t)$, para intervalos contínuos, e X_t, para intervalos discretos. Os intervalos, nesse caso, estão diretamente relacionados à categorização de uma série temporal, discutida anteriormente, como sendo discreta ou contínua.

Uma forma de se descrever um processo estocástico é especificar a distribuição da probabilidade conjunta de $X_1, ..., X_n$, embora seja uma opção não muito usada na prática. Uma alternativa mais simples e útil é definir-se os momentos do processo (nesse caso, o primeiro e segundo momentos), que são as funções de média, variância e auto-covariância (CHATFIELD, 2003):

- Média: A função de média μ_t é definida por

$$\mu_t = E[X_t] \tag{2.1}$$

- Variância: A função de variância $\sigma^2(t)$ é definida por

$$\sigma_t^2 = Var[X_t] \tag{2.2}$$

- Auto-covariância: A função de auto-covariância $\gamma(t_1, t_2)$ é definida por

$$\gamma(t_1, t_2) = E[X_{t_1} - \mu_{t_1}][X_{t_2} - \mu_{t_2}] \tag{2.3}$$

Segundo Chatfield (2003), apenas a função de variância não é suficiente para especificar o segundo momento do processo estocástico e, por isso, a função de auto-covariância precisa ser usada. Além disso, as equações demostradas anteriormente referem-se à um processo de tempo discreto, mas tem definições similares para tempos contínuos.

2.2.2 Série Temporal Como Processos Estacionários

Uma das classes mais importantes de processos estocásticos são os processos estacionários. Há diferentes tipos de níveis de restrições para que um processo seja considerado estacionário. Uma série temporal será estritamente estacionária se as distribuições conjuntas de $X_{t_1}, ..., X_{t_n}$ e $X_{t_1+\tau}, ..., X_{t_n+\tau)}$ são iguais, ou seja, as distribuições não dependem do tempo t ou de qualquer deslocamento τ aplicado à série temporal. Essa definição, entretanto, possui um nível de restrição muito alto, sendo comum a utilização de uma definição com menos restrições, conhecida como estacionária de segunda ordem. Nessa definição, uma série temporal será estacionária quando sua média for constante e sua função de auto-covariância dependa somente da diferença entre intervalos distintos $(t_2 - t_1)$, também conhecido como *lag*, ou seja

$$E[X_t] = \mu$$

e

$$Cov[X_t, X_{t+\tau}] = \gamma(\tau)$$

A análise estatística das propriedades de uma série temporal permite a identificação da categoria de processo estocástico da qual ela faz parte. Assim, alguns dos processos mais comuns que auxiliam na classificação e modelagem de séries temporais são (CHATFIELD, 2003):

- **Puramente aleatório**: um processo que consiste de uma sequência de variáveis aleatórias $\{Z_t\}$ que são mutuamente independentes e identicamente distribuídas. Esse tipo de processo possui média e variância constantes e, como sua função de auto-covariância não depende do tempo, será estacionário de segunda-ordem.

- **Passeio Aleatório**: Supondo-se que $\{Z_t\}$ seja um processo puramente aleatório com média μ e variância σ_z^2. Um processo $\{X_t\}$ será um passeio aleatório se $X_t = X_{t-1} + Z_t$. Nesse caso, a média e a variância são dependentes do tempo e o processo é não-estacionário.

- **Média Móvel**: Considerando que $\{Z_t\}$ seja um processo aleatório com média 0 e variância σ_z^2, um processo $\{X_t\}$ será de médias móveis de ordem q, representado por MA(q) (do inglês, *Moving Average*), se:

$$X_t = \beta_0 Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q}$$

onde B_i, com $i = 0, ..., q$ são constantes. Esse processo é estacionário pois sua função de auto-covariância não depende de do tempo t e a média é constante.

- **Auto-regressivo**: Sendo $\{Z_t\}$ um processo puramente aleatório com média 0 e variância σ_z^2, um processo $\{X_t\}$ será auto-regressivo de ordem p se:

$$X_t = \alpha_1 X_{t-1} + ... + \alpha_p X_{t-p} + Z_t$$

A função de auto-covariância para esse tipo de processo não depende do tempo t, logo ele é estacionário de segunda-ordem para os casos em que $|\alpha| < 1$. Assim como em um processo de médias móveis, um processo auto-regressivo é normalmente abreviado para AR(p) (*Auto-Regressive*, na sigla em Inglês).

Considerando os tipos de processos apresentados, é comum a definição de métodos capazes de modelar mais de um processo simultâneamente. Como foi visto, um processo Puramente Aleatório é usado na definição dos processo de Passeio Aleatório, Média Móvel e Auto-regressivos. Além disso, uma dos tipos de modelos mais úteis para séries temporais é resultado da combinação de um processo Auto-regressivo (AR) e um processo de Médias Móveis (MA), resultando em um modelo conhecido como ARMA. Desta forma, um processo ARMA de ordem (p,q), abreviado para ARMA(p,q), é definido por:

$$X_t = \alpha_1 X_{t-1} + ... + \alpha_p X_{t-p} + Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q}$$

A vantagem da utilização deste tipo de modelo é que mais comum que uma série temporal estacionária seja descrita por um processo ARMA contendo poucos parâmetros do que por um processo MA ou AR sozinhos (CHATFIELD, 2003).

A maioria das séries temporais, na prática, não são estacionárias, e exigem que, antes de se tentar usar algum método para modelá-la, remova-se o que a torna não-estacionária. Levando isso em consideração, desenvolveu-se um modelo, derivado do ARMA, capaz de tratar de certos tipos de séries não-estacionárias. Tal modelo, conhecido como ARIMA, será descrito com mais detalhes na Seção 2.2.3, a seguir.

2.2.3 ARIMA

O ARIMA (Auto-Regressive Integrated Moving Average) refere-se a um modelo estatístico linear bastante conhecido na análise de séries temporais (ZHANG, 2003). Ele é uma generalização de um outro modelo também bastante conhecido, o ARMA (*AutoRegressive Moving Average*), e é utilizado em séries temporais não-estacionárias, ou seja, uma série cujos dados oscilam sobre uma média que pode variar ao longo do tempo.

Um modelo ARIMA é definido utilizando-se os parâmetros p, d e q, onde p representa o número de parâmetros auto-regressivos, d o número de diferenciações para tornar a série estacionária e q o número de parâmetros utilizado nas médias móveis. Um modelo ARIMA definido pelos parâmetros p, d e q é comumente representado por ARIMA(p,d,q).

Existem diversas metodologias que podem ser utilizadas para se encontrar os parâmetros p, d e q de um modelo ARIMA, sendo o de Box & Jenkins (BOX et al., 2015) o mais conhecido deles. Nessa metodologia, existem 3 etapas fundamentais (Identificação do modelo, Estimação dos parâmetros e Verificação do modelo), descritas a seguir.

- **Identificação**

  Nesta fase, identifica-se o processo aleatório que gerou os dados. Para isso, algumas técnicas são utilizadas para se encontrar a estrutura do modelo, ou seja, os parâmetros p, d, e q.

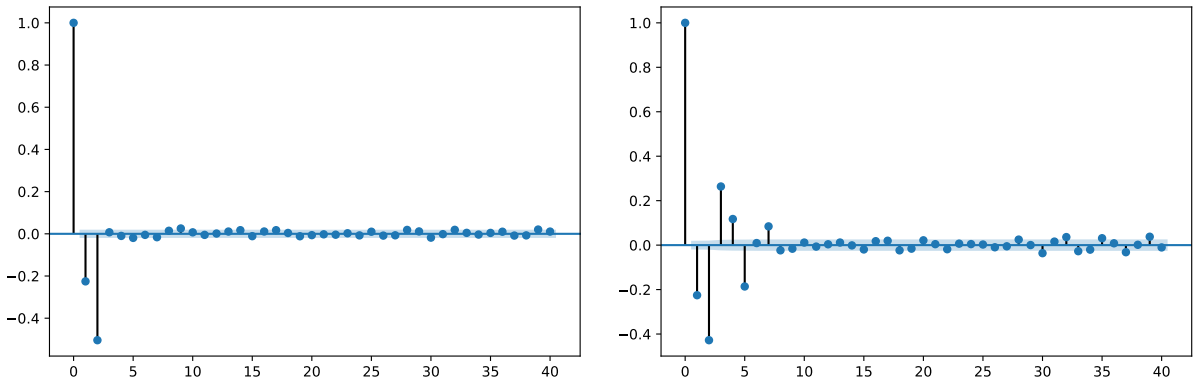
  O parâmetro d (a parte integrada) pode ser identificada por meio de uma sequência de diferenciações aplicadas à série temporal para torná-la estacionária. O número de vezes que a diferenciação precisa ser realizada de define o valor de d.

  Os parâmetros p e q podem ser identificados ao se analisar, respectivamente, as funções de autocorrelação parcial e de autocorrelação (respectivamente, PACF e ACF, nas siglas em Inglês) utilizando-se a série que já passou pelo processo de diferenciação para se tornar estacionária. Cada uma dessas funções exibe a correlação entre a série temporal e um certo número de defasagens. Assim, elas representam como os coeficientes de correlação se comportam em diferentes números de defasagens.

  Para fins de exemplificação, a Figura 3 apresenta exemplos de gráficos PACF e ACF de uma série temporal gerada por um processo ARIMA(2,0,0). Pode-se observar na Figura 3a que, após o número de defasagem 2, os coeficientes são muito próximos de zero 0, indicando que a série pode ser representada por um processo com $p = 2$. Além disso, a Figura 3b mostra que os valores dos coeficientes decrescem à medida que se aumenta o número de defasagens (eixo X), indicando que é improvável que a série seja gerada por um processo de médias móveis, ou seja, $d = 0$.

- **Estimação**

  Nesta etapa utiliza-se os parâmetros identificados na etapa anterior com algum procedimento que estime uma métrica desejada, por exemplo, funções de verossimilhança e de somas dos quadrados, estimação não-linear, análise dos processos específicos (caso seja verificado que a série é gerada por um processo auto-regressivo, de médias-móveis ou uma combinação de ambos) e estimação usando o Teorema de Bayes. Uma descrição mais detalhada de tais procedimentos pode ser encontrada em Box et al. (2015).



(a) PACF (b) ACF

Figura 3 – Gráficos PACF (a) e ACF (b) de um processo ARIMA$(2, 0, 0)$

- **Verificação**

  Esta etapa consiste na realização de testes estatísticos para avaliar o desempenho do modelo ARIMA ajustado à série temporal. A ideia geral é identificar se o modelo definido está adequado para a série em questão e quais modificações no modelo precisam ser realizadas para garantir uma melhor adequação.

  Um dos testes bastante conhecidos é a análise dos resíduos, em que verifica-se,se os resíduos do modelo ajustado apresentam algum tipo de dependência temporal, o que pode ser feito através dos gráficos de autocorrelação parcial anteriormente citados. Caso seja identificada qualquer tipo de dependência, significa que o modelo não foi capaz de capturar corretamente as dependências temporais da série original.

A metodologia de Box & Jenkis é largamente utilizada para definição e escolha dos parâmetros de um modelo ARIMA. Entretanto, existem alternativas mais rápidas e livres de má interpretação das características da série temporal, que usam a metodologia de Box & Jenkins internamente. Por exemplo, em alguns casos, não é simples identificar se uma série é estacionária e, caso seja, quantas vezes a diferenciação precisa ser feita para torná-la estacionária. Dessa forma, algumas linguagens de programação, como o Python e R, possuem bibliotecas que facilitam as análises preliminares temporais e possibilitam a identificação automática dos parâmetros do modelo.

Apesar de seu bom desempenho na modelagem e previsão de séries temporais, o ARIMA possui algumas limitações como, por exemplo, ser mais eficiente em bases de dados pequenas (MONTGOMERY; HINES, 1980). Ainda assim, pela facilidade em se encontrar os parâmetros p, d e q e por ser um modelo relativamente rápido de se treinar, o ARIMA mostra-se um bom modelo para se realizar as primeiras análises dos dados, indicando se um modelo não linear, por exemplo uma Rede Neural Recorrente, precisa ser ou não utilizado, evitando o desperdício de tempo e de poder computacional.

Embora o ARIMA seja comumente utilizado para a previsão de novas observações de séries temporais, também pode ser aplicado em geração de novas séries, em que ajusta-se um modelo a uma série temporal e simula-se uma nova série com base nos parâmetros do modelo que melhor se ajustam à série.

2.3 Inteligência Artificial, Aprendizado de Máquina e Aprendizado Profundo

Em um contexto computacional, a Inteligência Artificial (IA) surgiu por volta do ano 1950 quando, do nascimento do campo de ciência da computação, surgiram também os questionamentos se seria possível fazer com que as máquinas pensassem como humanos (CHOLLET et al., 2018). Apesar de ter surgido já há algumas décadas, não há uma definição exata para a IA. Alguns autores preferem segmentar a definição de IA de acordo com sua aplicação em alguma tarefa (PATTERSON; GIBSON, 2017) específica. Chollet et al. (2018) sugere uma definição mais abrangente, em que a IA seria o esforço em se automatizar tarefas mentais que são normalmente feitas por humanos.

É comum a utilização de IA, Aprendizado de Máquina e Aprendizado Profundo como se fossem sinônimos. Apesar de relacionados, esses termos designam coisas diferentes. Como pode ser visto na Figura 4), Aprendizado de Máquina é uma (não a única) subárea de IA e Aprendizado Profundo é uma subárea de Aprendizado de Máquina.

Figura 4 – Diagrama mostrando a relação entre Inteligência Artificial, Aprendizado de Máquina e Aprendizado Profundo. Adaptado de Chollet et al. (2018).

Antes de um melhor detalhamento sobre Aprendizado de Máquina e Aprendizado Profundo, é importante definir-se o que significa “aprendizado”. Aprendizado, no contexto

de Aprendizado de Máquina (ou Profundo), pode ser entendido como um processo de busca automática por melhores representações de dados (CHOLLET et al., 2018) ou pela utilização de algoritmos para que os computadores adquiram descrições estruturais a partir de amostras de dados (PATTERSON; GIBSON, 2017).

O Aprendizado de Máquina refere-se ao campo do estudo em que treina-se, com uma base de dados e informações sobre ela, o computador para realização de alguma tarefa específica. Diferente da programação mais comum, em que as tarefas executadas por um computador seguem uma série de regras definidas explicitamente, no Aprendizado de Máquina o processo de treinamento consiste em fazer com que o computador aprenda as regras que descrevem os dados. Um exemplo de tarefa feita por Aprendizado de Máquina é a marcação automática de fotos, em que, com base em um grande número de fotos já marcadas por humanos, treina-se um algoritmo para realizar o mesmo processo em fotos ainda não marcadas. Uma das classes mais conhecidas de Aprendizado de Máquina são as Redes Neurais, que serão descritas com maiores detalhes adiante.

Por fim, o Aprendizado Profundo, como sendo um subconjunto do Aprendizado de Máquina, refere-se a estrutura das Redes Neurais que compõem o algoritmo de Aprendizado de Máquina. Numa definição comumente utilizada, o Aprendizado Profundo é uma Rede Neural com um grande número de camadas em sua arquitetura. Pode ser entendido também como um aprendizado que é feito com base em diversas camadas de representações sucessivas (CHOLLET et al., 2018). Nesse caso, o termo representações se refere às possíveis formas que os dados podem ser visualizados ou codificados. Uma imagem colorida, por exemplo, pode ser codificada tanto no formato RGB (vermelho-verde-azul, do Inglês, *Red-Green-Blue*) ou no formato HSV (matiz-saturação-valor, do Inglês, *Hue-Saturation-Value*).

2.3.1 Redes Neurais

As Rede Neurais (também conhecida como Redes Neurais Artificiais) surgiram a partir de uma tentativa de simular a forma com que o cérebro humano executa uma tarefa particular (HAYKIN, 2009). Como bem ressalta Chollet et al. (2018), embora alguns conceitos centrais de redes neurais tenham sido inspirados no entendimento que se tem do cérebro humano, as redes neurais não são um modelo do cérebro. Computacionalmente, entretanto, uma rede neural é definida como um grande processador paralelo e distribuído construído a partir de unidades de processamento menores que são capazes de armazenar um conhecimento experimental e disponibilizá-lo para uso (HAYKIN, 2009).

Da definição, tem-se que um dos principais componentes de uma rede neural são suas unidades de processamento ou neurônios. Um dos primeiros neurônios propostos mais conhecidos foi o Perceptron de Rosenblatt (ROSENBLATT, 1958), exibido na Figura 5. Ele possui 3 elementos básicos: I) um conjunto de conexões com pesos associados; II) um nó de junção que irá realizar uma combinação linear dos sinais de entrada, ponderados

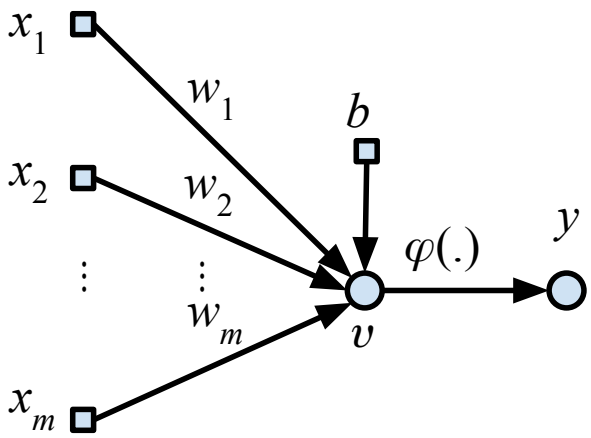


Figura 5 – Arquitetura de um perceptron. Adaptado de Haykin (2009) e Patterson e Gibson (2017).

pelos seus respectivos pesos e; III) uma função de que limita a amplitude do sinal de saída do neurônio. Apesar de simples, um único neurônio é capaz de realizar tarefas de classificação que possuam mais de duas classes desde que elas sejam linearmente separáveis. Ou seja, considerando que as classes estejam em um plano cartesiano tridimensional, é possível traçar-se um hiperplano que separe cada classe de dados.

Na Figura 5, as entradas do neurônio são representadas por $x_1, x_2, ..., x_m$, em que cada entrada possui um peso associado, representado por $w_1, w_2, ..., w_m$. Um bias, representado por b, pode ser externamente aplicado às entradas do neurônio. A combinação linear dos sinais de entrada, conhecida como campo local induzido, realizada por v é

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.4}$$

e $\varphi(.)$ é a função que calcula a saída que será retornada pelo perceptron, com base em v. Em redes neurais, $\varphi(.)$ é comumente conhecida como função de ativação. Atualmente existem diversas funções de ativação, sendo algumas delas:

- **Sigmoide**:
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- **Tangente Hiperbólica**:
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
- **ReLU**:
$$R(x) = max(0, x)$$
- **Softmax**:
$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

Cada função de ativação é usada para problemas diferentes. Por exemplo, a função sigmoide é comumente aplicada em problemas de classificação binários (compostos por duas classes) e geralmente retorna valores muito próximos de 0 ou 1. Similarmente, a função tangente hiperbólica pode ser usada para problemas de classificação, mas com a vantagem de poder tratar números negativos com mais facilidade. A função ReLU, embora simples, funciona em diversas situações e, inclusive, com desempenho melhor que funções de ativação similares à sigmoide (PATTERSON; GIBSON, 2017). Por fim, a função softmax geralmente é aplicada em classificações com múltiplas classes, pois retorna uma distribuição de probabilidade sobre as diferentes classes dos dados.

Um único perceptron possui capacidade limitada em relação aos tipos de problemas em que pode ser aplicado. Nesse sentido, foi construída uma arquitetura mais complexa baseada no perceptron, conhecida como perceptron multicamadas. Como o nome sugere, essa arquitetura consiste de uma série de perceptrons conectados sequencialmente, em camadas, em que cada perceptron propaga sua saída para os demais aos quais ele se conecta. A Figura 6, mostra um exemplo desse tipo de arquitetura. O número de camadas de um perceptron corresponde às suas camadas internas (ou seja, não são contabilizadas as camadas de entrada e saída), conhecidas como "camadas escondidas", por isso o perceptron da Figura 6 possui apenas 2 camadas.

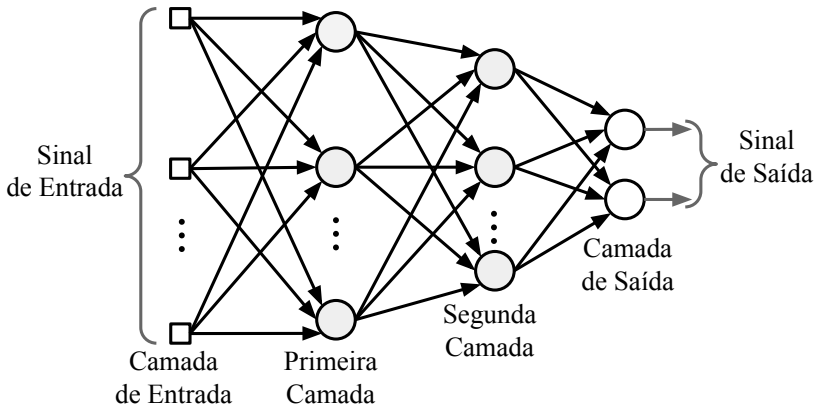


Figura 6 – Arquitetura de um perceptron multicamadas, com duas camadas escondidas. Adaptado de Haykin (2009).

Um perceptron multicamadas possui 3 características básicas (HAYKIN, 2009): cada neurônio da rede possui uma função de ativação não linear diferenciável (por exemplo, uma das anteriormente citadas); a rede contem 1 ou mais camadas que são escondidas (não são visíveis pelas camadas de entrada e saída) e; a rede possui um alto grau de conectividade (cada neurônio de uma camada conecta-se com todos os neurônios de próxima camada). Essas características, apesar de aumentarem o poder computacional disponível para o aprendizado, são também responsáveis por dificultar o entendimento que se tem do comportamento da rede (HAYKIN, 2009). Em primeiro lugar, é difícil

realizar uma análise teórica do perceptron multicamadas, devido a presença de uma rede altamente conectada e com não-linearidade distribuída. Em segundo, a presença de neurônios escondidos dificulta a visualização do processo de aprendizagem.

2.3.2 Arquiteturas de Redes Neurais

Com base na ideia geral dos perceptrons multicamadas, outras arquiteturas de redes neurais foram desenvolvidas. Patterson e Gibson (2017) apresenta uma categorização interessante para os diferentes tipos de redes neurais que podem existir. Nessa categorização as redes neurais são divididas em quatro grupos principais:

- **Redes Não-supervisionadas Pre-treinadas**: são redes treinadas de maneira não-supervisionada (os diferentes tipos de aprendizagem são abordados na Seção 2.3.2.1) e que, após uma etapa de pré-treino, podem ser utilizadas para aprender a realização de alguma outra tarefas. Exemplos desse tipo de rede são as *Deep Belief Networks* (DBNs), os *Autoenconders* e as GANs. Esses dois últimos são arquiteturas utilizadas em modelos generativos e as GANs, especificamente, serão discutidas com mais detalhes na Seção 2.3.4.

- **Redes Neurais Convolucionais** (CNN): é uma arquitetura bastante conhecida pela sua aplicação em visão computacional, principalmente no reconhecimento de imagens. A principal característica desse modelo é sua capacidade de criar representações internas dos dados, possibilitando que características específicas dos mesmos sejam aprendidas com eficiência.

- **Redes Neurais Recorrentes**(RNN): é uma arquitetura otimizada para a modelagem de sequências e são largamente usadas no processamento de linguagem natural. Exemplos desse tipo de arquitetura são a *Gated Recurrent Unit* (GRU) e a *Long Short-Term Memory* (LSTM), que será mais bem descrita na Seção 2.3.3.2.

- **Redes Neurais Recursivas**: Funcionam de forma similar às redes neurais recorrentes, usando alguma arquitetura específica, como os *autoencoders*, na sua composição. De maneira similar às RNNs, podem ser usadas em processamento de linguagem natural ou em dados em que há uma organização hierárquica interna.

Na prática, é comum que algumas arquiteturas de redes neurais sejam combinadas para resolver problemas específicos. As GANs, por exemplo, podem ser construídas com uma CNN (RADFORD; METZ; CHINTALA, 2015) ou com uma RNN (MOGREN, 2016; ESTEBAN; HYLAND; RÄTSCH, 2017; YOON; JARRETT; SCHAAR, 2019). É comum encontrar também combinações entre RNNs e CNNs (WANG et al., 2016b).

2.3.2.1 Aprendizagem em Redes Neurais

Como anteriormente mencionado, no Aprendizado de Máquina o objetivo é fazer com que uma máquina aprenda a realizar uma tarefa específica, com base em informações que lhe são fornecidas. O processo de aprendizado, assim como para humanos, pode ocorrer de diferentes formas e, no caso de redes neurais, pode variar de acordo com a tarefa a ser realizada. Assim, é possível identificar-se 4 tipos aprendizado (CHOLLET et al., 2018):

- **Aprendizado Supervisionado**: durante o treinamento a rede neural possui conhecimento das saídas que ela deve produzir, ou seja, a cada entrada a rede tem acesso à saída que deve ser produzida e os pesos de suas conexões são ajustados para reduzir a diferença entre o que foi gerado e a saída esperada. Nesse sentido, dados utilizados nesse tipo de aprendizado já foram anotados, ou classificados, por humanos.
- **Aprendizado não-supervisionado**: neste tipo de aprendizado os modelos não possuem acesso a nenhum tipo de saída esperada e consiste do processo de encontrar transformações importantes dos dados. Como exemplos clássicos de aprendizado não-supervisionado tem-se as reduções de dimensionalidade e clusterização.
- **Aprendizado Auto-supervisionado**: é uma subcategoria do aprendizado supervisionado. A principal diferença é que as saídas esperadas não foram previamente marcadas por humanos. Nesse caso, utiliza-se métodos específicos (conhecidos como algoritmos heurísticos) que vão gerar as saídas esperadas com base nas próprias entradas dos dados. São exemplos desse tipo de aprendizado a previsão de uma palavra em um texto, com base em palavras anteriormente vistas.
- **Aprendizagem por Reforço**: Nesse tipo de aprendizagem a Rede Neural não tem conhecimento das saídas que ela precisa produzir. A rede aprende por meio de um sistema que busca maximizar a obtenção de algum tipo de recompensa (a recompensa, nesse caso, depende da tarefa que a rede neural precisa executar). Como exemplo tem-se a rede neural criada por pesquisadores do *DeepMind* de Google, que foi treinada para executar jogos da plataforma Atari (MNIH et al., 2013).

Em redes neurais, o mesmo problema pode ser resolvido utilizando-se tipos de aprendizagem diferente. Entretanto, entre os tipos de aprendizagem existentes, o aprendizado supervisionado é o mais comum entre eles. Dessa forma, apresentaremos a seguir algumas características importantes para o treinamento de redes neurais utilizando aprendizado supervisionado.

2.3.2.2 Função de Perda

As funções de perda, também conhecidas como funções objetivo, são uma forma de se mensurar, durante o treinamento, o quão perto a rede neural está do seu estado ideal

(quando a rede passa a produzir os resultados esperados) (PATTERSON; GIBSON, 2017; CHOLLET et al., 2018). Utilizando as saídas esperadas e geradas pela rede, calcula-se o "erro" das previsões da rede. Esse "erro" é indica o quão longe das saídas esperadas estão as saídas que foram produzidas pela rede neural. Nesse sentido, o objetivo geral ao se treinar a rede neural é minimizar o valor do erro produzido pela função de perda.

Uma função de perda pode ser utilizada para diferentes problemas, entretanto, algumas funções possuem um desempenho melhor dependendo do problema que se pretende tratar. Existem diversas funções e, dependendo da especificidade do problema, é possível criar uma função de perda própria e utilizá-la durante o treinamento. Para tarefas que envolvem regressão, por exemplo, tem-se a função de erro quadrático médio (*mean squared error*, em inglês). Para problemas de classificação com duas classes é comum a utilização de uma função conhecida como entropia cruzada. As equações que representam cada uma dessas funções estão descritas a seguir:

- **Erro quadrático médio**:

$$L(w, b) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{2.5}$$

- **Entropia Cruzada**:

$$L(w, b) = -\sum_{i=1}^{N} y_i \times \log \hat{y}_i + (1 - y_i) \times \log(1 - \hat{y}_i) \tag{2.6}$$

Nas Equações 2.5 e 2.6 L é uma função de perda definida por pesos w e bias b. N é o número de amostras de dados e i é a i-ésima iteração (época) em que o cálculo é feito. A saída esperada é representada por y e $\hat{y}$ é a saída que foi prevista pela rede neural.

2.3.2.3 Métodos de Otimização

Os métodos de otimização são utilizados para minimizar os erros calculados pelas funções de perda (PATTERSON; GIBSON, 2017; CHOLLET et al., 2018). Um dos métodos mais conhecidos que realiza essa função é o Descida do Gradiente (*Stochastic Gradient Descent*, em Inglês). Em redes neurais, o objetivo geral do algoritmo é calcular, em uma dada iteração, as atualizações que devem ser feitas nos pesos para que a função de perda retorne os menores valores possíveis (CHOLLET et al., 2018). Assim, à cada nova iteração a função de perda passa a retornar um valor menor que a iteração anterior. O termo "gradiente", nesse caso, refere-se ao fato de que as funções cujas derivadas serão calculadas possuírem entradas multidimensionais (CHOLLET et al., 2018).

Assim, considerando que em uma iteração i uma função de perda $f(w)$ tenha um conjunto de parâmetros w, seu gradiente será representado por $\Delta f(w)$. Durante o passo

de otimização dos parâmetros, o algoritmo irá calcular $w = w - \eta\Delta f(w)$, onde η é a taxa de aprendizagem. Em outras palavras, os parâmetros são movidos na direção contrária ao gradiente calculado, reduzindo o valor que será gerado pela função de perda. A velocidade com o que os parâmetros são movidos depende do valor de η.

Uma das principais desvantagens desse algoritmo é que, para problemas não-lineares, as funções podem apresentar mais de um mínimo local, ou seja, existem diversas combinações de parâmetros que fazem com que a função de perda retorne um valor mínimo. Dessa maneira, o algoritmo pode ficar preso em um mínimo local, sem nunca conseguir atingir o menor dos mínimos locais, também conhecido como mínimo global (PATTERSON; GIBSON, 2017; CHOLLET et al., 2018) que é, de fato, o objetivo do algoritmo.

Existem diversas variações do algoritmo de Descida do Gradiente como o *Momentum* (RUMELHART; HINTON; WILLIAMS, 1986), o *Adagrad* (DUCHI; HAZAN; SINGER, 2011) o *RMSProp* (HINTON; TIELEMAN, 2012), Adam(KINGMA; BA, 2014). A *Momentum*, por exemplo, surgiu como uma proposta para solucionar a desvantagem anteriormente mencionada, em que o algoritmo fica preso em um mínimo local.

2.3.2.4 Algoritmo de Retro-Propagação

Um dos algoritmos mais utilizados para o treinamento de redes neurais atualmente, é conhecido como retro-propagação (do Inglês, *back-propagation*). O treinamento, em si, possui duas fases (HAYKIN, 2009):

- **Propagação**: os pesos das conexões são fixados e os sinais de entradas são propagados pela rede, camada à camada, até chegar à saída da rede.
- **Retrocesso**: um sinal de erro (e) é produzido a partir da comparação entre a saída (y) da rede com a resposta desejada (d). O erro resultante é propagado, de trás para frente, por cada camada da rede. Os pesos das conexões entre os neurônios são ajustados com base nos erros propagados. Assim, o sinal de erro produzido por um neurônio de saída j na iteração n é definido pela Equação 2.7:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.7}$$

  Essa equação, nesse caso, é uma representação geral das funções de perda anteriormente mencionada.

A seguir, será apresentada uma visão geral das equações que são usadas no algoritmo de retro-propagação. Uma descrição com mais detalhes do algoritmo pode ser vista em (HAYKIN, 2009) e um pseudocódigo do mesmo pode ser encontrado em (PATTERSON; GIBSON, 2017). A ideia geral do algoritmo é calcular gradientes através de aplicações recursivas de regras da cadeia, com base no sinal de saída dos neurônios que fazem parte

da rede neural. Para facilitar o entendimento, as equações são demostradas considerando um caso mais simples, em que a rede neural é um perceptron multicamadas.

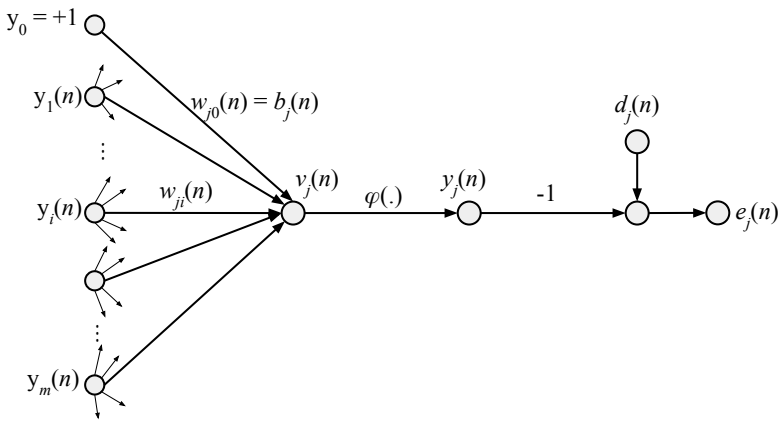


Figura 7 – Gráfico de fluxo de sinais de um neurônio de saída j. Adaptado de Haykin (2009).

Assim, a Figura 7 mostra os fluxos de sinais que passam por um neurônio de saída j, na iteração n. Na etapa de propagação, os sinais que chegam ao neurônio precisam ser propagados pela rede. Considerando uma época de treinamento n, com vetores de entrada $x(n)$ e saídas desejadas $d(n)$, o campo local induzido $v_v^e(n)$ para o neurônio j em uma camada l será

$$v_j(n) = \sum_{i=0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \tag{2.8}$$

em que $y_i^{(l-1)}$ é a função de saída do neurônio i na camada anterior à l e $w_{ji}^{(l)}(n)$ é o peso das conexões entre o neurônio i e j na iteração n. Assumindo uma função de ativação φ qualquer, o sinal de saída do neurônio j na camada l será:

$$y_j(n)^{(l)} = \varphi_j(v_j(n)) \tag{2.9}$$

Assim, caso o neurônio esteja na primeira camada ($l = 1$), o sinal de saída será dado por

$$y_j^{(1)}(n) = x_j(n) \tag{2.10}$$

onde x_j é o valor da j-ésima entrada. Por outro lado, se o neurônio estiver na camada l, o sinal de saída será calculado por

$$y_j^{(L)}(n) = o_j(n) \tag{2.11}$$

onde o_j é o j-ésimo elemento do vetor de saída do neurônio da camada $l-1$ que se conecta com j. Por fim, calcula-se o sinal de erro com

$$e_j(n) = d_j(n) - o_j(n) \tag{2.12}$$

onde d_j é o j-ésimo elemento no vetor de saídas esperadas $d(n)$.

Na etapa de retrocesso, calcula-se os gradientes locais para cada neurônio da rede, considerando dois casos específicos. No primeiro, o neurônio j está na saída da camada L e o gradiente local é

$$\delta_j^{(l)}(n) = e_j^{(L)}(n)\varphi'_j(v_j^{(L)}(n)) \tag{2.13}$$

No segundo caso, se o neurônio j estiver na camada escondida l, o gradiente local será

$$\delta_j^{(l)}(n) = \varphi'_j(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) \tag{2.14}$$

É importante ressaltar que a função principal do algoritmo de retro-propagação é calcular os gradientes para cada neurônio da rede. O processo de treinamento ocorre em conjunto com algoritmos de otimização, como o de descida do gradiente, que irão reduzir o erro calculado pela função de perda, da Equação 2.7.

2.3.3 Redes Neurais Recorrentes

Boa parte das arquiteturas de redes neurais tratam de dados multidimensionais em que os atributos, ou variáveis, são altamente independentes entre si (AGGARWAL et al., 2018). Entretanto, em outros tipos de dados, como séries temporais, textos e dados biológicos, os atributos apresentam uma dependência entre si e, em muitos casos, são inerentemente ordenados, em que valores futuros dependem de valores passados (PATTERSON; GIBSON, 2017).

Esses tipos de dados são modelados com uma arquitetura conhecida como Redes Neurais Recorrentes (RNN). A principal diferença desse tipo de arquitetura em relação às outras é a capacidade de modelar a dependência do tempo que os dados possuem. Outras arquiteturas de aprendizagem de máquina foram usadas sucesso na modelagem desse tipo de dado, como modelos de classificação, de regressão logística e redes neurais convencionais, em que assumia-se que as variáveis eram independentes do tempo. Esse tipo de abordagem, porém, impede que uma relação de dependência temporal que apresente um intervalo muito grande seja modelada com eficiência.

Outra diferença importante entre as RNNs e redes neurais convencionais é que as RNNs são capazes de modelar entradas e saídas de tamanhos não-fixos. Em alguns problemas de sequência-para-sequência (tradução de frases, por exemplo) o tamanho da entrada da rede pode variar, bem como o tamanho da saída que será produzida pela rede. Nesse sentido, os tipos de entrada e saída que uma RNN pode tratar podem ser divididos em três categorias (PATTERSON; GIBSON, 2017; CHOLLET et al., 2018):

- Um-para-muitos: a rede recebe um valor como entrada e retorna uma sequência (vetor) como saída.

- Muitos-para-um: a rede recebe uma sequência como entrada e retorna um valor como saída.

- Muitos-para-muitos: tanto a entrada como a saída são sequências.

Em diferentes domínios é possível encontrar aplicações práticas de RNNs, que se encaixa pelo menos em uma das categorias anteriormente mencionadas. Para o primeiro caso, as RNN tem sido usadas para, entre outras situações, a criação de legendas em imagens (YOU et al., 2016), geração de texto em nível de caracteres (SUTSKEVER; MARTENS; HINTON, 2011), síntese de fala (GRAVES; JAITLY, 2014) e geração de músicas (NAYEBI; VITELLI, 2015). No segundo caso, é possível utilizar as RNNs para prever séries temporais (CONNOR; MARTIN; ATLAS, 1994; GILES; LAWRENCE; TSOI, 2001), transcrever notas de piano (BÖCK; SCHEDL, 2012) e analisar emoções em vídeos (KAHOU et al., 2015). Por fim, no último caso, as RNNs são utilizadas em problemas como tradução de linguagem natural (VENUGOPALAN et al., 2014) e modelagem de diálogos (HENDERSON; THOMSON; YOUNG, 2014; WEN et al., 2015; SERBAN et al., 2017)

2.3.3.1 Arquitetura de uma Rede Neural Recorrente

Conceitualmente, uma RNN é uma variação de uma rede neural convencional, com a adição da ideia de conexões recorrentes (por isso o nome rede neural recorrente). Essas conexões recorrentes consistem em sinais de ativação e erro de uma camada sendo usadas como entrada da mesma camada (PATTERSON; GIBSON, 2017).

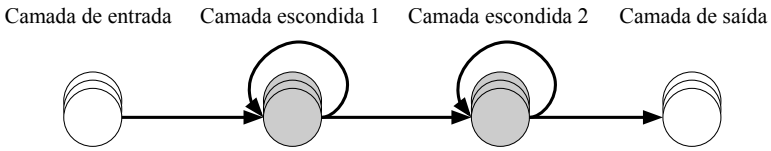


Figura 8 – Arquitetura geral de uma Rede Neural Recorrente. Adaptado de Patterson e Gibson (2017).

Ilustrando essa situação, a Figura 8 exibe a arquitetura de uma RNN com duas camadas escondidas. Para facilitar a visualização, cada camada da rede é representada por círculos sobrepostos, em que cada círculo representa um neurônio. Como mostra a figura, nas camadas escondidas há uma conexão que conecta cada camada à si mesma. Dessa forma, à cada iteração i a camada escondida recebe entradas de ativação tanto dos neurônios da própria camada quanto de neurônios em camadas anteriores. Uma representação de como as informações em uma RNN são propagadas à cada iteração pode ser vista na Figura 9.

Considerando a sua arquitetura, em que as conexões recorrentes são fundamentais, as RNNs sofrem de dois tipos de desafios principais durante seu treinamento, conhecidos como

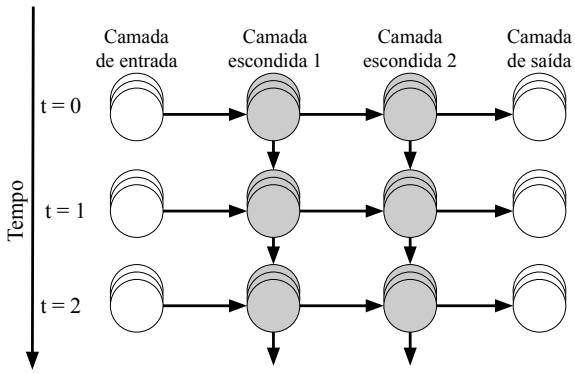


Figura 9 – Fluxo de informações em uma RNN ao longo do tempo. Adaptado de Patterson e Gibson (2017).

problemas de *Vanishing* e *Exploding Gradient* (HOCHREITER et al., 2001; PASCANU; MIKOLOV; BENGIO, 2012; PASCANU; MIKOLOV; BENGIO, 2013). Esses problemas são comuns em redes neurais, principalmente por que as multiplicações internas envolvendo os pesos das conexões na rede são inerentemente instáveis (AGGARWAL et al., 2018). Nesse sentido, após sucessivas multiplicações, durante a retro-propagação, tanto o gradiente pode desaparecer (*Vanishing Gradient*) ou tornar-se extremamente grande (*Exploding Gradient*) impossibilitando que a rede neural consiga atualizar corretamente seus pesos. Isso explica, por exemplo, porque em situações em que são modeladas sequências cujos valores possuem dependências temporais próximas entre si esses problemas ocorrem com menos frequência (AGGARWAL et al., 2018).

2.3.3.2 Long Short-Term Memory

Uma das arquiteturas de redes neurais recorrentes mais conhecidas e utilizadas atualmente é a LSTM (do Inglês, *Long Short-Term Memory*), introduzida em 1997 por Hochreiter e Schmidhuber (1997). A principal diferença entre a arquitetura de uma LSTM e uma RNN convencional são as células de memória e os portões de entrada, saída e esquecimento que a LSTM possui. Enquanto as células de memória armazenam informações dos dados, a estrutura de portões permite que informações sejam mantidas durante várias iterações (PATTERSON; GIBSON, 2017). Devido à essas características adicionais, bem como da forma como as operações sobre as conexões recorrentes são feitas nessa nova arquitetura, a LSTM não sofre dos problemas de *Vanishing* e *Exploding Gradient* anteriormente mencionados.

Um dos componentes mais importantes da LSTM, é conhecido como bloco LSTM, que pode ser visto na Figura 10. Alguns componentes mostrados na figura são contribuições de Gers, Schmidhuber e Cummins (2000) e Gers e Schmidhuber (2000). Os principais componentes de um bloco LSTM são:

- **Célula de memória**: é o componente responsável por armazenar informações

importantes sobre os dados.

- **Portões**: um bloco LSTM possui três portões. O portão de entrada é responsável por impedir que a célula de memória armazene informações irrelevantes. O portão de saída controla a propagação do conteúdo armazenado pela saída do bloco LSTM. O portão de esquecimento permite que a unidade descarte conteúdos previamente aprendidos.

Como pode ser visto na Figura 10, os portões do bloco LSTM implementam a função de ativação sigmoid. Entretanto, as funções de ativação de entrada e saída podem implementar funções diferentes, embora geralmente seja utilizada a função *tanh* (tangente hiperbólica), que foi anteriormente descrita.

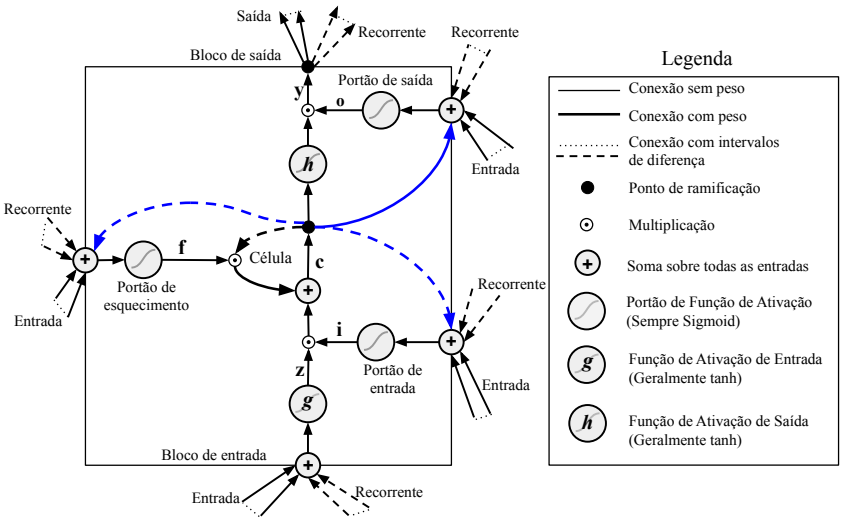


Figura 10 – Diagrama de um bloco LSTM adaptado de Patterson e Gibson (2017).

Como a arquitetura original da LSTM ainda não era capaz de aprender dependências temporais que exigiam uma maior precisão, Gers, Schmidhuber e Cummins (2000) propuseram a adição de um componente que aprimorava o controle sobre os portões de entrada e esquecimento. Esses componentes são conhecidos como conexões *peephole*, mostrados em azul na Figura 10, que se conectam às entradas do bloco LSTM e à célula de memória.

As equações a seguir, na notação de Greff et al. (2016) apresentam a etapa de propagação (*forward pass*) em uma camada LSTM. Na notação, os pesos de entrada são representados por $W_z, W_i, W_f, W_o \in \mathbb{R}^{N \times M}$; os pesos recorrentes são representados por $R_z, R_i, R_f, R_o \in \mathbb{R}^{N \times N}$; os pesos das conexões *peephole* por $P_z, P_i, P_f, P_o \in \mathbb{R}^N$; e os pesos dos bias por $b_z, b_i, b_f, b_o \in \mathbb{R}^N$.

$$
\begin{aligned}
z^t &= g(W_z x^t + R_z y^{t-1} + b_z) && \text{(bloco de entrada)} \\
i^t &= \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i) && \text{(portão de entrada)} \\
f^t &= \sigma(W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f) && \text{(portão de esquecimento)} \\
c^t &= i^t \odot z^t + f^t \odot c^{t-1} && \text{(célula de estado)} \\
o^t &= \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o) && \text{(portão de saída)} \\
y^t &= o^t \odot h(c^t) && \text{(bloco de saída)}
\end{aligned}
$$

Atualmente, devido à sua superioridade em relação à arquiteturas de RNN convencionais, boa parte das aplicações utilizam modelos construídos com LSTM (AGGARWAL et al., 2018). Alguns exemplos de aplicação de LSTM são não criação automática de legendas para imagens (WANG et al., 2016a), sistemas de tradução (WU et al., 2016) (o sistema utilizado pelo Google Tradutor[3]), análise de sentimento em textos (WANG et al., 2016c), reconhecimento de fala (GRAVES; JAITLY, 2014) e classificação de séries temporais multivariadas (KARIM et al., 2019).

2.3.3.3 Treinamento de Redes Neurais Recorrentes

Em relação ao tipo de treinamento, as RNN utilizam o aprendizado supervisionado para atualizar seus pesos na rede. A entrada de uma RNN é um vetor, mesmo nos casos um-para-muitos. Por exemplo, em problemas de geração de legendas para imagens, a entrada de uma RNN será uma imagem (que representa um único valor), mas sua representação, para o treinamento é feita através de um vetor multidimensional.

Nos neurônios de entrada os vetores tornam-se sequências de ativações e cada neurônio nas camadas não correspondentes à entrada calculam sua própria ativação para a iteração atual. O cálculo feito pelo neurônio corresponde à uma função não-linear das somas ponderadas das ativações de todos os neurônios que se conectam à ele (PATTERSON; GIBSON, 2017).

Assim como nas redes neurais convencionais, nas RNN as atualizações dos pesos dependem da propagação dos gradientes pela rede. Dessa forma, as RNNs utilizam uma variação do algoritmo de retro-propagação, conhecido como retro-propagação através do tempo (do Inglês, *backpropagation through time* ou BPTT). Em sequências muito longas e com muito intervalos de tempo, o ideal é que seja utilizada BPTT truncado, que reduz a complexidade de cada atualização dos parâmetros em uma RNN (PATTERSON; GIBSON, 2017).

Conceitualmente, a ideia geral do BPTT é a mesma do algoritmo de retro-propagação, onde são calculados, recursivamente, os gradientes (derivadas) das funções de

[3] https://translate.google.com/

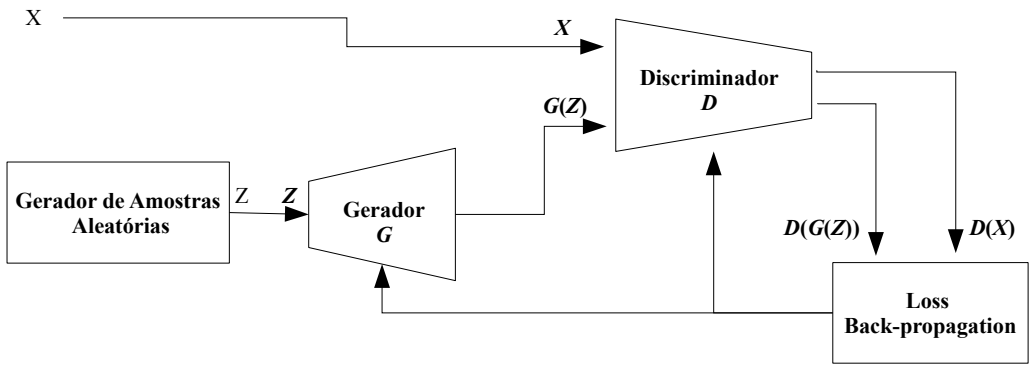


Figura 11 – Arquitetura de uma *Generative Adversarial Network* mostrando o seus principais componentes. Adaptado de Goodfellow et al. (2014a).

perda em cada camada da rede. A principal diferença, neste caso, é que no BPTT, alguns dos gradientes e sinais de erros calculados em uma determinada iteração são transmitidos para iterações passadas (PATTERSON; GIBSON, 2017). A demonstração das equações do BPTT dentro de um bloco LSTM podem ser encontradas em detalhes em Greff et al. (2016).

2.3.4 Generative Adversarial Networks

A Redes Adversárias Generativas (GANs, em inglês) é um *framework* que tem o objetivo de otimizar o treinamento de modelos generativos não-supervisionados, por meio de um "jogo" de min-max (GOODFELLOW et al., 2014b). No jogo, são treinadas simultaneamente e por tempo indeterminado, duas redes neurais que competem entre si (por isso o nome *adversarial*): um Gerador ($G(z;\theta_g)$), um perceptron multicamadas que irá gerar dados falsos com base em entradas aleatórias $p_z(z)$ e um Discriminador ($D(x;\theta_d)$), outro perceptron multicamadas que classificará a qualidade dos dados gerados, considerando uma base de dados reais x. Nas redes, z, θ_g e θ_d significam, respectivamente, o espaço latente dos dados (entradas aleatórias, como distribuições normais), os parâmetros para o perceptron que define G e os parâmetros para o perceptron que define D. O objetivo de G é gerar amostras de dados cuja distribuição se aproxime tanto da distribuição real $p_{data}(x)$, que D não consiga distinguir dos reais. Assim, ao mesmo tempo que D é treinado para distinguir entre os dados oriundos de G e os reais, G terá seus pesos atualizados considerando o *feedback* fornecido por D. A Figura 11 apresenta um diagrama da estrutura básica de uma GAN.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{2.15}$$

Em teoria, como descrito na Equação 2.15 presente em (GOODFELLOW et al., 2014b), ao utilizar a técnica de min-max e com tempo e recursos computacionais suficientes,

espera-se que D e G entrem em equilíbrio. Entretanto, o objetivo geral da GAN é encontrar um equilíbrio de Nash de um jogo não-convexo e com parâmetros contínuos e de alta-dimensionalidade (SALIMANS et al., 2016). Porém, as técnicas usadas para o treinamento de GANs normalmente se baseiam em gradientes descendentes (SALIMANS et al., 2016) que não são apropriadas para encontrar o equilíbrio de Nash de um jogo. Dessa forma, as primeiras GANs poderiam apresentar algumas instabilidades, por exemplo, dada a complexidade dos dados, D e G poderiam nunca chegar um ponto de convergência. Nesse sentido, o trabalho de (SALIMANS et al., 2016) apresentam uma série de técnicas que podem ser aplicadas para a melhorar o treinamento de GANs. Um dos primeiros resultados

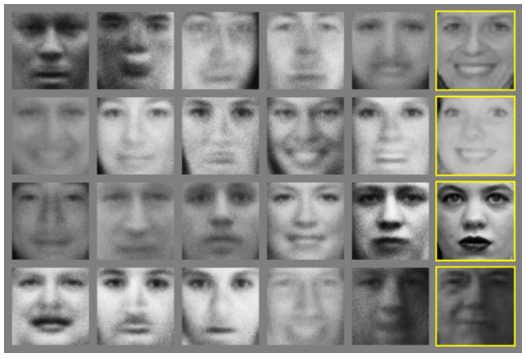


Figura 12 – Rostos gerados por uma GAN treinada com a base de dados TFD (SUSSKIND; ANDERSON; HINTON, 2010). Figura de Goodfellow et al. (2014b).

do *framework* proposto por Goodfellow et al. (2014b) pode ser visto na Figura 12. Os rostos sintéticos foram gerados a partir da base original TFD (SUSSKIND; ANDERSON; HINTON, 2010), uma base de dados com faces de pessoas em baixa resolução e preto e branco. Trabalhos posteriores à (GOODFELLOW et al., 2014b), utilizando algumas das técnicas propostas por Salimans et al. (2016) em conjunto com novas estruturas de GANs, mostraram a capacidade das GANs de gerarem imagens extremamente realistas, por exemplo, imagens sintéticas com base em diferentes categorias de imagens reais (Figura 13) (BROCK; DONAHUE; SIMONYAN, 2018) ou com base em imagens de celebridades (Figura 14). No primeiro caso, os autores utilizaram uma DCGAN (RADFORD; METZ; CHINTALA, 2015) em conjunto com uma GAN condicional (uma GAN em que é possível definir a categoria da imagem que se quer gerar). Já no segundo caso, os autores utilizaram uma estratégia de treinamento que consistia em, aos poucos, aumentar a resolução das imagens reais utilizadas durante o treinamento.

Apesar dos seu bom desempenho, o *framework* proposto por Goodfellow et al. (2014b) apresenta algumas desvantagens durante o treinamento como o fato de, geralmente, não ser possível inferir a qualidade dos dados gerados sem analisá-los visualmente durante cada etapa de treinamento. G pode gerar dados que são estatisticamente indistinguíveis dos reais, mas que não fazem sentido para um observador humano. Assim, treinar uma GAN por mais tempo não implica em melhores resultados.

Figura 13 – Diferentes categorias de imagens geradas pela BigGAN. Figura obtida de Brock, Donahue e Simonyan (2018).

Figura 14 – Rostos gerados com base em imagens de celebridades. Figura obtida de Karras et al. (2017).

2.4 Modelagem de Séries Temporais

Como visto anteriormente, uma série temporal pode ser resultado de diversos eventos que ocorrem naturalmente ou gerados por ações humanas. Nesse sentido, a modelagem de uma série temporal consiste em identificar as propriedades de tais eventos que geraram a série em estudo.

Uma forma de identificar essas propriedades é utilizar um sistema de modelagem (e previsão) de séries temporais, que pode ser compreendido em duas etapas principais: uma etapa de construção do modelo e uma de previsão, em que gera-se observações previstas pelo modelo construído. A Figura 15 apresenta uma visão geral de um modelo para modelagem e previsão de séries temporais e, a seguir, tem-se a descrição de cada etapa:

- **Construção do Modelo**: Nesta etapa defini-se modelo a partir de uma base de dados e de conhecimentos teóricos disponíveis, que podem sugerir um modelo específico para os dados. O modelo provisório geralmente possui parâmetros que podem ser estimados utilizando-se algumas técnicas, como os quadrados mínimos. Por fim, avalia-se a adequação do modelo e, caso a adequação não seja satisfatória, o processo se repete até que se encontre um modelo adequado.

- **Predição**: É comum, na modelagem de séries temporais, a utilização de um modelo ajustado aos dados para a realização de previsões. Cada modelo é construído para um problema específico e seus parâmetros precisam manter-se constantes durante a etapa de previsão (modelo estável). Isso pode ser avaliado ao comparar-se as previsões com novas observações dos dados.

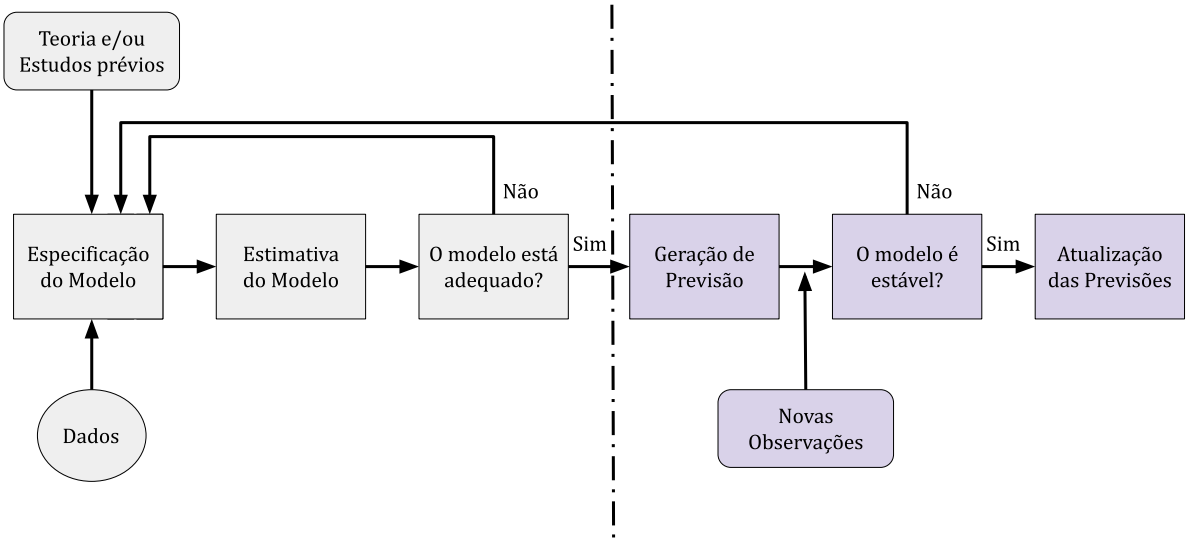


Figura 15 – Visão conceitual de um sistema para modelagem e previsão de séries temporais. Adaptado de Abraham e Ledolter (2009).

Em modelos mais simples, como os estatísticos, partes de cada etapa geralmente são realizadas separadamente. Nesse caso, primeiro identifica-se as propriedades da série temporal, como sazonalidades e tendências e qual modelo estatístico melhor descreve essa série, com base nas propriedades dos dados, como modelos puramente auto-regressivos, de médias móveis ou ARIMA, por exemplo.

Modelos mais complexos, como baseados em redes neurais e aprendizado profundo, podem implementar mais de uma das partes, por exemplo, como visto na Figura 15, a etapa de definição do modelo possui três componentes (especificação do modelo, estimativa do modelo e verificação da adequação do modelo). Por exemplo, a maioria das implementações da LSTM atualmente já possui, internamente, todos os componentes da etapa de construção do modelo. A definição dos parâmetros iniciais do modelo é feita antes do treinamento e, durante o treinamento, o próprio modelo utiliza técnicas otimizadas para ajustar seus parâmetros. É comum que seja exibido, à cada iteração do treinamento, o desempenho do modelo, que representa o componente de avaliação de adequação do modelo aos dados.

2.5 Considerações Sobre o Capítulo

Neste capítulo apresentamos os conceitos fundamentais para o melhor entendimento da presente dissertação. Primeiramente definimos o que é uma série temporal, apresentando algumas das suas principais propriedades estatísticas e modelos normalmente são usados para descrevê-la. Em seguida apresentamos uma visão de geral dos principais conceitos sobre redes neurais e aprendizado profundo e de como as séries temporais são modeladas.

3 Trabalhos Relacionados

Neste capítulo apresentaremos uma revisão dos trabalhos que abordam o problema de geração de dados de mobilidade urbana utilizando métodos baseados em aprendizado profundo, especificamente, as GANs. Na Seção 3.1, sumarizamos o mapeamento sistemático feito para identificação dos trabalhos relacionados à esta dissertação.

3.1 Mapeamento Sistemático

O mapeamento sistemático realizado neste trabalho utilizou a metodologia apresentada em Falbo (2018). Objetivo do mapeamento foi identificar os trabalhos que abordam a geração de dados de mobilidade urbana utilizando GANs. A partir do objetivo, definimos a palavra-chave à ser usada nos mecanismos de busca. Para aumentar o número de resultados retornados, as palavras usadas estão em Inglês. Assim, utilizamos uma combinação dos termos "*generative adversarial network*" e "*urban mobility*" ou "*human mobility*", resultando na palavra-chave final (("*Generative Adversarial Network*") AND ("*urban mobility*" OR "*human mobility*")). Como mecanismos de busca utilizamos o IEEE Xplore[1] e o Scopus[2]. Utilizamos esses dois mecanismos porque, durante os primeiros testes iniciais com as palavras-chave, foram os únicos que retornavam artigos minimamente relevantes.

Tabela 1 – Resultados do mapeamento sistemático realizado

Mecanismo de busca	**Resultados**
Scopus	52
IEEE Xplore	4
Total	55
Seleção final	15

Dessa forma, a Tabela 1 mostra os resultados das pesquisas feitas nos mecanismos de busca. Para a pesquisa dos artigos, filtramos os resultados pela data de publicação, entre 2014 (ano de criação das GANs) e 2020. Ademais, o mapeamento foi conduzido durante o ano de 2020. Após a aplicação da palavra-chave e obtenção dos artigos, identificamos, por meio da análise dos títulos, resumos e palavras-chaves, os artigos relevantes que pudessem responder às questões de pesquisa, reduzindo o número de artigos candidatos para 15. Nesta etapa, todos os artigos encontrados passaram pelos critérios de exclusão definidos e, devido o número reduzido de publicações encontradas, utilizamos apenas um critério de inclusão:

[1] http://ieeexplore.ieee.org
[2] http://www.scopus.com

CI: Os trabalhos abordam a utilização de GANs na modelagem de mobilidade urbana

O número reduzido de artigos retornados nas buscas pode ser explicado por dois fatores principais. Primeiro, as GANs são relativamente jovens (foram criadas em 2014). Logo, qualquer artigo que trata-se de GANs e mobilidade urbana, seria produzido após esse ano. Segundo, as GANs foram desenvolvidas com o foco principal na área de visão computacional, principalmente para a geração de imagens. Trabalhos abordando a geração de dados de mobilidade urbana começaram a aparecer alguns anos depois.

Os trabalhos resultantes do mapeamento sistemático podem ser classificados em 4 categorias, com relação ao tipo de problema que resolvem: geração de trajetórias, geração de dados de serviços de transporte, geração de contatos e geração de tráfego urbano. Nas seções a seguir discutiremos os trabalhos encontrados em cada uma das categorias.

3.2 Geração de trajetórias

Em mobilidade urbana, as trajetórias feitas por indivíduos são obtida com base nos lugares em que tal indivíduo esteve. A geração desse tipo de mobilidade urbana depende das localizações GPS dos usuários para que seja possível se construir a trajetória feita por cada um deles. Nesse sentido, alguns desafios para modelagem desses dados se destacam:

- Geograficamente, as trajetória de cada indivíduo são bem distintas;
- A mobilidade das pessoas pode apresentar alguns comportamentos comuns, por exemplo, a rotina de sair de casa ir pro trabalho e retornar para casa;
- A mobilidade das pessoas sofre influência de características da região analisada. Consequentemente, regiões distintas apresentarão comportamentos distintos

Considerando os desafios descritos, alguns trabalhos que buscam a geração de trajetórias utilizam metodologias similares (OUYANG et al., 2018; SONG; BAEK; SUNG, 2019; ZHANG et al., 2020). Basicamente, os autores modelam a região de origem dos dados como uma imagem, representadas por uma matriz $N1 \times N2$ e utilizam GANs construídas com CNNs como modelos básicos para a geração dos dados. Em Ouyang et al. (2018) os autores utilizam uma base de dados com as trajetórias feitas por 170 pessoas na cidade de Lausanne (KIUKKONEN et al., 2010), Song, Baek e Sung (2019) usam localizações de pessoas em 5 cidades da Coreia do Sul (Seoul, Jeju, Gwangju, KyeongJu e Pohang) e Zhang et al. (2020) usam dados de trajetórias de táxis da cidade de Beijing, na China.

Em (GUPTA et al., 2018) os autores descrevem um modelo de geração de trajetórias socialmente aceitáveis em ambientes com muitas pessoas, por exemplo, uma calçada ou rua muito movimentada. Assim, no referido trabalho, a expressão “socialmente aceitável”

indica uma trajetória de um indivíduo que, simultaneamente, seja fisicamente possível e respeite o espaço físico de indivíduos próximos. Entretanto, em relação às GANs, neste trabalho os autores utilizam modelos que são otimizados para geração de imagens, como ocorreu nos trabalhos anteriores.

Modelando os dados de maneira distinta do que foi visto até aqui, Feng et al. (2020) propõe uma arquitetura de GAN específica para tratar dos dados de mobilidade urbana. Considerando que os dados de mobilidade ocorrem de maneira sequencial, como uma série temporal, os autores definem que o Gerador da GAN terá duas redes internas: uma, chamada de *SeqNet* que vai modelar as transições temporais de mobilidade e outra (*RegNet*) que irá capturar os efeitos que as características das cidades têm nos dados, como atributos específicos das localizações visitadas e relações entre os locais visitados.

Com foco específico em prover privacidade para as trajetórias geradas, Rao et al. (2020) propõe a LSTM-TrajGAN, um modelo para geração de trajetórias feitas por pessoas com informações como localizações GPS e de pontos de interesse. Considerando os requisitos do modelo proposto, que dever gerar dados de tipos diferentes (numéricos e categóricos), o autores propuseram também uma nova função de perda, denominada de *TrajLoss*. Tanto para o Gerador quanto para o Discriminador da GANs, os autores utilizam uma combinação de Perceptron Multicamadas e unidades LSTM. Para os experimentos, são utilizados uma base de dados de usuários do Foursquare (PETRY et al., 2020).

3.3 Geração de dados de serviços de transporte

Alguns serviços, como táxi ou de compartilhamentos de corridas (*Riding Sharing*), geram informações importantes sobre mobilidade urbana, com base, principalmente, nas localizações de partida e chegada de um usuário. Dessa forma, assim como para as trajetórias, a geração desse tipo de dado depende de informações que indiquem, com um bom nível de precisão, a localização dos usuários.

Especificamente para geração de dados de compartilhamentos de corrida, Jauhri et al. (2020) desenvolveram um modelo para a geração de bases de dados de pedidos de carona utilizando dados de serviços de carona de quatro cidades dos Estados Unidos. Os autores mapearam os dados contendo o ponto de partida, ponto de chegada e tempo em uma sequências de imagens, de maneira que, a cada instante de tempo, tem-se um *snapshot* da configuração dos pedidos de carona naquele instante.

A simulação de dados gerados por serviços de táxi é abordada em Yin e Yang (2018), Yu et al. (2019), Yu et al. (2020a). No primeiro trabalho, Yin e Yang (2018) abordam o problema de preservação de privacidade considerando a distribuição da densidade de dados de mobilidade. Neste trabalho os autores propõe um método utilizando a arquitetura original das GANs, apenas utilizando uma função de perda diferente, conhecida como

Wasserstain distance (VALLENDER, 1974). Finalmente, Yu et al. (2019) e Yu et al. (2020a) abordam problemas parecidos, simulando dados de pedidos de táxi, considerando os locais de partida e chegada dos usuário. Da mesma forma, os modelos de GANs propostos nos dois trabalhos possuem arquiteturas similares, compostas de GANs condicionais (CGAN) e LSTMs.

3.4 Geração de contatos

Os dados, nesse tipo de problema, podem ser visualizados como uma rede, com nós e conexões entre eles. Do ponto de vista de mobilidade, um "contato" entre indivíduos pode ocorrer, por exemplo, quando eles estão próximos. Assim, a geração de dados para esse problema visa simular a ocorrência de contatos entre nós de uma rede. No contexto de mobilidade urbana os nós podem ser pessoas, dispositivos ou veículos.

Dessa forma, Chen et al. (2019) apresentam um modelo genérico de geração de contatos em diversos cenários de redes. Especificamente para o cenário de mobilidade urbana, os autores utilizam as bases CONTACT (CHAINTREAU et al., 2007) e HYPERTEXT09 (ISELLA et al., 2011), ambas com dados representando contatos feitos por pessoas. Combinando uma GAN com uma *Graph Convolutional Network* (GCN), os autores modelam os dados de mobilidade como uma rede móvel dinâmica, representada por um grafo, em que a GCN possibilita a identificação das principais características do grafo oriundo dessa rede e, internamente, utilizam uma LSTM, para capturar os padrões de mudanças da topologia do grafo ao longo do tempo. Lei et al. (2019) propõe uma arquitetura similar à anterior, composta de uma GAN formada por uma GCN. A principal diferença é que os autores em propõe uma nova função de perda para tratar das conexões ponderadas entre os vértices do grafo. Também considerando uma estrutura de rede dinâmica, (ZHANG, 2019) modela contatos entre passageiros do sistema de metrô de Washington. O modelo proposto pelo autor define que tanto o Gerador quanto o Discriminador da GAN serão LSTMs.

Considerando a privacidade dos dados em uma base de contatos entre pessoas, Qu et al. (2020) propõe uma GAN que internamente implementa o método de privacidade diferencial durante a geração de dados de mobilidade. Isso garante um maior nível de privacidade em relação à outros métodos de anonimização. Assim como no trabalho anterior, as relações entra as pessoas são estruturadas como um grafo, com a diferença de os grafos que representem os dados são estáticos.

3.5 Geração de tráfego urbano

Apenas um trabalho foi encontrado para essa categoria, que responde à geração de dados referentes às dinâmicas dos tráfegos nas ruas, especificamente ao número de carros

nas interseções em um determinado horário.

No trabalho de (HE et al., 2020), os autores utilizam GANs para recuperar a informação geral do trânsito, considerando analisando interseções específicas do trânsito. Os dados utilizados são estáticos, tratados como imagens e, diferente dos trabalhos anteriores, foram simulados especificamente para os experimentos que seriam realizados. O modelo de GAN proposto pelos autores são constituídos basicamente de perceptrons multicamadas, tanto no Gerador quanto no Discriminador.

3.6 Sumarização dos trabalhos e posicionamento da pesquisa

A Tabela 2 sumariza os trabalhos relacionados à essa dissertação. Embora eficientes para a geração de dados de mobilidade urbana, as propostas aqui apresentadas possuem particularidades que limitam a sua aplicação para outros tipos de dados. Primeiramente, boa parte dos trabalhos analisados trata os dados de mobilidade, que são dependentes do tempo, como sendo dados estáticos. Os modelos apresentados aparentam conseguir gerar dados de mobilidade, mas é evidente que essa abordagem impõe limites nas relações dos dados que os modelos conseguem aprender. Por exemplo Ouyang et al. (2018), Gupta et al. (2018), Song, Baek e Sung (2019), Zhang et al. (2020), Jauhri et al. (2020) tratam os dados como imagens, Yin e Yang (2018), He et al. (2020) modelam os dados como matrizes e Qu et al. (2020) modelam os dados como grafos estáticos.

Nesse sentido, apenas Feng et al. (2020) e Rao et al. (2020) modelam os dados como séries temporais e Lei et al. (2019) e Chen et al. (2019) como grafos dinâmicos. Estes trabalhos, contudo, são limitados em relação ao tipo de dados de entrada. Por exemplo, em Feng et al. (2020) e Rao et al. (2020) os dados precism ser de localizações GPS. Especificamente para Rao et al. (2020), além da localização, o modelo espera receber o dia da semana, a hora e a categoria de ponto de interesse que a localização representa (por exemplo, se a coordenada representa um ponto de alimentação, mercado ou lazer). Outros dados de mobilidade, por exemplo, fluxos de veículos/pessoas, não seriam modeláveis. Por fim, tanto (LEI et al., 2019) quanto Chen et al. (2019) são trabalho específicos para gráfos. Não podendo ser aplicados em que envolvam localizações GPS ou fluxos de trânsito, por exemplo.

Dessa forma, nesta dissertação, propomos a utilização de modelos de aprendizado profundo para geração de séries temporais, como uma alternativa que permita uma maior generalização dos tipos de problemas que podem ser tratados. Por exemplo, diferente dos modelos apresentados anteriormente, que tratam os dados como séries temporais, nossa proposta permite a geração de diferentes tipos de informações de mobilidade urbana, como fluxos no trânsito (número de veículos/pessoas) e velocidade. O único requisito da nossa proposta, atualmente, é que os dados sejam do tipo numérico, especificamente

Tabela 2 – Sumarização dos trabalhos relacionados

Problema abordado	**Trabalho**	**Tipo de entrada**
Geração de trajetórias	Gupta et al. (2018) Ouyang et al. (2018) Song, Baek e Sung (2019) Zhang et al. (2020)	Imagens
	Feng et al. (2020) Rao et al. (2020)	Séries temporais
Geração de dados de transporte	Jauhri et al. (2020) Yin e Yang (2018) Yu et al. (2019) Yu et al. (2020a)	Imagens Matrizes Séries temporais
Geração de contatos	Lei et al. (2019) Chen et al. (2019) Qu et al. (2020)	Grafos dinâmicos Grafos estáticos
Geração de tráfego	He et al. (2020)	Matrizes

pertencentes aos números reais. Assim, podemos gerar dados que envolvem localizações GPS (e consequentemente trajetórias), assim como os trabalhos anteriores. Além disso, uma outra vantagem, em relação aos outros trabalhos descritos, é que não é preciso desenvolver nenhum modelo específico para tratar do problema de geração de dados de mobilidade. Por fim, propomos um arcabouço que permite a utilização de diversos modelos baseados em aprendizado profundo para a geração de séries temporais de mobilidade urbana, facilitando tanto o processo de treinamento de um modelo gerador quanto a avaliação dos modelos treinados.

3.7 Considerações Sobre o Capítulo

Neste capítulo apresentamos os trabalhos relacionados à esta dissertação. A principal limitação dos trabalhos revisados é que, apesar de eficientes, possuem restrições para serem usados em outros cenários de mobilidade urbana que não tenham sido abordados nas pesquisas. Além disso, boa parte das pesquisas revisadas trata os dados de mobilidade urbana como sendo informações estáticas, o que pode prejudicar a correta representação das dinâmicas que dados de mobilidade urbana possuem.

4 O MobDeep

Neste capítulo será apresentado o MobDeep, o arcabouço proposto para geração e avaliação de dados de mobilidade urbana baseado em aprendizado profundo. As próximas seções apresentam os detalhes do arcabouço, com uma visão geral na Seção 4.1 e o pré-processamento que deve ser aplicado aos dados na Seção 4.3. Os três componentes principais do MobDeep são apresentados nas Seções 4.4, 4.5 e 4.6. O código do está disponível no Github[1].

4.1 Visão geral

Dados de mobilidade urbana geralmente são dependentes do tempo e podem ser tratados como séries temporais. Para a geração desse tipo de dado, modelos mais simples, como os puramente estocásticos apresentam limitações quanto às propriedades dos dados que são capazes de modelar. Por outro lado, modelos com maior capacidade de aprendizado, como os baseados em aprendizado profundo, são mais complexos de se utilizar.

Desta forma, o objetivo do MobDeep é facilitar o treinamento e a avaliação de modelos generativos baseados em aprendizado profundo para a geração de dados de mobilidade urbana. Tendo como entrada uma base de dados de mobilidade, é esperado que o MobDeep consiga gerar bases sintéticas com propriedades similares à base de entrada. Além disso, é esperado que os modelos utilizados garantam a privacidade dos dados de entrada. O MobDeep foi implementando na linguagem Python e duas bibliotecas principais: o *tensorflow*[2] e o *keras*[3].

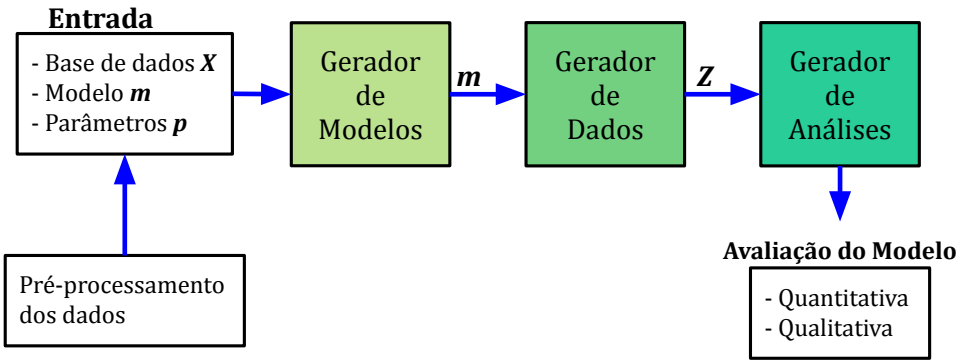


Figura 16 – Visão geral do MobDeep, com seus três componentes principais: Gerador de Modelos, Gerador de Dados e Gerador de Análises

[1] https://github.com/ifribeiro/MobDeep
[2] https://www.tensorflow.org/
[3] https://keras.io/

A Figura 16 apresenta uma visão geral dos principais componentes do MobDeep: o Gerador de Modelos, o Gerador de Dados e o Gerador de Análises. A descrição de cada componente pode ser vista a seguir:

- **Gerador de Modelos**: gera um modelo recebendo como entrada uma base de dados X e o tipo de modelo a ser usado.
- **Gerador de dados**: utiliza um modelo m previamente treinado para gerar novas bases de dados sintéticas Z.
- **Gerador de Análises**: Gera análises quantitativas e qualitativas dos dados gerados. As análises quantitativas são avaliadas automaticamente pelo arcabouço, e servem como um bom indicativo da eficiência do modelo. As análises qualitativas são feitas através de análises visuais dos dados sintéticos e precisam ser feitas manualmente.

#	**date**	**hour**	**count**
1	2011-01-01	0	16
2	2011-01-01	1	40
...	...	...	...
78888	2019-12-31	23	107

Tabela 3 – Exemplo de uma base de dados de entrada para a solução proposta, com as colunas *date* (ano, mês e dia em que as bicicletas foram alugadas), *hour* (hora em que as bicicletas foram alugadas) e *count* (número de bicicletas alugadas).

Uma visão geral das implementações de cada um dos três componentes pode ser vista no Algoritmo 1, em que cada componente é representado por uma função. Para exemplificar o funcionamento do MobDeep suponha que deseja-se gerar bases sintéticas à partir da base da Tabela 3 (essa é uma das bases usadas neste trabalho), utilizando o modelo TimeGAN (uma descriação dos modelos usados pode ser vista na próxima Seção). A base contém informações sobre o número de bicicletas alugadas, por hora, em um sistema de compartilhamento de bicicletas. O número de bicicletas alugadas é representado pela coluna **count** e é a variável que será modelada.

Os valores na coluna **count** constituem um vetor de 78888 valores e, como será explicado na Seção 6.1.2, para os modelos de aprendizado profundo esse vetor precisa ser transformado em um vetor multidimensional X com N registros, T vetores por registro e V variáveis por vetor. Considerando que os dados foram coletados à cada 24 horas, o valor de T será 24. Assim, X será um vetor de dimensões (3287, 24, 1).

Usando a função GeradorDeModelos(X, “timegan”, “exp1”), são gerados diferentes modelos TimeGAN. Internamente, o gerador usa o nome do experimento que será realizado para carregar os testes a serem feitos. Cada teste contém informações como parâmetros

Algoritmo 1: arcabouço para geração de dados de mobilidade

Entrada: Dados reais X de dimensões (N, T, V), modelo *nome_modelo*

```
1  X_treino ← ProcessarDados(X, m);
2  function GeradorDeModelos(X, nome_modelo, nome_exp):
3      lista_testes ← LerTestes(nome_exp);
4      foreach t ∈ lista_testes do
5          TreinarModelo(X, nome_modelo, nome_exp, t)
6  end function
7  function GeradorDeDados(nome_exp, num_z = 10, m = None):
8      if m then
9          z ← GerarDados(nome_exp, m, num_z);
10         SalvarBases(nome_exp, z);
11     else
12         lista_testes ← LerTestes(nome_exp);
13         foreach t ∈ lista_testes do
14             lista_modelos ← LerModelos(t);
15             foreach modelo ∈ lista_modelos do
16                 z ← GerarDados(nome_exp, modelo, num_z);
17                 SalvarBases(nome_exp, t, z);
18 end function
19 function GeradorDeAnalises(nome_exp):
20     lista_testes ← LerTestes(nome_exp);
21     foreach t ∈ lista_testes do
22         lista_modelos ← LerModelos(t);
23         foreach modelo ∈ lista_modelos do
24             z ← LerBaseSintetica(t, modelo);
25             GerarAnalises(z);
26 end function
```

e identificador do teste. Cada teste deve possuir parâmetros diferentes para que seja possível gerar modelos diferentes. Por fim, os modelos gerados são salvos em disco para uso posterior.

Com o nome do experimento, a função GeradorDeDados é utilizada para gerar 100 bases sintéticas que serão salvas em disco, da seguinte forma: GeradorDeDados(“exp1”, 100). Caso um modelo final já tenha sido escolhido, as bases sintéticas podem ser geradas usando-se, além do nome do experimento, o nome do teste e do modelo salvo, por exemplo, GeradorDeDados(“exp1”,100, “t1”, “m_exp1_t1_20”). É importante ressaltar que o GeradorDeDados permite que sejam gerados dados com modelos não gerados pelo usuário, contanto que os modelos tenham sido gerados por meio do MobDeep e sigam o padrão de organização, nos diretórios e nomes de arquivo, definido por ele. Isso permite que usuários diferentes compartilhem os modelos gerados.

Usando o nome do experimento, a componente GeradorDeAnalises irá gerar análises quantitativas e qualitativas para cada base sintética salva nos testes definidos nos experimento. As análises quantitativas são feitas automaticamente e indicam qual modelo possivelmente teve o melhor desempenho na geração dos dados. Análise qualitativa é feita para todos os modelos, porém o modelo indicado pela análise quantitativa tende a ser o melhor em ambos os casos. Após a definição do melhor modelo, o Gerador de Dados pode ser usado novamente para gerar as bases sintéticas dos dados.

A construção do arcabouço em componentes permite que, com recursos computacionais suficientes, seja possível executar cada componente em paralelo. Deste modo, enquanto os modelos são gerados, o Gerador de Dados pode ser utilizado para gerar dados à partir de modelos já salvos e o Gerador de Análises gera as análises de dados sintéticos previamente salvos.

4.2 Modelos Utilizados

Atualmente, o MobDeep possibilita a utilização de modelos puramente estocásticos e baseados em aprendizado profundo. Os modelos do primeiro caso são úteis para bases de dados menores em que não é necessário modelos mais complexos. O modelo disponível nesse caso é o ARIMA, que já foi apresentado anteriormente. No segundo caso, o MobDeep usa três modelos baseados em C-RNN-GAN, RGAN e TimeGAN que são usados em bases grandes e multivariadas. Em versões futuras adicionaremos outros modelos generativos baseados em aprendizado profundo, como os *Variational Autoenconders* (KINGMA; WELLING, 2013; REZENDE; MOHAMED; WIERSTRA, 2014) e modelos estocásticos para séries multivariadas, como o ARIMAX (BOX; TIAO, 1975). Á seguir apresentamos uma breve descrição de cada um dos modelos de aprendizado profundo atualmente disponíveis:

- **C-RNN-GAN** (MOGREN, 2016): um dos primeiros modelos de GAN desenvolvidos para a geração de séries temporais. Em (MOGREN, 2016), o autor utilizou o modelo proposto para a geração de músicas no formato .midi, utilizando bases de dados de diferentes compositores de músicas clássicas. Conceitualmente, a arquitetura do modelo é basicamente a mesma da GAN original, com a diferença de que o Gerador e o Discriminador são modelos LSTMs.

- **RGAN** (ESTEBAN; HYLAND; RÄTSCH, 2017): esse modelo possui praticamente a mesma arquitetura da C-RNN-GAN, onde o Gerador e o Discriminador são modelos LSTMs. Porém, neste modelo, os autores possibilitam que informações categóricas sejam geradas em conjunto com as numéricas. No trabalho, o objetivo principal dos autores foi gerar dados de prontuários médicos de pacientes de um hospital, que são uma base de dados com informações numéricas e categóricas. Entretanto o modelo foi

avaliado, também, com outras séries temporais: uma onda sinusoidal, séries formadas por processos gaussianos e na base de dados MNIST (LECUN; CORTES; BURGES, 1998) (nesse caso, essa base foi tratada como uma série temporal).

- **TimeGAN** (YOON; JARRETT; SCHAAR, 2019): é o modelo mais recente e complexo para a geração de séries temporais. Além do Gerador e Discriminador, comumente presentes em uma GAN, os autores implementam dois componentes adicionais. Uma rede neural *Embedding* e uma *Recovery* que fornecem um mapeamento entre características e espaço latente dos dados, permitindo que a rede adversária aprenda a dinâmica interna dos dados via representações de baixa dimensão. A vantagem desse modelo, é que essas duas redes podem ser implementadas com modelos diversos modelos, permitindo que um modelo de rede neural otimizado para uma determinado problema possa ser utilizado. Nos experimentos, os autores comparam este modelo com os dois modelos anteriores utilizando séries temporais de ondas sinusoidais, ações, energia e eventos (uma base de dados de informações categóricas).

4.3 Pré-Processamento

Esta seção descreve o pré-processamento necessário que deve ser aplicada às base de dados antes da utilização do MobDeep, para os modelos estocásticos e baseados em aprendizado profundo. Para os dois tipos de modelos, é necessário que seja feita a correção de erros e dados faltantes na base de dados de entrada. Diferentes problemas podem exigir técnicas diferentes de correção, por exemplo, dados faltantes podem ser substituídos por 0 ou pela média das observações dos dados, mas é preciso avaliar qual o impacto que uma ou outra técnica pode ter durante o treinamento dos modelos. Por isso assume-se que o usuário já tenha corrigido erros e dados faltantes nos dados.

4.3.1 Modelos Estocásticos

O MobDeep, utiliza modelos ARIMA, logo é preciso garantir que os dados utilizados sejam estacionários. Assim como na correção de erros, existem técnicas diferentes para o mesmo propósito, e assume-se que o usuário tenha realizado transformações adequadas para deixar os dados estacionários, caso já não sejam.

A estabilização da variância, por exemplo, pode ser feita por meio de transformações de raízes quadradas ou da aplicação de logaritmos. A remoção de tendência e sazonalidade pode ser feita com o auxílio de modelos de regressão ou por meio de diferenciações. Uma melhor descrição das transformações citadas pode ser consultada em (MONTGOMERY; JENNINGS; KULAHCI, 2015).

Por fim, atualmente, os modelos ARIMA utilizados ainda não são capazes de tratar

dados multidimensionais. Logo, a base de dados precisa ser transformada para uma vetor com uma única dimensão que, em outras palavras, é o número de observações dos dados.

4.3.2 Modelos Baseados em Aprendizado Profundo

Os modelos baseados em aprendizado profundo atualmente disponíveis no arcabouço são modelos GANs que, por sua vez, utilizam redes neurais recorrentes. Dessa forma, os dados de entrada precisam ser transformados em um vetor multi-dimensional com N registros, T intervalos de tempo (que representam o número de observações por registro) e V variáveis (que representam o número de variáveis em cada observação).

Antes de um modelo ser treinado, os dados que serão enviados para os modelos necessitam passar por algum tipo de processamento, principalmente dados com valores muito grandes (com mais de 2 casas decimais) ou dados heterogêneos, por exemplo, onde uma variável está no intervalo $[0, 1]$ e outra entre $[100, 200]$. Dados com essas características podem dificultar o processo de treinamento e impedir que os modelos convirjam. O processamento realizado é conhecido como normalização e consiste em garantir que os dados possuam valores pequenos (entre 0 e 1, por exemplo) e que todas as variáveis tenham valores aproximadamente no mesmo intervalo (CHOLLET et al., 2018).

Por ser um procedimento comum para esses tipos de modelos, o MobDeep fornece um método auxiliar para a normalização dos dados, usando uma transformação conhecida como Min-Max *Scaler*. Como usamos a linguagem Python para as implementações, utilizamos a função *MinMaxScaler* fornecida pela biblioteca sklearn[4]. Além de realizar a normalização dos dados, a biblioteca permite realizar a transformação inversa. Essa funcionalidade é importante durante a geração dos dados sintéticos. A transformação que a função realiza é definida pelas Equações 4.1 e 4.2, onde X_{min} e X_{max} são, respectivamente, o menor e maior valor presentes na base X e min e max são os intervalos (menor e maior) em que deseja-se normalizar os dados.

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{4.2}$$

O método fornecido pelo MobDeep permite que objeto utilizado para realizar a transformação seja salvo em disco, permitindo que seja usado para realizar as transformações inversas futuramente. Como exemplo, a normalização dos dados da Tabela 3, onde $X = [16, 40, ..., 107]$, resultaria em $X_{scaled} = [0.00785469, 0.00834561, 0.05252823]$.

[4] https://scikit-learn.org

4.4 Gerador de Modelos

O gerador de modelos consiste basicamente do treinamento de modelos usando parâmetros diferentes. Os parâmetros são definidos em arquivos de testes, que contem o nome do experimento a ser realizado e os nomes dos testes (usado para definir os diretórios onde modelos e arquivos importantes serão salvos). Para faciliar a criação dos arquivos, o MobDeep possui alguns exemplos para cada um dos modelos disponíveis. Nesta seção serão apresentados as especificidades do componente gerador de modelos para modelos estocásticos e baseados em aprendizado profundo.

4.4.1 Modelos Estocásticos

Como foi visto na Seção 2.2.3, para modelos estocásticos, como o ARIMA, normalmente é feita uma análise das propriedades estatísticas da base de dados para auxiliar na definição dos parâmetros que devem ser utilizados no treinamento do modelo. Como alternativa, o MobDeep utiliza a biblioteca *pmdarima*[5] que automaticamente identifica os melhores parâmetros para o modelo. Com base em diferentes combinações de parâmetros definidos pelo usuário, ela avalia para quais delas o modelo melhor se ajustou aos dados. Além disso, na Seção 4.3.1 explicamos que é preciso verificar se os dados são estacionários, o que pode ser feito também com a biblioteca. Essa verificação indicará quantas diferenciações precisam ser feitas na série para deixá-la estacionária, o que corresponde ao parâmetro d, do ARIMA.

Para exemplificar com mais detalhes o funcionamento do Gerador de Modelos, considere uma série temporal estacionária. Para a geração dos modelos, é definido um arquivo de experimento contendo uma lista de testes para serem realizados. Cada experimento possui informações importantes para o funcionamento do arcabouço, como identificadores dos testes e parâmetros do modelo. Nesse exemplo, os testes "t1" e "t2". Em "t1", p e q variam de 0 a 2 e em "t2" varia de 0 a 3. Como a série é estacionária, nos dois testes $d = 0$, ou seja, não será preciso realizar diferenciações.

Em cada teste, o Gerador de Modelos identifica a melhor combinação de parâmetros e salva o modelo treinado com essa combinação. Por exemplo, na situação hipotética anteriormente mencionada, para "t1" e "t2" foram salvos os modelos ARIMA(0,0,2) e ARIMA(0,0,3), respectivamente, que foram os modelos que melhor se ajustaram aos dados em cada teste.

4.4.2 Modelos Baseados em Aprendizado Profundo

O funcionamento do Gerador de Modelos para os modelos de aprendizado profundo é similar ao que foi explicado anteriormente. A principal diferença é que, para os modelos

[5] https://alkaline-ml.com/pmdarima/index.html

de aprendizado profundo, em cada teste os modelos serão salvos periodicamente durante o treinamento.

Suponha que deseja-se gerar um modelo de GAN, e para isso, são definidos dois testes, "t1" e "t2". Em "t1" o modelo será treinado por 500 épocas ($num\_epoch = 500$) e será salvo à cada 10 épocas ($save\_int = 10$). Em "t2", o modelo será treinado por 1000 épocas $num\_epoch = 1000$ e salvo à cada 10 ($save\_int = 10$). Em ambos os testes, a taxa de aprendizagem é 0.0001 ($learning\_rate = 0.0001$) Assim, para "t1" e "t2" serão salvos, respectivamente, 50 e 100 modelos.

Salvar modelos periodicamente auxilia o processo de avaliação, principalmente para os modelos de GANs, em que a avaliação periódica é fundamental. Além disso, salvar os modelos que estão sendo treinados evita que, ao se identificar um modelo adequado, seja necessário realizar todo o processo de treinamento novamente.

O Gerador de Modelos, na maioria dos casos, é o que possui a execução mais demorada entre os três componentes, principalmente para os modelos baseados em aprendizado profundo, que possuem dezenas de parâmetros que influenciam no tempo de execução. Uma lista dos parâmetros mais importantes pode ser vista a seguir:

- **num_epochs**: indica o número de iterações pelo qual o modelo será treinado. Um número maior de iterações exige que o modelo seja treinado por mais vezes.

- **learning_rate**: determina a velocidade em que o aprendizado ocorrerá durante o treinamento. Um valor muito baixo deixa o aprendizado lento e pode ser necessário que o valor de *num_epochs* seja maior para permitir que o modelo aprenda corretamente as propriedades dos dados. Um valor muito alto fará com que aconteça oscilações durante o treinamento (em uma época, o erro de ajuste calculado pela função de perda será baixo e, na próxima época, alto) prejudicando a convergência do modelo. Um valor menor para *num_epochs* pode ser usado para controlar o comportamento de um modelo com um *learning_rate* alto.

- **hidden_dimensions**: indica o tamanho das unidades LSTM em cada camada LSTM. Uma camada pode ter de uma à dezenas de unidades por camada. Um modelo com um grande número de unidades por camadas exigirá mais recursos computacionais e levará mais tempo para ser treinado.

- **num_layers**: define o número de camadas LSTM do modelo. Assim como para o *hidden_dimensions*, um modelo com muitas camadas demora mais tempo para ser treinado, pois o algoritmo de retro-propagação precisa atualizar os neurônios em um número maior de camadas. Na C-RNN-GAN é possível definir, separadamente, o tamanho de camadas para o gerador e para o discriminador.

4.5 Gerador de Dados

O funcionamento deste componente consiste em utilizar os modelos criados pelo Gerador De Modelos, descrito na Seção 4.4, para gerar bases de dados sintéticas e salvá-las em disco. É necessário gerar um número considerável de bases sintéticas pois isso permite avaliar a variabilidade dos dados e se os dados gerados apresentam as propriedades dos reais. Os modelos serão lidos com base no nome do experimento utilizado para a geração dos modelos.

Como exibido na função GeradorDeDados do Algoritmo 1, a geração ocorre em duas situações. Na primeira, considera-se que avaliação de todos os modelos do experimento já foi feita, e a geração utilizará apenas o modelo com melhor desempenho. Na segunda, todos os modelos serão utilizados para a geração dos dados. Assim, nesta seção serão apresentadas as especificidades do Gerador de Dados considerando cada tipo de modelo (estocástico e baseado em aprendizado profundo).

4.5.1 Modelos Estocásticos

Os modelos estocásticos salvos possuem métodos específicos que possibilitam a geração de dados com base nas propriedades que o modelo aprendeu. Na primeira situação, o gerador utiliza o modelo escolhido nas avaliações para gerar a quantidade desejada de dados sintéticos. Caso a base usada no treino tenha passado por algum tipo de transformação, as bases sintéticas precisam passar pela transformação inversa para ficarem na mesma escala dos dados reais (sem transformação).

Na segunda situação, o gerador utiliza os modelos salvos em cada teste para gerar os dados sintéticos. Nos testes “t1” e “t2” existe apenas 1 modelo para cada teste definido. Nesse caso, para facilitar as avaliações, os dados gerados estão na mesma escala dos dados usados durante a geração dos modelos.

Como foi mostrado no Algoritmo 1, na função GeradorDeDados, o número de bases sintéticas geradas é definida pelo usuário, por meio do parâmetro num_z. O tempo de geração de dados para cada teste, evidentemente depende do número de testes que serão gerados, entretanto, em comparação com os modelos baseados em aprendizado profundo, a geração é mais rápida.

4.5.2 Modelos Baseados em Aprendizado Profundo

A geração de dados nos modelos baseados em aprendizado profundo é muito similar aos modelos estocásticos e cada modelo, no último caso, possui métodos próprios para a geração de dados sintéticos. A principal diferença é que, após escolhido o melhor modelo, as transformações inversas dos dados sintéticos podem ser feitas automaticamente, caso

tenha sido usado a função de normalização fornecida pelo arcabouço. Assim, durante a geração, os dados sintéticos já estarão na escala dos reais (sem normalização). Se foi utilizado outra técnica de normalização, será preciso realizar as transformações inversas com as técnicas adequadas, após a geração dos dados.

Outra diferença está no fato de os modelos baseados em aprendizado profundo serem mais complexos, tornando a geração de dados mais demorada. Nesse sentido, para acelerar o processo de geração, GPUs podem ser utilizadas. Para a segunda situação da função GeradorDeDados, considerando que deseja-se gerar 100 bases sintéticas para serem avaliadas ($num_z = 100$), usando os modelos gerados pelos testes "t1" e "t2", seria gerado, respectivamente, 5000 e 50000 bases sintéticas, ou seja, 100 bases para cada modelo gerado ou seja, 50 modelos de "t1" e 100 de "t2".

4.6 Gerador de Análises

O Gerador de Análises é o mesmo para os dois tipos de modelos usados e gera dois tipos de avaliação. No primeiro são geradas, de forma automáticas, análises quantitativas dos dados sintéticos e, no segundo, são geradas análises visuais comparando os dados sintéticos com os reais. As próximas seções apresentam, com mais detalhes, os dois tipos de avaliação.

4.6.1 Avaliação Quantitativa

A avaliação quantitativa é feita usando os erros residuais dos modelos treinados. Os erros residuais e de um modelo consistem da diferença entre o que se espera de saída de um modelo (representado por y) e o que o modelo retorna (representado por $\hat{y}$), ou seja, $e = y - \hat{y}$. Por exemplo, suponha que era esperado que um modelo tivesse como saída uma sequência de valores $y = [0.5, 0.8, 0.01]$, mas produziu a sequência $\hat{y} = [0.01, 0.05, 0.8]$. Para esse modelo, o erro residual e será

$$\begin{aligned} e &= y - \hat{y} \\ e &= [0.5, 0.8, 0.01] - [0.01, 0.05, 0.8] \\ e &= [0.49, 0.75, -0.79] \end{aligned}$$

com $\mu(e) = 0.15$ e $\sigma(e) \approx 0.673$, respectivamente, a média e o desvio padrão dos erros residuais.

Um modelo que tenha capturado com eficiência as propriedades dos dados reais, terá resíduos próximos de 0. Assim, se caso o modelo anterior tivesse gerado a saída ($\hat{y}$) exatamente como esperado (y), tanto a média quanto o desvio padrão seriam 0. Entretanto, o ideal é que os dados sintéticos gerados pelos modelos possuam variabilidade, ou seja, cada base é diferente entre si e, ao mesmo tempo, possuam as propriedades dos dados

reais. Em outras palavras, a média e o desvio padrão precisam ser pequenos, mas não exatamente 0.

Na análise quantitativa, o MobDeep classifica como o melhor modelo aquele que, para os erros residuais de um número num_z de bases sintéticas, possui a menor média, em que $\mu > 0.000$, e que cujo desvio padrão está no intervalo $[0.000, \tau]$, em que τ é um limiar que indica o nível de variabilidade aceitável para os dados. Um τ muito alto, por exemplo $\tau = 0.7$, vai classificar como bons modelos que não conseguiram capturar com eficiência as propriedades dos dados.

As Seções 4.6.1.1 e 4.6.1.2 a seguir explicam como os resíduos para modelos estocásticos e baseados em aprendizado profundo são obtidos.

4.6.1.1 Modelos Estocásticos

Resíduos gerados por um modelo ARIMA são fáceis de obter. O modelo que gerou os dados sintéticos é ajustado usando cada base gerada e, após cada ajuste, utiliza-se uma função fornecida pelo modelo que retorna o resíduo do modelo para a base sintética.

Levando em consideração as etapas do ModDeep, foi gerado e salvo um modelo, por exemplo, ARIMA(0,0,1) e o gerador de dados utilizou esse modelo salvo para gerar os dados sintéticos. O Gerador de Análises vai utilizar o mesmo modelo salvo, ajustar na base sintética gerada e obter os resíduos para a base. Após a realização desse processo em todas as bases sintéticas geradas pelo modelo ARIMA(0, 0, 1), será calculada a média e o desvio padrão para aquele modelo.

4.6.1.2 Modelos Baseados em Aprendizado Profundo

Como nesses modelos os dados sintéticos são gerados utilizando redes neurais recorrentes (RNN), os resíduos também precisam ser gerados através de uma RNN. Não faz sentido, por exemplo, utilizar um modelo ARIMA para obter o resíduo de uma base de dados que foi produzida por uma LSTM, já que a LSTM possui uma capacidade muito superior de aprender as relações temporais dos dados e funciona de maneira diferente de um modelo ARIMA. O processo de geração de resíduos para os modelos baseados em aprendizado profundo envolve 4 etapas principais:

(a) A base de treino é dividida em vetores de entrada X e saída y. No treinamento de redes neurais, é comum utilizar uma parte da base para treino e outra para teste. Como estamos avaliando a geração dos dados, usamos a base toda como treino.

(b) Uma RNN é treinada com X (entrada) e y (saída). Após o treinamento, a rede é utilizada para prever a saída esperada y_{pred}. A previsão é feita utilizando X como entrada.

(c) Cada base sintética gerada pelos modelos é divida em entrada X_z e saída y_z. Nesse caso, apenas y_z será utilizado.

(d) O erro residual de cada modelo é calculado usando $y_{pred} - y_z$. Como, internamente, os dados sintéticos foram produzidos por um modelo RNN, é esperado que, se o modelo aprendeu corretamente as características dos dados, y_z e y_{pred} possuirão propriedades similares.

É importante ressaltar que os erros residuais não são uma métrica para avaliar a similaridade entre os dados sintéticos e reais. Os erros informam a eficiência dos modelos em gerar dados sintéticos que mantenham as propriedades dos reais. Por exemplo, um modelo que consegui gerar dados sintéticos com a mesma sazonalidade dos dados reais será, evidentemente, melhor que um outro em que os dados não apresentam nenhuma sazonalidade.

Por esse motivo, no cálculo dos erros, utiliza-se a saída esperada y_{pred} do modelo treinado com X e não y. Ademais, os erros residuais informam se os dados gerados possuem variabilidade, pois um modelo que sempre gera os mesmos dados sintéticos apresentará os mesmos valores para os erros residuais.

Por fim, a RNN utilizada para a geração dos resíduos deve ser a mesma usada nos modelos geradores. Os três modelos GANs disponíveis permitem que seja utilizado, durante a geração, um modelo LSTM ou GRU. Pelo mesmo motivo anteriormente citado, modelos diferentes tem capacidades diferentes de aprendizado e um modelo adequado deve ser escolhido. Dessa forma, o Gerador de Análises permite que se escolha entre um modelo LSTM ou GRU.

4.6.2 Avaliação Qualitativa

A avaliação qualitativa consiste em se gerar análises visuais comparando cada uma das variáveis dos dados sintéticos com suas correspondentes nos dos dados reais. Enquanto a avaliação quantitativa indica o melhor modelo entre os gerados, a qualitativa auxilia na identificação de imperfeições dos modelos, por exemplo, relações temporais que ele não capturou com eficiência. A desvantagem, nesse caso, é que não é possível automatizar o processo de comparação, que precisa ser feita manualmente pelo usuário.

Atualmente, o Gerador de Análises gera apenas um tipos de análises visual, que serão descritas nas próximas seções. Os exemplos apresentados consideram bases com apenas uma variável, mas o funcionamento é o mesmo para dados multivariados.

4.6.2.1 Soma por intervalo

Dependendo do tipo de dados utilizado, pode fazer sentido que os valores sejam agrupados por intervalos, por exemplo, à cada hora do dia. Dessa forma, a soma por intervalo agrupa as observações dos dados por cada dia da semana e soma os valores em cada intervalo.

Para exemplificar, considere uma das bases que foi utilizada neste trabalho, que corresponde à contagem de bicicletas alugadas por hora em cada dia, denominada de *BikeSharing*. A soma por intervalo irá somar, para cada dia da semana, todos os valores observados às 0hs, 1hs, 2hs, ..., 24hs tanto das bases sintéticas quanto das reais. Assim é possível comparar o comportamento das duas bases em cada dia e verificar se são similares. Para facilitar a comparação, cada visualização gerada pelo MobDeep irá representar duas séries temporais, representando os dados reais e sintéticos dos dias da semana.

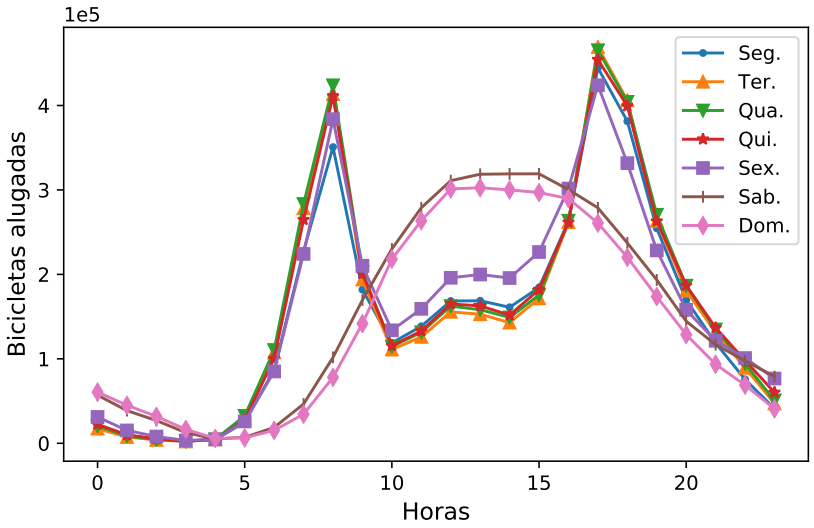


Figura 17 – Exemplo de agrupamento dos dados por dia da semana à cada hora do dia da base *BikeSharing*. Os dados são agrupados por dia da semana e os valores de cada hora para cada agrupamento são somados.

A Figura 17 mostra as visualizações da soma por intervalo apenas dos dados reais para a base *BikeSharing*. Pela figura fica evidente algumas propriedades dos dados, como horários com maior número de bicicletas alugadas durante dias úteis ($\approx 8h$ e $\approx 17h$) finais de semana ($\approx 13h$). De maneira geral, cada dia apresenta uma leve diferença entre si, sendo mais clara a disparidade entre dias úteis e finais de semana.

A vantagem de se utilizar esse tipo de visualização é que ela permite a identificação de propriedades claras nos dados, como pontos de pico e diferenças entre cada dia da semana. Uma outra vantagem é que a soma por intervalo permite a utilização de toda a base de dados para realizar as comparações. É esperado que um bom modelo consiga gerar dados sintéticos com propriedades similares (não exatamente iguais), por exemplo,

os picos de durante os dias úteis.

Para o agrupamento dos dados, é necessário a data inicial (com horas e minutos) e a frequência (segundos, minutos, horas ou dias) em que eles foram coletados. Essas informações são usadas para obtenção do dia da semana e hora em que cada observação ocorreu, que são fundamentais para o agrupamento.

4.7 Limitações do MobDeep

É importante ressaltar, contudo, algumas limitações do arcabouço proposto. A primeira delas diz respeito à organização dos dados. Os modelos usados para a geração são otimizados para tratar de séries temporais, logo, os dados de mobilidade precisam ser organizados como tal.

Uma outra limitação refere-se ao tipo de informação presente na série temporal. Uma série pode ser uma sequência de variáveis numéricas e categóricas (como textos ou valores que representem alguma classe) ou variáveis mistas. Embora alguns dos modelos utilizados (RGAN e TimeGAN) tenham sido utilizados para geração de séries com variáveis categóricas, neste trabalho, a utilização dos modelos considera que os dados nas séries temporais são variáveis numéricas pertencentes ao conjunto dos números reais $\mathbb{R}$. Dessa forma o arcabouço precisa ser avaliado em situações em que a série temporal de mobilidade possui variáveis numéricas e categóricas.

Ainda sobre o tipo de série temporal, o MobDeep espera entradas que são definidas pela sumarização de algum tipo de dado de mobilidade (por exemplo, número de carros nas ruas em um determinado horário). Dessa forma, precisa ser avaliado considerando outras fontes de dados, como séries temporais contendo localizações. Os bons resultados encontrados, porém, sugerem que a modelagem desse tipo de dados é possível com a solução proposta.

Considerando as avaliações aplicadas aos dados gerados pelos modelos, uma limitação do arcabouço proposto é que as análises qualitativas, que precisam ser avaliadas manualmente, são feitas para cada variável presente na base de dados. Assim, avaliação visual de uma base com um conjunto de variáveis muito grande pode se tornar muito custosa.

Finalmente, os modelos de aprendizado profundo foram construídas para tratar de outros problemas, não especificamente, o de geração de dados de mobilidade urbana. Dessa forma, uma limitação deste trabalho é compreender como otimizar esses algoritmos considerando a geração de dados de mobilidade urbana.

4.8 Considerações Sobre o Capítulo

Este capítulo apresenta o MobDeep, um arcabouço baseado em aprendizado profundo proposto para a geração de dados de mobilidade urbana. São apresentados a visão geral do MobDeep e seus componentes principais, constituídos do Gerador de Modelos, Gerador de Dados e Gerador de Análises.

O MobDeep foi proposto para gerar modelos de mobilidade a partir de algoritmos baseados em aprendizado profundo. Para investigar se modelos estocásticos podem gerar dados com similaridade e menor complexidade, o arcabouço permite a utilização de modelos estocásticos. A construção por meio de componentes separados permite que cada um deles seja executado em paralelo, otimizando a geração dos dados e que novos algoritmos sejam futuramente inseridos.

5 Dados de Mobilidade Urbana

Este capítulo descreve as base de dados que foram utilizadas durante os experimentos realizados neste trabalho. Utilizamos uma base de dados pública contendo informações sobre o compartilhamento de bicicletas (aqui referido como *BikeSharing*) e uma base de dados de uma rede social de motoristas, denominada de *Trânsito Vitória*. Nas próximas seções serão apresentadas as propriedades de bases utilizadas.

5.1 *BikeSharing*

É uma base de dados que contém informações sobre a alocação de bicicletas em 7 cidades dos Estados Unidos (*Washington*, *Arlington*, *Alexandria*, *Montgomery*, *Prince George's County*, *Fairfax County* e *City of Falls Church*) entre 2011 e 2019, através do serviço de locação de bicicletas *Capital Bikeshare*[1]. A localização das cidades pode ser vista na Figura 18.

A base de dados entre Janeiro de 2011 a Dezembro de 2012 foi obtida de (FANAEE-T; GAMA, 2013) e os dados de Janeiro de 2013 a Dezembro de 2019 foram coletados do site da *Capital Bikeshare*. No site, a forma como os dados estão organizados varia entre

[1] https://www.capitalbikeshare.com

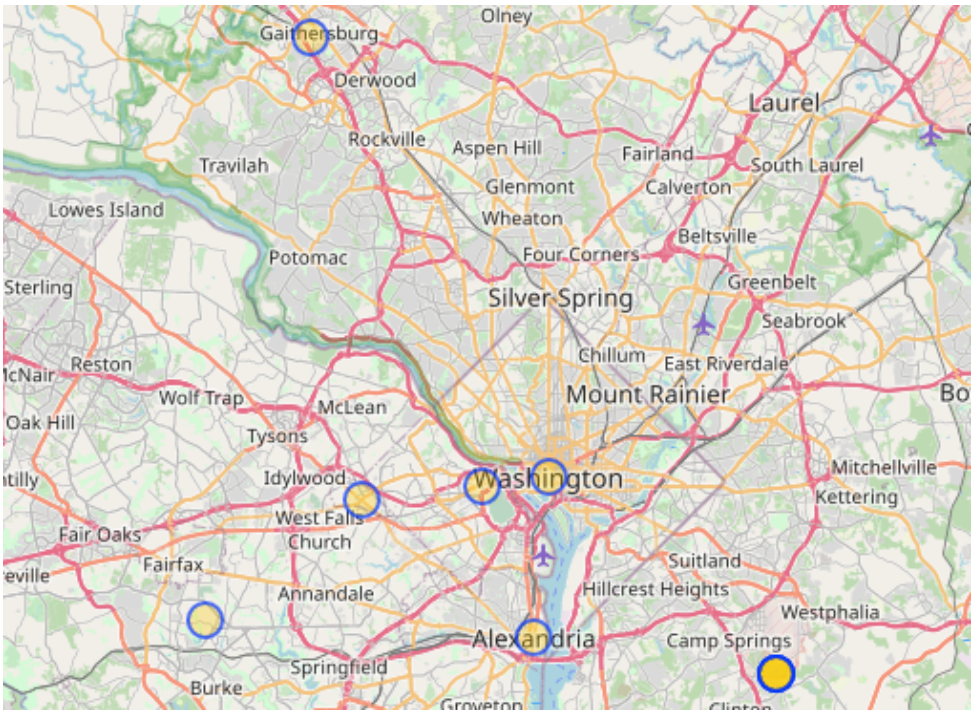


Figura 18 – Localização das 7 cidades no Estados Unidos onde o serviço *Capital Bikeshare* opera.

cada ano. De 2012 a 2017 os dados anuais podem ser baixados em um mesmo arquivo compactado e separados quadrimestralmente. Entretanto, de 2018 a 2019, os dados são disponibilizados quadrimestralmente em arquivos distintos, iniciando-se pelo primeiro quadrimestre de 2018 e finalizando com o último quadrimestre de 2019. Os atributos dessa base e suas respectivas descrições podem ser vistas na Tabela 4.

Tabela 4 – Descrição dos atributos da base de dados *BikeSharing* de 2011 a 2019.

Atributo	**Descrição**
Duration	Duração da viagem
Start Date	Data e hora do início da locação
End Date	Data e hora do fim da locação
Start Station	Nome e número da estação de origem
End Station	Nome e número da estação de destino
Bike Number	Identificador da bicicleta usada na viagem
Member Type	Indica o tipo de membro (registrado ou casual)

5.1.1 Simulação da base *BikeSharing*

Considerando as informações presentes na base de dados *BikeSharing* exibidas na Tabela 4, neste trabalho optamos por modelar e simular o número de bicicletas alugadas durante um intervalo de tempo, nesse caso, à cada hora do dia. Assim para o intervalo entre 2011 e 2012 o número de bicicletas alugadas, por hora, já está contabilizado. No intervalo subsequente, como não há informação sobre o número de bicicletas que foram alugadas, foi preciso que calculássemos usando as colunas disponíveis. Nesse caso, usamos as colunas *Start date* e *Bike number* e computamos o número de bicicletas alugadas no intervalo de 1 hora.

Dessa forma, obteve-se as informações de data, hora e número de bicicletas alugadas, mostradas na Tabela 5. Durante o treinamento dos modelos, apenas a coluna **cnt** é usada. As colunas de data e hora são usadas nas análises visuais dos dados, que serão apresentadas nas próximas seções.

Tabela 5 – Base de dados resultante após a contabilização do número de bicicletas alugadas.

data	**hora**	**cnt**
2011-01-01	0	16
2011-01-01	1	40
...	...	...
2019-12-31	23	107

5.1.2 Tendência e Sazonalidade

A Figura 19 apresenta a visualização no número de bicicletas alugadas por hora, entre 01 de Janeiro de 2011 e 31 de Dezembro de 2019. De acordo com a figura, o número de locações de bicicletas alugadas apresenta uma tendência de crescimento, que ocorre até o ano de 2019. A partir de 2019, o número de bicicletas alugadas aparenta ter um leve decrescimento em relação aos dois anos anteriores.

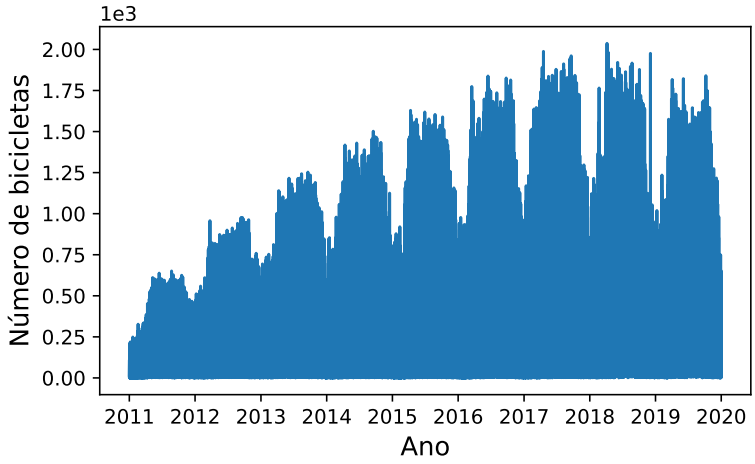


Figura 19 – Visualização dos dados da base *BikeSharing* de Janeiro de 2011 a Dezembro de 2019. Nota-se a presença clara de uma sazonalidade anual nos dados, com menos bicicletas sendo alugas no início de cada ano.

A Figura 19 mostra, também, que há uma sazonalidade nos números de bicicletas alugadas entre os anos observados, por exemplo, no início e fim de cada ano o número locações de bicicletas tende a ser menor que no meio. Outra sazonalidade é identificada quando analisamos um intervalo menor dos dados, como pode ser visto na Figura 20 que apresenta o primeiro mês da base *BikeSharing*.

apresenta comportamentos bem claros de números de bicicletas alugadas ao longo do tempo. Os picos, observados na figura, representam, aproximadamente, 8hs e 17hs de cada dia. O padrão claro observado nessa base auxiliou na definição do parâmetro *batch size* (tamanho do lote de dados) utilizado nos modelos baseados em aprendizado profundo. Nesse caso, os modelos precisam ser treinados, iterativamente, com um conjunto de dados que possua uma boa representatividade da base real. Assim, assume-se que usar 1 mês à cada iteração é o suficiente para que os modelos aprendam as correlações temporais dos dados. Apesar dos números de bicicletas no primeiro mês variarem de 0 a 250, nota-se, pelos histogramas das bases de dados na Figura 29, que a base *BikeSharing* (Figura 21) pode variar até 2037.

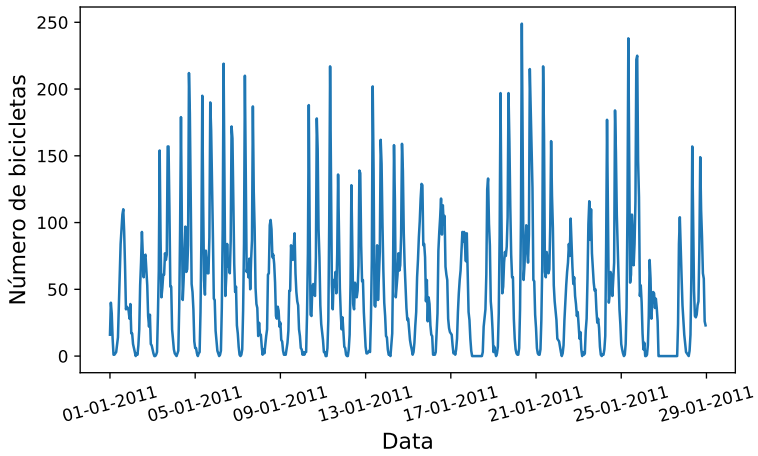


Figura 20 – Um mês de observação da base de dados *BikeSharing*. Cada par de picos de observações representa, aproximadamente, as 8 e 17hs de um dia. Os picos com menores valores representam os finais de semana.

5.1.3 Variabilidade

O histograma da base *BikeSharing* é mostrado na Figura 21. Em toda a base de dados, a maior frequência de bicicletas alugadas ocorre entre 0 e ≈ 200, com 37312 observações nesse intervalo. Em média, foram alugadas ≈ 329 bicicletas por dia, entre 01 Janeiro de 2011 e 31 Dezembro de 2019, com um desvio padrão de 337 bicicletas.

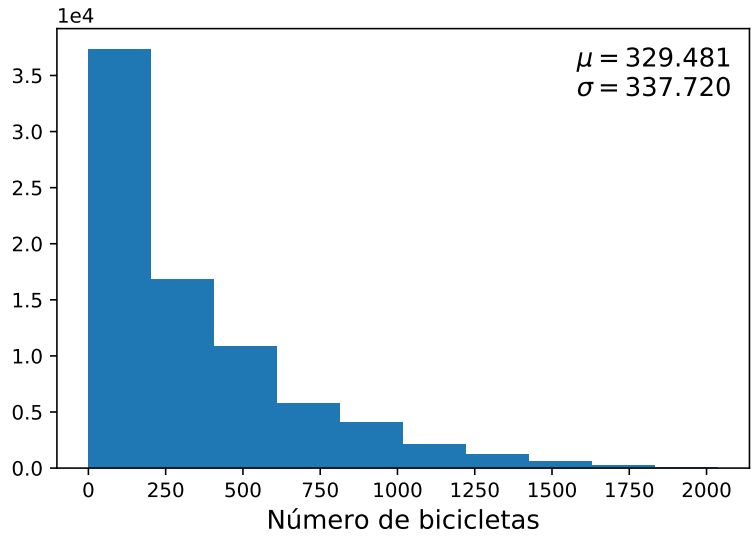


Figura 21 – Histograma do número de bicletas alugadas em cada hora do dia, da base de dados *BikeSharing*. Nota-se, pela distribuição do histograma, que o número de bicicletas alugadas por hora é, em maioria, inferior à 250.

Apesar de uma média de 329 bicicletas alugadas, alguns dias chegaram a apresentar número de locações de ≈ 1833 à ≈ 2037. No histograma, valores nesse intervalo ocorrem 33 vezes. Como visto nas Figura 20, os dados, durante o dia, tendem a oscilar entre valore muito altos e próximos de 0. Além disso, na Figura 19 os dados apresentam uma tendência

de crescimento ao longo dos anos. Essas características justificam o valor alto para o desvio padrão da base *BikeSharing*.

5.1.4 Soma por Intervalo

À fim de ter uma visão geral do comportamento dos dados, agrupamos a base de dados por cada um dos 7 dias da semana e somamos os valores observados em cada hora do dia. Dessa forma é possível visualizar com maior clareza como o dia da semana e hora do dia considerado influencia no número bicicletas alugadas.

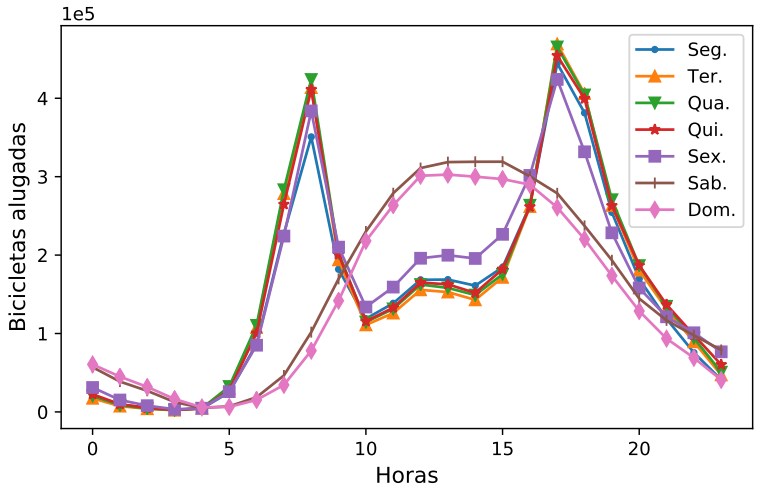


Figura 22 – Soma por intervalo da base de dados *BikeSharing*. O número de bicicletas alugadas é maior por volta das 8 e 17hs de segunda à sexta. Nos finais de semana o número de bicicletas é maior às 13hs.

A Figura 22 apresenta a soma por intervalo da base *BikeSharing*, de Segunda à Domingo. Os picos observados na Figura 20, que mostra o primeiro mês de dados, ficam mais evidentes quando agrupados por dia. Pela Figura 22 percebe-se que os dias úteis apresentam comportamento muito similares, apresentando um número maior de bicicletas alugadas por volta das 8 e 17hs, com picos que variam entre 350000 e 450000 locações às 8hs e 420000 e 480000 às 15hs.

Ainda é possível notar que há uma diferença clara entre os dias úteis e finais de semana. Nesse último caso, há uma maior locação de bicicletas por volta das 13hs, com picos de $\approx$ 300000. Assim como nos dias úteis, os dados se comportam de forma bem parecidas nos Sábados e Domingos.

5.2 Trânsito Vitória

Consiste em uma base de dados composta por informações sobre o trânsito da região da Grande Vitória, no Espírito Santo e é resultado de uma parceria entre a prefeitura

de Vitória e o aplicativo de trânsito para dispositivos móveis *Waze*. A parceira ocorreu através do programa *Connected Citizens Program* (CCP) (CCP, 2021), um serviço de colaboração entre o aplicativo e organizações de gerenciamento de trânsito, iniciado pelo *Waze* em 2014, que visava compartilhar informações em tempo real sobre o trânsito da região onde o aplicativo operava.

No *Waze*, os usuários reportam eventos atípicos nas vias, por exemplo, acidentes, alagamentos ou engarrafamentos. Além disso, o aplicativo mensura os níveis de congestionamento e contabiliza o número aproximado de carros nas vias congestionadas. Por ser um serviço de *croudsourcing*, a eficiência de utilização do aplicativo depende do número de usuários conectados gerando informações sobre o trânsito (LENKEI, 2018).

Ao fazer parte do CCP, uma organização passa a ter acesso, por meio de uma API fornecida pelo *Waze*, à três tipos de bases de dados: alertas reportados (anonimamente) por usuários do aplicativo; informações sobre lentidão no trânsito que são geradas pelo sistema; e irregularidades, como congestionamentos não usuais identificados pelo aplicativo. Em contrapartida, a organização parceira do *Waze* pode enviar informações oficiais ao *WAZE* tornando mais confiável a informação fornecida pelo aplicativo, como ruas fechadas ou algum incidente identificado por agências que monitoram o trânsito.

Assim, através de uma API, a prefeitura de Vitória recebe as informações em tempo real em intervalos de 5 minutos. É importante ressaltar que esta é uma base de dados esparsa, pois utiliza informação inseridas pelos usuários e contém informações de uma via apenas quando esta apresenta congestionamento. Deste modo, alguns horários ou mesmo dias podem não apresentar alguns registros.

Tabela 6 – Descrição de cada atributo presentes na base de dados de alertas fornecidas pelo *Waze*

Atributo	**Descrição**
uuid	Identificador único do alerta
street, city	Respectivamente, rua e cidade onde o evento foi reportado
confidence	Indica a confiança do evento reportado, com base no feedback de outros usuários. Varia entre 0 - 10
reliability	Indica a confiabilidade do evento reportado com base no feedback de outros usuários. Varia entre 0 - 5
type, subType	Tipo e subtipo do evento, respectivamente. A inserção do subtipo é opcional
roadType	Tipo da rua onde o evento foi reportado (Rua, Avenida, etc)
location	Coordenadas geográficas de onde o evento foi reportado
eventDate	Data em que o evento foi reportado
speed	Velocidade média atual no segmento engarrafado da via
reportDescription	Breve descrição do evento

Entre as três bases de dados fornecidas pelo *Waze* (alertas, congestionamento ou

lentidão no trânsito e irregularidades) , utilizamos a base com informações sobre alertas gerados pelos usuários. A Tabela 6 mostra os atributos presentes na base de dados de alertas e suas respectivas descrições, obtidas de (LENKEI, 2018; THOMÉ et al., 2020).

Tabela 7 – Sumarização da base de dados Trânsito Vitória

Registros	Uuids	Ruas	Cidades	Intervalo
1695712	89198	864	6	17/03/2019 - 02/12/2019

A sumarização da base de alertas usada neste trabalho pode ser encontrada na Tabela 7. A tabela mostra que os ≈ 1.7 milhão de registros na base de dados foi gerado por ≈ 90 mil eventos reportados. Isso significa que os eventos ocorriam mais de uma vez e em datas distintas. Além disso, é importante ressaltar que embora a parceria tenha sido feita com a prefeitura de Vitória, a base foi preenchida com informações de outras cidades do Espírito Santo, onde o *Waze* operava. Como mostra a Figura 23, a maioria dos eventos ocorreu na cidade de Vitória com ≈ 1.45 milhões eventos registrados. As outras cidades, em conjunto, apresentaram ≈ 65 mil eventos.

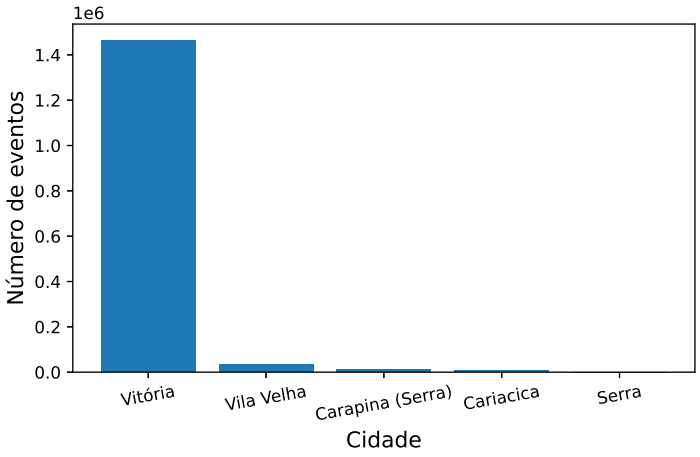


Figura 23 – Número de eventos por cidade entre Março e Dezembro de 2019. A cidade de Vitória apresentou o maior número de eventos reportados, com ≈ 1.45 milhões.

Dentre as diversas informações presentes na base de dados, nosso objetivo foi modelar o número de carros nas ruas em um determinado intervalo de tempo. A Tabela 6 mostra que cada evento reportado por um usuário possui um identificador único, ou seja, número de eventos na base de dados está relacionado ao número de carros. Assim é possível obter um valor aproximado com base no identificador do evento em que, ao assumir que cada evento é gerado por um único usuário, obtemos um valor mínimo de carros nas ruas. Para evitar que o mesmo carro seja contabilizado mais de uma vez no intervalo considerado, os carros repetidos (com o mesmo uuid) são removidos da contabilização.

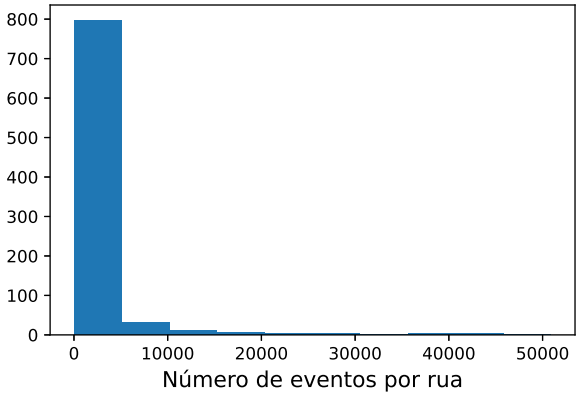


Figura 24 – Histograma do número de carros por rua. A maioria das ruas apresentou um valor baixo de carros, com ≈ 92% delas apresentando apenas 5962 carros.

Desta forma, nos experimentos realizados neste trabalho, utilizamos as colunas: *uidd*, para contar o número de carros em cada rua; *street*, para especificar as ruas que serão utilizadas; e a coluna *eventDate* para selecionar um intervalo de tempo específico, bem como agrupar os dados por dia da semana. Nesse sentido, removemos os eventos que ocorreram na Avenida Leitão da Silva que, durante o ano de 2019, estava em processo de reformas. Por conta disso, não seria possível que os usuários que reportaram algum evento nessa rua estivessem, de fato, nela. O histograma da Figura 24 mostra que a maioria das ruas apresentou poucos carros entre Março e Dezembro de 2019. Por exemplo, 796 ruas (≈ 92%) apresentaram apenas 5962 carros. Em outras palavras, a grande maioria dos carros esteve em apenas 8% das ruas.

Tabela 8 – Oito primeiras ruas da base Trânsito Vitória com números de carros superiores à 25000. Para facilitar o entendimento, cada rua foi associada à um nome fictício que representa sua posição em relação ao número de carros.

Nome real	**Nome usado**
Av. Nossa Senhora da Penha	Rua I
Av. Robert Kennedy	Rua II
Av. Norte Sul	Rua III
Av. Fernando Ferrari	Rua IV
Av. Vitória	Rua V
Rod. São Geraldo	Rua VI
Av. Dante Michelini	Rua VII
Av. Mal. Mascarenhas de Moraes	Rua VIII

Considerando que a maioria dos carros esteve em poucas ruas, decidimos utilizar aquelas com os maiores valores de carros. Selecionamos, portanto, as ruas com um número de carros maiores que 25000, que são apresentadas na Tabela 8. Para simplificar as referências às ruas ao decorrer deste texto, atribuímos um nome fictício representando a ordem de cada uma, variando entre I e VIII. Exceto pela Rua VI, que é uma rodovia,

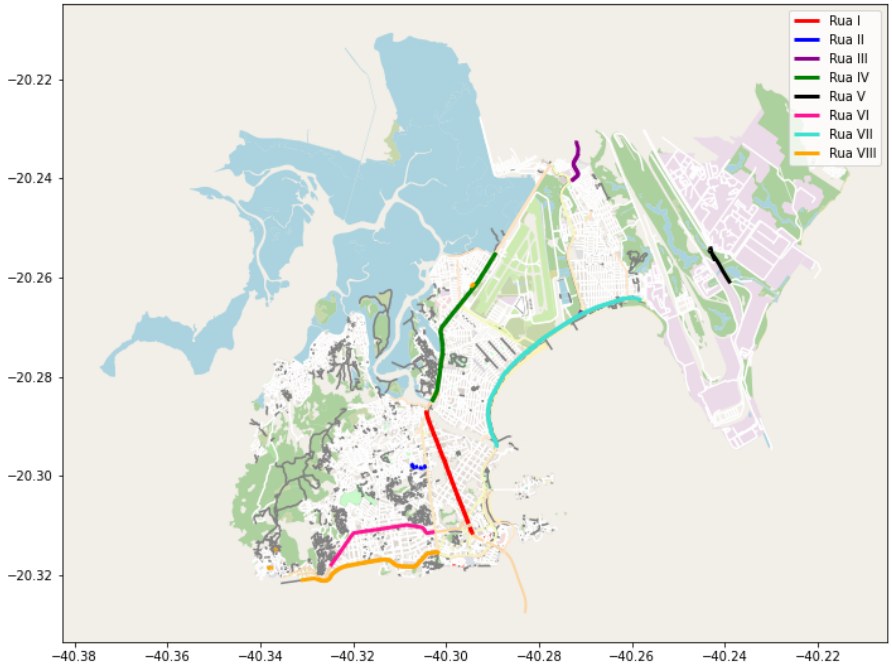


Figura 25 – Mapa da cidade de Vitória, com destaque para as oito ruas utilizadas. Sete das oito ruas mostradas na figura são avenidas

todas as outras ruas são avenidas.

Uma mapa da cidade de Vitória pode ser visto Figura 25, com destaque para as 8 ruas utilizadas. Pelo mapa é possível notar que a há três grandes ruas (nesse caso, avenidas) cortando a cidade (Rua I, Rua Rua IV e Rua VII). As Ruas I e IV possuem uma conexão entre si, por isso é esperado que essas ruas apresentem comportamentos similares. Nesse sentido, as Ruas II, VI e VIII estão relativamente próximas da Rua I. Os número de carros nessas ruas possivelmente sofrem influência do fluxo gerado pela Rua I. Por fim, as as Ruas III e V se mostram isoladas no mapa.

A Figura 26 mostra o número de carros em cada uma das oito ruas, entre Março e Dezembro de 2019. Os valores observados mostram que não há uma variação considerável entre uma rua e a próxima, onde a rua com o maior número de carros (Rua I), possui pouco menos que o dobro de carros que a rua com o menor número (Rua VIII), respectivamente, 50916 e 26896.

Para os experimentos, dividimos a base de dados Trânsito Vitória em dois tipos. No primeiro, cada rua é usada, em cada modelo, como uma base de dados separada. Isso possibilita entender o desempenho dos modelos quando considera-se apenas uma única

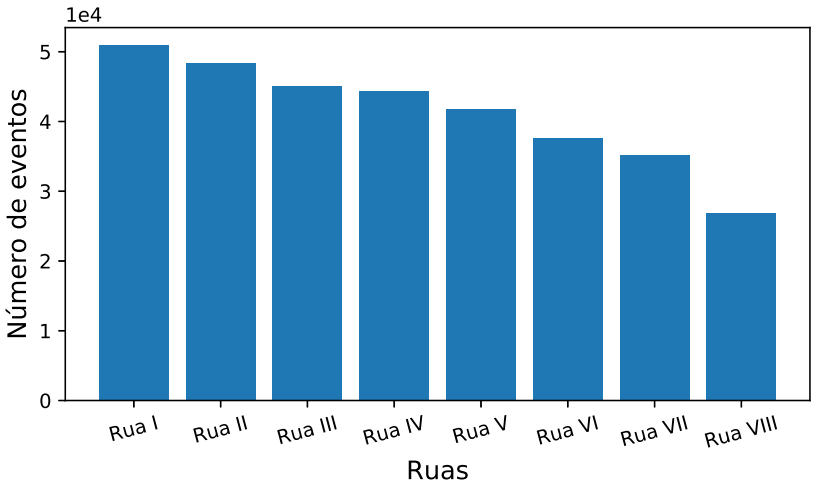


Figura 26 – Oito primeiras ruas com número de carros maiores que 25000, entre Março e Dezembro de 2019. Nota-se que as ruas apresentam um número muito próximo de carros.

rua. Para esse tipo, usamos as Ruas I, II e III. No segundo, usamos mais de uma rua em uma única base de dados, em que cada rua é uma variável à ser modelada. Essa base permite avaliar o desempenho dos modelos em situações onde há mais de uma variável sendo modelada ao mesmo tempo. Para esse caso, usamos as Ruas de IV à VIII. Como essas ruas iniciam e terminam e horários diferentes, utilizamos o dados compreendidos no mesmo intervalo, que vai de 28 de Março de 2019 à 02 de Dezembro do mesmo ano. Para facilitar a leitura, essa base é denominada de Ruas IV_VIII.

As próximas seções apresentam as base de dados selecionadas à partir da base Trânsito Vitória e as propriedades temporais dos dados, considerando as Ruas I, II e III, individualmente, as Ruas de IV à VIII, como uma única base de dados multivariada.

5.2.1 Simulação da base Trânsito Vitória

Com as informações presentes na base de dados Trânsito Vitória, exibidas na Tabela 6, neste trabalho optamos por simular o número de carros presentes nas ruas em um determinado horário, nesse caso, em intervalos de 30 minutos de cada dia. Para isso utilizamos as colunas **uuid** (para contagem do número de carros), **street**, para seleção das ruas, e **eventDate** para obter o horário.

Como usamos dois grupos de bases (um grupo com uma variável e outro multivariado) considerando as informações da Trânsito Vitória, as contabilizações dos números de carros resultaram em dois tipos de bases de dados. Assim, na Tabela 9 tem-se as informações presentes nas bases com apenas uma variável e na Tabela 10 as informações das bases multivariadas.

Tabela 9 – Estrutura da base de dados resultante paras as Ruas I, II e III após a contabilização do número de carros

data	hora	cnt
2011-01-01	0	16
2011-01-01	1	40
...	...	...
2019-12-31	23	107

Tabela 10 – Estrutura da base de dados resultante paras as Ruas de IV a VIII após a contabilização do número de carros

data	hora	Rua IV	Rua V	Rua VI	Rua VII	Rua VIII
2019-03-28	0	0	0	0	0	0
2019-03-28	1	0	0	0	0	0
...	...	...	...	...	...	...
2019-12-12	23	1	1	1	1	1

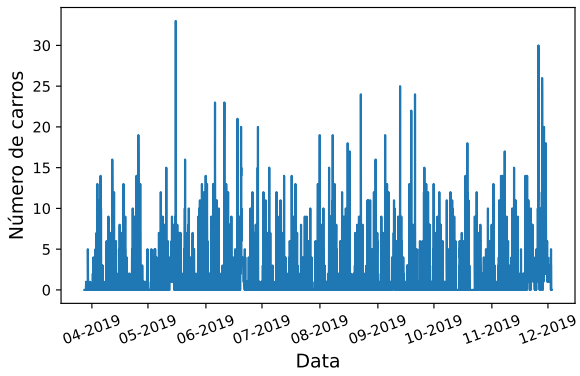
5.2.2 Tendência e Sazonalidade

Identificar tendencias e repetições (sazonalidade) nos dados reais é essencial entender quais principais características os dados sintéticos gerados à partir desses dados precisam possuir. Desta forma, a Figura 27 apresenta a visualização dos números de carros nas Ruas I, II, III. Nas Ruas I (Figura 27a) e III (Figura 27c), as observações do número de carros nas ruas aparentam repetir-se ao longo do tempo, porém com uma sazonalidade menos evidente que a base *BikeSharing*, anteriormente mostrada.

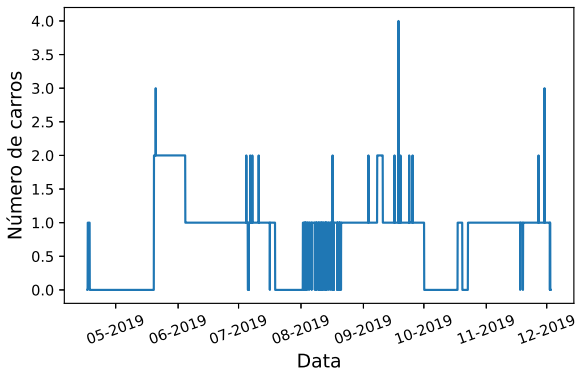
A Rua II (Figura 27b), entretanto, não apresenta nenhuma repetição no número de observações para o primeiro mês, com apenas os dias iniciais apresentando carros na rua e, em toda a base, não é possível identificar repetições nas observações dos dados. Além disso, em todas as três bases não há a presença de uma tendência de comportamento dos dados, por exemplo, de crescimento ou decrescimento. As três apresentam dados aparentemente constantes.

Dados com presença clara de repetição de comportamentos podem tornar o treinamento mais fácil, e, no caso das GANs, fazer com que o modelo convirja mais rápido e consiga reproduzir o padrão existente. Particularmente em dados de mobilidade urbana, espera-se que exista horários de pico para a base de dados em questão, por exemplo, horários em que as pessoas estão indo ou voltando do trabalho. Dessa forma, não era esperado que os modelos modelassem corretamente os dados da Rua II, pois ela não possui padrões claros de comportamentos (tanto sazonalidades ou tendências).

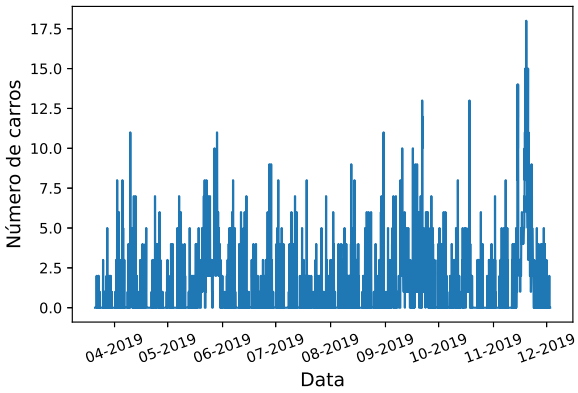
A Figura 28 mostra a visualização das séries que formam a base Ruas Multivariadas que é composta pelas Ruas IV, V, VI, VII e VIII, entre Março e Dezembro de 2019. Na Figura, percebemos que apenas as Ruas IV (em azul) e VI (em vermelho) apresentam uma



(a) Rua I



(b) Rua II



(c) Rua III

Figura 27 – Visualização do número de carros observados para a Ruas I (a), Rua II (b) e Rua III (b). As Ruas I (a) e III (b) aparentem possuir um número de observações de carros que se repetem ao longo do tempo, com picos e vales. Esse comportamento não é tão evidento quanto o observado pela base *BikeSharing*. Os dados na Rua II não seguem nenhum comportamento aparente, com observações de dados que variam bruscamente entre 0 e 4.

tendência de crescimento dos dados, aproximadamente em dezembro. Esse comportamento é mais evidente na Rua VI.

Em relação à sazonalidade, todas as ruas aparentam ter uma certa repetição nos dados, com alguns picos de observação se destacando. Por exemplo, é interessante notar que há um pico de observações que ocorre no mês de setembro, na Rua VII e VIII. A Rua V (em verde), entretanto, apresenta uma repetição esparsa dos dados, com praticamente nenhuma observação até Junho e, novamente, nenhuma observação do final de Julho até Outubro.

5.2.3 Variabilidade

A Figura 29 mostra, por meio de histogramas, a variabilidade das bases Rua I, II e III. A Rua I (Figura 29a) apresenta a maior variabilidade, com dados variando entre 0 e 30 e com a maioria das observações sendo menor que 5. Em seguida, na Figura 29c,

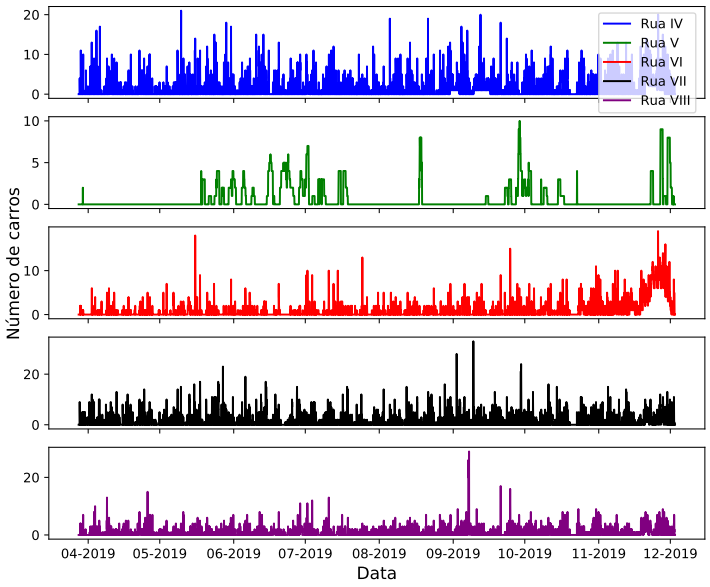


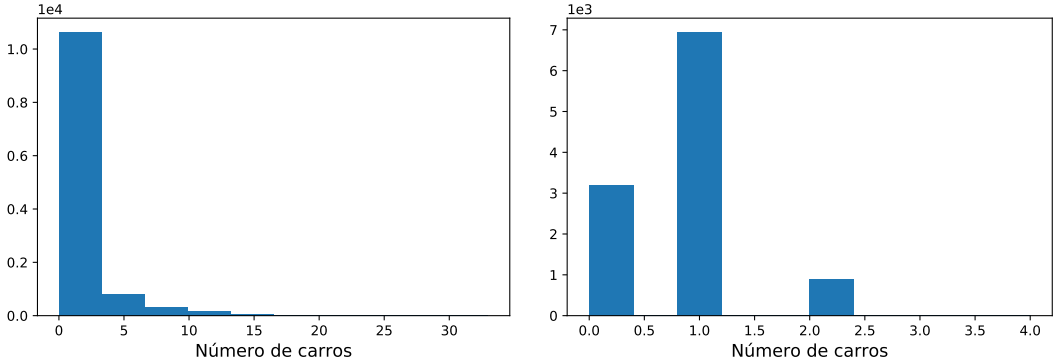
Figura 28 – Número de carros nas Ruas IV, V, VI, VII, VIII entre Março e Dezembro de 2019. As ruas não apresenta um comportamento que se destaca, exceto pela Rua V, que apresenta um número esparso de carros durante algumas datas. As Ruas IV, V e VI, logo no final das observações, apresentam uma leve tendência de crescimento.

tem-se a Rua III, com uma variabilidade similar à Rua I, mas variando entre 0 e 12 e com maiores frequências de observações sendo menores que 2.

Em contrapartida, a base Rua II (Figura 29b) possui a menor variabilidade, com apenas 4 valores de carros nas ruas (0, 1, 2 e 4). Pela pouca variabilidade, talvez fosse a base mais fácil de modelar. Porém, como visto anteriormente, os dados dessa base não possuem um comportamento que seja aprendível e, por isso, os modelos não conseguem modelá-la corretamente.

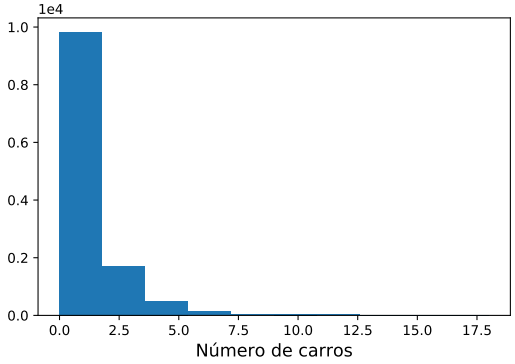
Os histogramas para a base Ruas IV_VIII podem ser vistos na Figura 30. Nessa base, a rua com a maior variabilidade é a Rua VII, com um número maior de diferentes valores para os dados (25 valores únicos), que variam de 0 à 33. Em seguida a Rua VIII, com 23 valores únicos para os dados, variando entre 0 e 29. A Rua IV possui 22 valores únicos, de 0 à 21. A quarta rua com maior variabilidade é a Rua VI, com 19 valores diferentes de 0 à 19. Por fim, a Rua V, com apenas 11 valores diferentes, que variam entre 0 e 10.

Como pode ser visto na Figura 30, exceto pela Rua V, as 5 ruas da Ruas IV_VIII apresentam variabilidades similares. Esse último caso era esperado, pelo que foi constatado na análise de sazonalidade e tendência para a Rua V, na Figura 28, com a maioria das observações para essa rua sendo 0.



(a) Rua I

(b) Rua II



(c) Rua III

Figura 29 – Histograma para os números de carros na ruas, para bases Rua I (a), Rua II (b) e Rua III (c). Em todas as bases, a maioria das observações fica entre 0 e 1. Nota-se também que a Rua II (b) apresenta a menor variabilidade.

5.2.4 Soma por Intervalo

As somas por intervalo das bases Rua I, II e III podem ser vistas na Figura 31. A Rua I, na Figura 31a, apresenta uma diferença clara entre os dias úteis e finais de semana. Nos dias úteis, o pico do número de carros na rua ocorre por volta do intervalo 40, ou seja, por volta das 20hs, com picos que variam de ≈ 250 à ≈ 350. Nos finais de semana, apenas o sábado (linha marrom) apresenta pico por volta do intervalo 26, ou seja, às 13hs, chegando próximo de 100 carros na rua. No domingo, contudo, o número de carros é inferior à 50.

Na Rua II (Figura 31b) não é possível verificar nenhum comportamento claro nos dados, como distinção evidente entre dias úteis e finais de semana. Como pode ser visto, os dados, em todos os dias da semana, variam em um intervalo relativamente pequeno (de 23 à 29 carros na rua), se comparado às bases Rua I e III.

Por fim, a Rua III, na Figura 31c, apresenta dados com padrões bem similares à base *BikeSharing*, tanto nos picos nos períodos da manhã (no intervalo 22) e à tarde (no intervalo 42) dos dias úteis, quanto na distinção entre dias úteis e finais de semana. Percebe-se pela figura que, durante os dias úteis, os dados possuem picos que variam entre

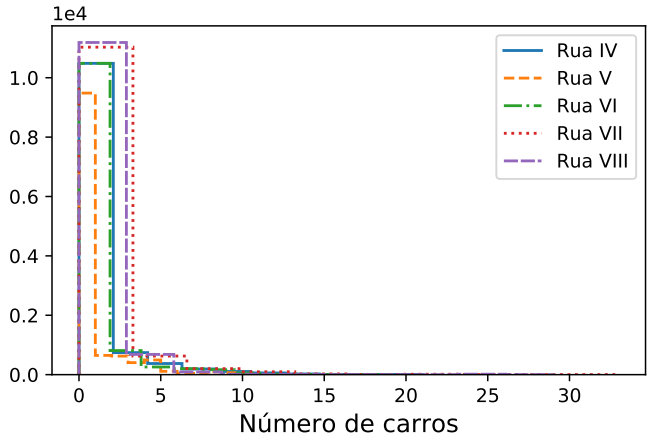
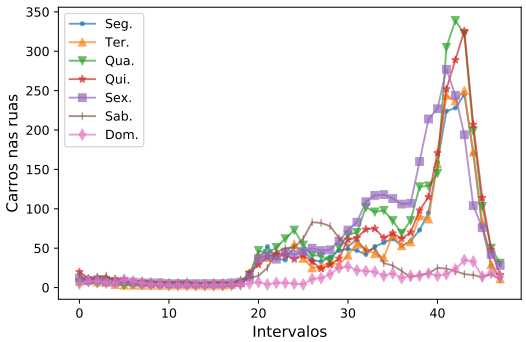
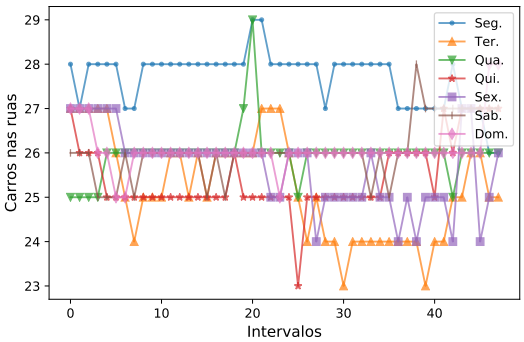


Figura 30 – Histograma das Ruas IV, V, VI, VII e VIII. Nota-se que os histogramas são muito parecidos, com maiores frequências de poucos carros nas ruas e com a Rua VII apresentando a maior variabilidade.

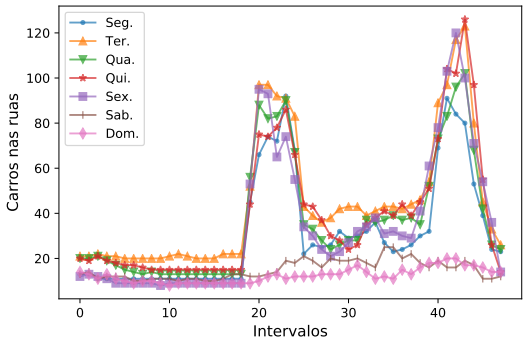
≈ 70 e ≈ 100 (intervalo 22) e entre ≈ 90 e 130 (intervalo 42). Nos finais de semana, assim como ocorre na Rua II, os dados não apresentam nenhum padrão claro e variam entre valores inferiores à 20 carros na rua.



(a) Rua I



(b) Rua II

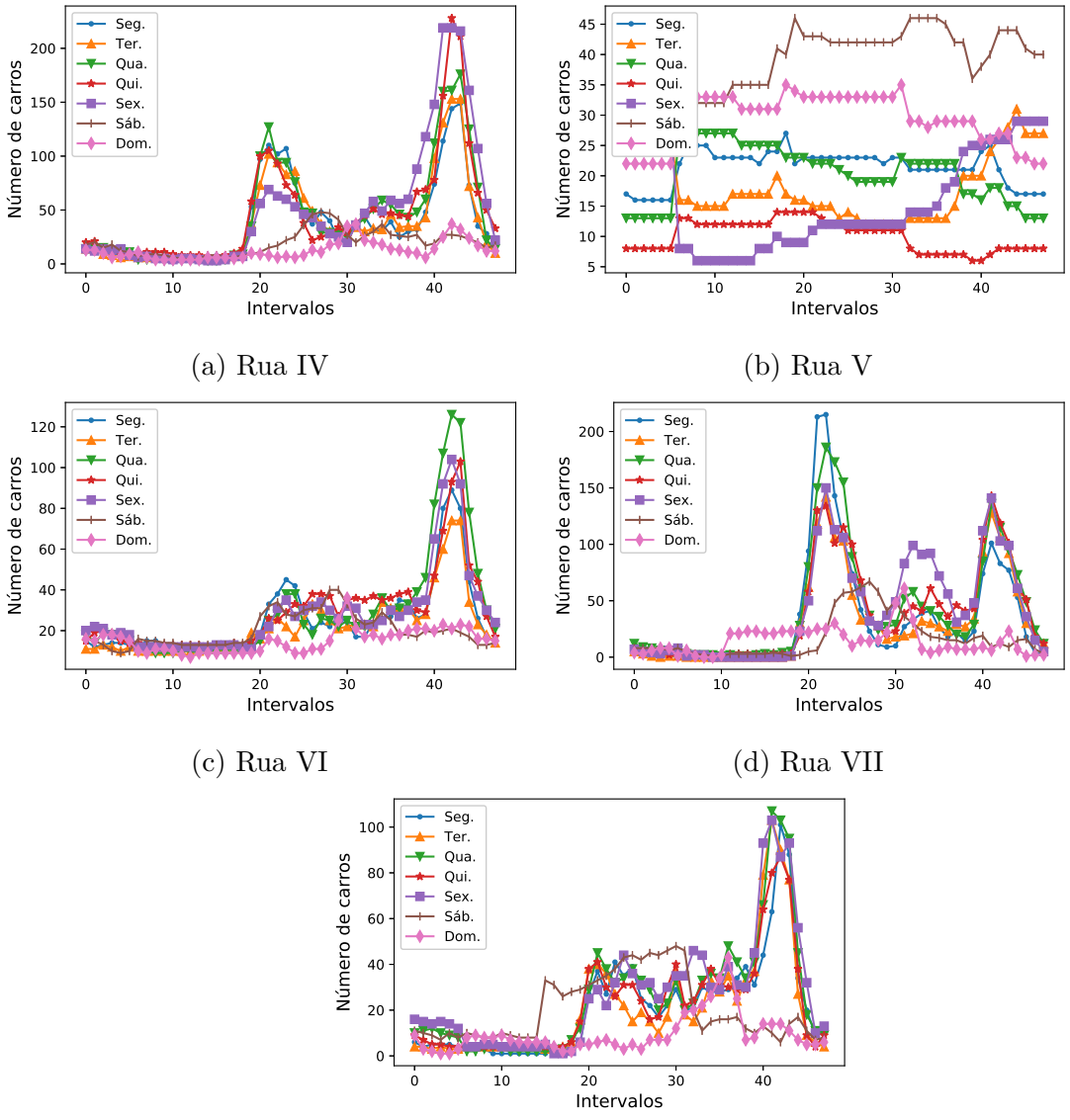


(c) Rua III

Figura 31 – Soma por intervalo para as bases Rua I (a), Rua II (b) e Rua III (c). Exceto pela Rua II (b), é possível identificar padrões de carros nas bases analisadas

A Figura 32 apresenta as somas por intervalo para as Ruas IV, V, VI, VII e VIII.

Nas figuras as somas consideraram os 7 dias da semana. Dentre as 5 ruas, apenas na Rua V (Figura 32b) não é possível identificar uma tendência dos dados que se destaque. Novamente, essa foi a rua sem uma sazonalidade e tendência aparente e com a menor variabilidade. Os dados nas Ruas VI (Figura 32c) e VIII (Figura 32e) comportam-se de maneira similar, com uma maior observação dos dados do início da tarde até à noite (entre os intervalos 35 e 40). Em ambas as ruas isso ocorre apenas durante os dias úteis.



(a) Rua IV

(b) Rua V

(c) Rua VI

(d) Rua VII

(e) Rua VIII

Figura 32 – Somas por intervalo para as Ruas IV (a), V (b), VI (c), VII (d) e VIII (e), considerando todos os dias da semana. No eixo X, os intervalos representam intervalos de 30 minutos presentes em um dia. Exceto pela Rua V (b), em todas as ruas os dados nos dias da semana e dias úteis são diferentes. Picos nos dados podem ser observados no início da noite, nas Ruas VI (c) e VIII (e), e durante a manhã e início da noite, nas Ruas IV (a) e VII (d).

Por fim, as Ruas IV (Figura 32a) e VII (Figura 32d) apresentam observações dos dados similares à base *BikeSharing*, com dois picos principais: um durante o período da

manhã, que se inicia por volta do intervalo 16, e outro durante à o final tarde e início da noite, que tem início aproximadamente no intervalo 36. Em ambas as ruas a diferença de observações entre dias úteis e finais de semana também é evidente.

5.3 Considerações sobre o Capítulo

Neste capítulo apresentou-se as bases de dados utilizadas nos experimentos e suas caracterizações, bem como descreveu-se o ambiente de execução e pré-processamento que foi aplicado aos dados. Entender o comportamento dos dados é fundamental para se analisar quais características os modelos precisam conseguir aprender corretamente podem auxiliar nos ajustes que serão feitos aos parâmetros dos modelos.

6 Avaliação do MobDeep

O MobDeep foi avaliado com as bases de dados *BikeSharing* e *Trânsito Vitória* e *Ruas Multivariadas*. Neste capítulo são apresentados os resultados da avaliação dos modelos utilizados. A Seção 6.1 descreve a metodologia utilizada nos experimentos. A Seção 6.2 apresenta como as métricas foram utilizadas para avaliação dos modelos treinados. A Seção 6.3 descreve os parâmetros definidos para os modelos. A Seção 6.4 apresenta as avaliações quantitativas e qualitativas dos modelos. As avaliações quantitativas foram feitas com 1000 bases sintéticas para cada base real usada e as avaliações qualitativas usaram a primeira base sintética gerada.

6.1 Experimentos

Esta seção apresenta o ambiente de execução usado e o pré-processamento aplicado aos dados. Em relação ao ambiente de execução, os modelos baseados em aprendizado profundo necessitam de uma atenção especial, pois normalmente exigem mais recursos computacionais. O pré-processamento dos dados irá variar com relação ao modelo que será utilizado e às características de cada base de dados. Por fim, o ARIMA é aplicado apenas nas bases com uma única variável.

6.1.1 Ambiente de execução

Todos os experimentos foram executados na plataforma *Google Colaboratory*[1]. Essa é uma plataforma que oferece uma máquina virtual Linux com 12 GB de memória RAM, com 107 GB de armazenamento. Além disso, a plataforma possibilita que os códigos sejam executados tanto em GPU (Tesla T4), por um período limitado de horas que pode variar dependendo do uso, ou CPU.

Os modelos ARIMA são os mais fáceis de treinar, não exigindo tantos recursos computacionais como os modelos baseados em aprendizado profundo. Em relação às GANs, os modelos podem ser encontrados em seus respectivos repositórios. Os modelos C-RNN-GAN[2] e RGAN[3], foram os primeiros à serem desenvolvidos e já não recebem mais atualizações. O código da TimeGAN[4] utilizado neste trabalho é uma implementação mais recente do código original[5] e, atualmente, ainda recebe melhorias e novas funcionalidades.

[1] https://colab.research.google.com/
[2] https://github.com/olofmogren/c-rnn-gan
[3] https://github.com/ratschlab/RGAN
[4] https://github.com/ydataai/ydata-synthetic
[5] https://github.com/jsyoon0823/TimeGAN

Nesse sentido, algumas alterações mínimas nos códigos da C-RNN-GAN, RGAN e TimeGAN foram necessárias. Para os três modelos, que foram usados no arcabouço, adicionamos códigos que adaptaram o salvamento de modelos e dados gerados para a estrutura do arcabouço. Como a C-RNN-GAN foi criada para modelar arquivos no formato *.midi*, a alteração mais crucial foi adaptar o código para selecionar as amostras (*batchs*) dos dados reais usados. Por fim, a RGAN em sua implementação original, possui métricas que causavam erros durante a realização dos experimentos. Por ser uma métrica específica para alguns dos problemas abordados no trabalho original, removemos, sem risco de influenciar nos resultados, as partes do código faziam referência à essa métrica.

A maioria dos modelos GANs exigem algumas especificidades para serem treinadas. A RGAN e C-RNN-GAN precisam ser executadas na versão 1.5 do TensorFlow[6] e, especificamente a C-RNN-GAN necessita que os códigos sejam executados na versão 2.7 do Python. Além disso, a TimeGAN possui alguns requisitos de pacotes que precisam ser pré-instalados no ambiente de execução.

6.1.2 Pré-processamento dos dados

O principal pré-processamento aplicado aos dados foi inserir valores em datas ou horários onde não houve contabilização de dados. Nesse caso, tanto para as bases *BikeSharing* quanto para a base Trânsito Vitória, inserimos o valor 0 onde não havia dados contabilizados. O objetivo dos modelos geradores é conseguir capturar as características principais das ruas. Sendo assim, se uma rua ficou vários dias sem contabilização do número de carros, essa característica deveria ser aprendida pelos modelos. Nas próximas seções serão descritos os processamentos específicos para os modelos estocásticos (ARIMA) e baseados em aprendizado profundo (modelos GANs).

6.1.2.1 ARIMA

O ARIMA é treinado em dados estacionários, então utilizamos biblioteca pmdarima[7] do Python para identificar automática se alguma diferenciação seria necessária para tornar as séries estacionárias. Tanto para as bases *BikeSharing* quanto para as Ruas I, II e III, a biblioteca informou que nenhuma diferenciação seria necessária. Ainda assim, a base *BikeSharing*, que possui valores que variam entre 0 e 2037 e, para facilitar o treinamento, aplicamos a função *log* sobre essa base. Isso diminui a diferença de escala entre os dados e permite que os dados gerados possam facilmente ser transformados para a escala dos dados reais.

Finalmente, os dados de entrada para um modelo ARIMA, precisam ser no formato de um vetor. Caso a base esteja em organizada com multi-dimensões, por exemplo, (10,24,1),

[6] https://www.tensorflow.org/
[7] https://alkaline-ml.com/pmdarima/

precisa ser transformado em um vetor com 240 registros, ou seja, 10*24*1. Dessa forma, as bases BikeSharing, Ruas I, II e III, são, respectivamente, um vetor com 78888, 12000, 11040 e 12336 registros.

6.1.2.2 Modelos baseados em GANs

Para os modelos de GANs utilizados, o principal pré-processamento realizado foi a normalização dos dados. A normalização consiste garantir que os dados variem em um intervalo pequeno, geralmente $[-1, 1]$ ou $[0, 0]$ e é importante em dados, como a *BikeSharing*, onde os valores variam de algumas dezenas até milhares. Para a normalização dos dados usamos a transformação MinMax Scaler, da biblioteca *scikit-learn*.

Além disso, as bases precisam estar organizadas em um vetor com três dimensões: n amostras, t intervalos de tempo e m variáveis (n, t, m). Assim, as dimensões para as bases *BikeSharing*, Rua I, Rua II, Rua III e Ruas Multivariadas são, respectivamente, (3287, 24, 1), (250, 48, 1), (230, 48, 1), (257, 48, 1) e (250, 48, 5).

Especificamente para a utilização do RGAN, foi preciso que criássemos um dicionário com a estrutura: {'**samples**':{'train':[ ], 'test':[ ], 'vali':[ ]}, '**labels**':{'train':[ ], 'test':[ ], 'vali':[ ]}}, onde *'samples'* contém aos dados divididos em treino, teste e validação e *'labels'* contém as classes dos dados (caso possuam alguma).

6.2 Métricas para Avaliação

Os modelos foram avaliados quantitativamente por meio da análise dos resíduos dos modelos e qualitativamente, por meio da visualização da soma por intervalo dos dados. A especificidade de cada avaliação será apresentada à seguir.

6.2.0.1 Analises Quantitativas

Para as análises quantitativas, como explicado sobre o funcionamento do MobDeep, o obtenção dos resíduos ocorre de duas maneiras. No ARIMA, os melhores modelos ajustados aos dados reais foram usados para geração dos dados sintéticos e, posteriormente, o mesmo modelo (com os mesmos parâmetros) foi ajustado à cada uma das bases sintéticas. Após cada ajuste, o resíduo para a respectiva base sintética é obtido.

Para as GANs usamos uma rede com 2 LSTM, cada uma com 200 unidades. Para a base *BikeSharing*, em que os dados estavam organizados por cada hora do dia, a entrada do modelo foi definida com X samples, 24 intervalos de tempo e n variáveis. Para as bases Rua I, Rua II e Rua III, os dados estão organizados em à cada 30 minutos do dia e a entrada do modelo foi definida como X samples, 48 intervalos de tempo, 1 variável. Por fim, a as bases de Rua IV à VIII também são organizadas por cada 30 minutos de um dia,

mas possuem 5 variáveis (ruas), logo, a entrada do modelo será X samples, 48 intervalos de tempo e 5 variáveis.

6.2.0.2 Analises Qualitativas

O MobDeep gera análises visuais comparando os dados sintéticos com os reais. Denominamos essa análise visual de soma por intervalo que, como explicado anteriormente, consiste em se agrupar os dados por cada dia da semana e somar os valores com base na data e horário de observação dos dados. Por exemplo, para a base *BikeSharing* os dados são agrupados nos 7 dias da semana, e os valores correspondente à cada hora de cada dia são somados. Para as Ruas de I à VIII a diferença é que os dados estão organizados de 30 em 30 minutos. Para facilitar a visualização apresentamos os resultados correspondentes à apenas à Segunda, Quarta, Sexta, e Sábado de cada base de dados.

6.3 Hiperparâmetros

Nos experimentos feitos, buscamos identificar quais parâmetros que apresentavam os melhores resultados para os modelos avaliados. Para o ARIMA, como todas a bases usadas eram estacionárias, definiu-se que $d = 0$ e p e q iriam variar de 0 e 3. Por fim, utilizamos a biblioteca *pmdarima* que automaticamente identifica a combinação de parâmetros que melhor se ajustam aos dados. Nesse caso, variamos os parâmetros p e q entre os intervalos 0 e 3. Os melhores parâmetros para cada base de dados podem ser vistos na Tabela 11.

Tabela 11 – Parâmetros p, d, q do ARIMA para cada uma das bases de dados. Como as bases são estacionárias, o parâmetro d será sempre 0.

	BikeSharing	**Rua I**	**Rua II**	**Rua III**
p	2	2	3	2
d	0	0	0	0
q	3	1	3	3

Para as GANs, verificamos os parâmetros que mais influenciavam o treinamento, definimos valores distintos para cada um deles, treinamos os modelos nos diferentes parâmetros e avaliamos os resultados. Após uma analise visual dos dados, como discutido na No capítulo anterior, definimo alguns parâmetros antes do treinamento. Por exemplo, para a base de dados *BikeSharing*, percebemos que os padrões se repetem aproximadamente à cada 24 horas e, portanto, um tamanho razoável para cada lote de treinamento poderia ser de aproximadamente 1 mês (cerca de 28 dias). No referido capítulo, as sub-bases Rua I, II e III, não apresentaram padrões tão evidentes e não foi possível utilizar a mesma lógica. Nesse caso, variamos o tamanho do lote de treinamento entre 25, 50, 75, e 100 e avaliou-se desempenho em cada caso.

Tabela 12 – Parâmetros para a base de dados *BikeSharing*, com o *batch size* (tamanho do lote), *learning rate* (taxa de aprendizagem), *hidden dimensions* (dimensões escondidas) e *epoch* (épocas de treinamento).

	batch size	**learning rate**	**hidden dimensions**	**epoch**
C-RNN-GAN	28	.0001	100	50
RGAN	28	.1	25	3000
TimeGAN	28	.0005	24	5000

Tabela 13 – Parâmetros para as bases Rua I, II e III, com o *batch size* (tamanho do lote), *learning rate* (taxa de aprendizagem) , *hidden dimensions* (dimensões escondidas) e *epochs* (épocas de treinamento).

	batch size	**learning rate**	**hidden dimensions**	**epoch**
C-RNN-GAN	50	.0001	100	300
RGAN	50	.1	32	3000
TimeGAN	50	.0005	72	4000

Tabela 14 – Parâmetros para a base Ruas IV_VIII com o *batch size* (tamanho do lote), *learning rate* (taxa de aprendizagem) e *hidden dimensions* (dimensões escondidas) e *epochs* (épocas de treinamento).

	batch size	**learning rate**	**hidden dimensions**	**epoch**
C-RNN-GAN	50	.0001	28	200
RGAN	50	.1	64	3000
TimeGAN	50	.0005	72	3000

As Tabelas 12, 13 e 14 apresentam os parâmetros com melhor desempenho para as bases de dados *BikeSharing*, Ruas I, II e III e Ruas IV_VIII. Os diferentes parâmetros testados para cada modelo podem ser vistos no Apêndice B. Como cada modelo possui um número grande de parâmetros, optamos por exibir apenas aqueles que mais influenciam o desempenho do treinamento e que estavam presentes em todos os modelos. Ressalta-se que o parâmetro *epochs* indica o número aproximado de iterações em que foi possível obter bons resultados do modelo sendo treinado. Em alguns casos os modelos apresentavam bons resultados algumas épocas antes do valor apresentado.

O comportamento das GANs, considerando as modificações feitas nos parâmetros, será muito similar ao comportamento de uma Rede Neural, que são a base de um modelo GAN. Por exemplo, dados com relações muito complexas podem exigir que os modelos possuam uma maior capacidade de aprendizagem, ou seja, valores maiores para o parâmetro *hidden dimensions*. Além disso, diminuir o valor do parâmetro *learning rate* faz com que os modelos demorem mais para atualizar seus pesos e, consequentemente, demorem mais para gerar dados de qualidade. Por outro lado, um *learning rate* muito alto pode causar sobreajuste (do Inglês, *overfitting*), diminuindo a variabilidade dos dados gerados. Similarmente, um *batch size* muito grande reduz a capacidade de generalização dos modelos (KESKAR

et al., 2016). No caso das GANs isso pode resultar em uma geração de dados com pouca variabilidade, ou seja, os dados sintéticos irão ser praticamente os mesmos.

Finalmente, vale ressaltar que o ambiente de execução em que o arcabouço será executado possui uma forte influência sobre o tempo de treinamento e de geração dos dados sintéticos. De preferência, a execução deve ser feita em uma máquina com acesso à uma boa GPU e com um tamanho razoável de memória RAM. A Tabela 15 mostra os tempos de execução, em segundos, para o treinamento dos modelos e geração dos dados sintéticos, considerando o ambiente de execução utilizado. Os tempos mostrados são referentes à execução de uma época de cada modelo, exceto pelo ARIMA, que é executado uma única vez. A tabela mostra que a TimeGAN é um dos modelos mais rápidos durante o treinamento, mas é o modelo que leva mais tempo para geração dos dados.

Tabela 15 – Tempos aproximados de execução, em segundos, para o treinamento (durante uma época) e geração de uma base sintética de cada um dos modelos utilizados. O ARIMA não pode ser treinado numa base multivariada, por isso não tempos de execução para a base Ruas IV_VIII.

Etapa	**Base de dados**	**ARIMA**	**C-RNN-GAN**	**RGAN**	**TimeGAN**
Treinamento	Ruas I, II, III	4.77	1.5	.257	.273
	BikeSharing	80	9.5	1.64	1.74
	Ruas IV_VIII	-	6.84	1.25	1.325
Geração	Ruas I, II, III	.048	.587	.056	1.48
	BikeSharing	3	3.74	.364	9.5
	Ruas IV_VIII	-	2.85	.277	7.2

6.4 Resultados

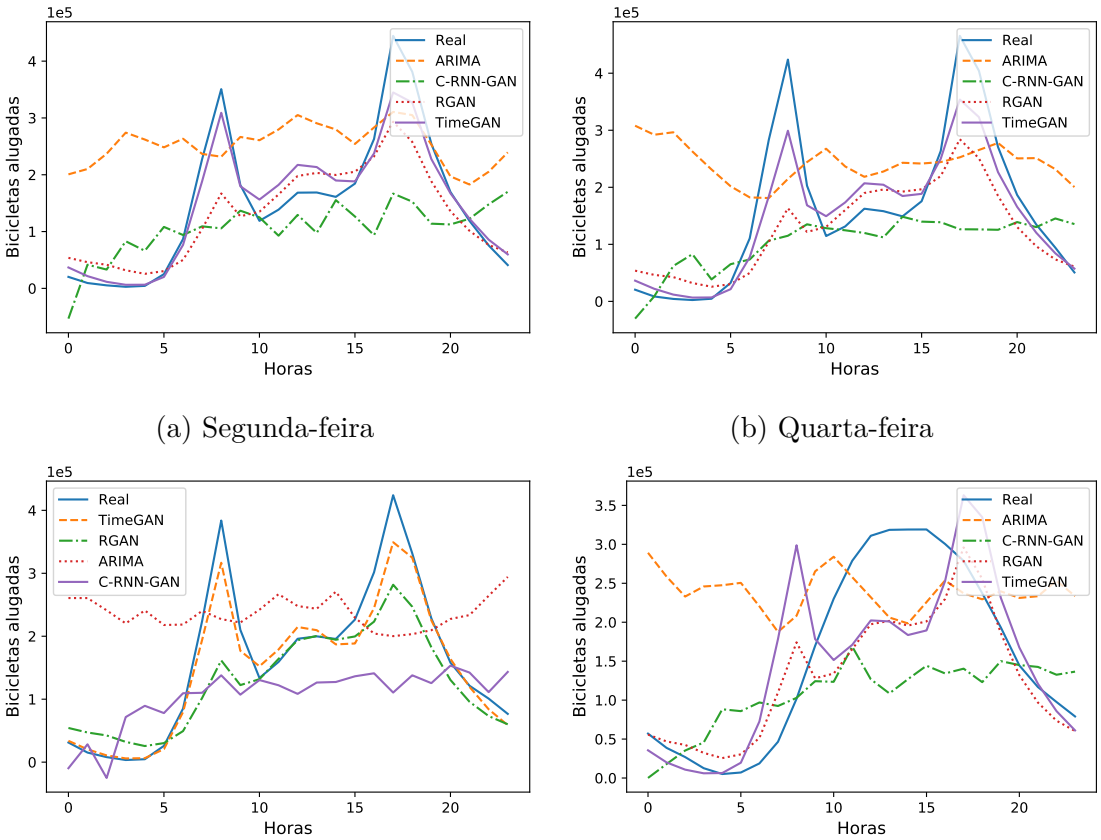
Os resultados obtidos das avaliações dos modelos treinados com o MobDeep podem ser vistos à seguir. Inicialmente são apresentados os resultados das avaliações qualitativas, com as comparações entre os dados sintéticos e reais. Em seguida discutimos a avaliação quantitativa, exibindo os resíduos dos modelos utilizados. Nas seções à seguir serão apresentadas as respectivas análises para as bases *BikeSharing* e para as Ruas de I à VIII, selecionadas da base Trânsito Vitória.

6.4.1 Análises Qualitativas

Nas análises qualitativas apresentamos os resultados das somas por intervalo das bases BikeSharing e Ruas I, II e III. As somas por intervalo das 5 ruas da base Ruas IV_VIII no Apêndice A.

6.4.1.1 *BikeSharing*

A Figura 33 apresenta as somas por intervalo da base *BikeSharing* considerando a Segunda-feira, Quarta-feira, Sexta-feira e Sábado, paras as 24 horas de cada dia. As Figuras 33a, 33b e 33c mostram que a TimeGAN foi o modelo que gerou dados com propriedades que mais se aproximam dos reais, com picos de bicicletas alugadas por volta das 8 e 17 horas e com valores menores entre as 10 e 15 horas. Similarmente, a RGAN captura levemente os padrões da base *BikeSharing*, para os três dias úteis, com picos por volta das 8 e 17 horas. Para esses dois modelos, as somas por intervalo dos dados sintéticos dos três dias são muito parecidas. Como visto anteriormente, entretanto, na base real esse comportamento também ocorria.



(a) Segunda-feira

(b) Quarta-feira

(c) Sexta-feira

(d) Sábado

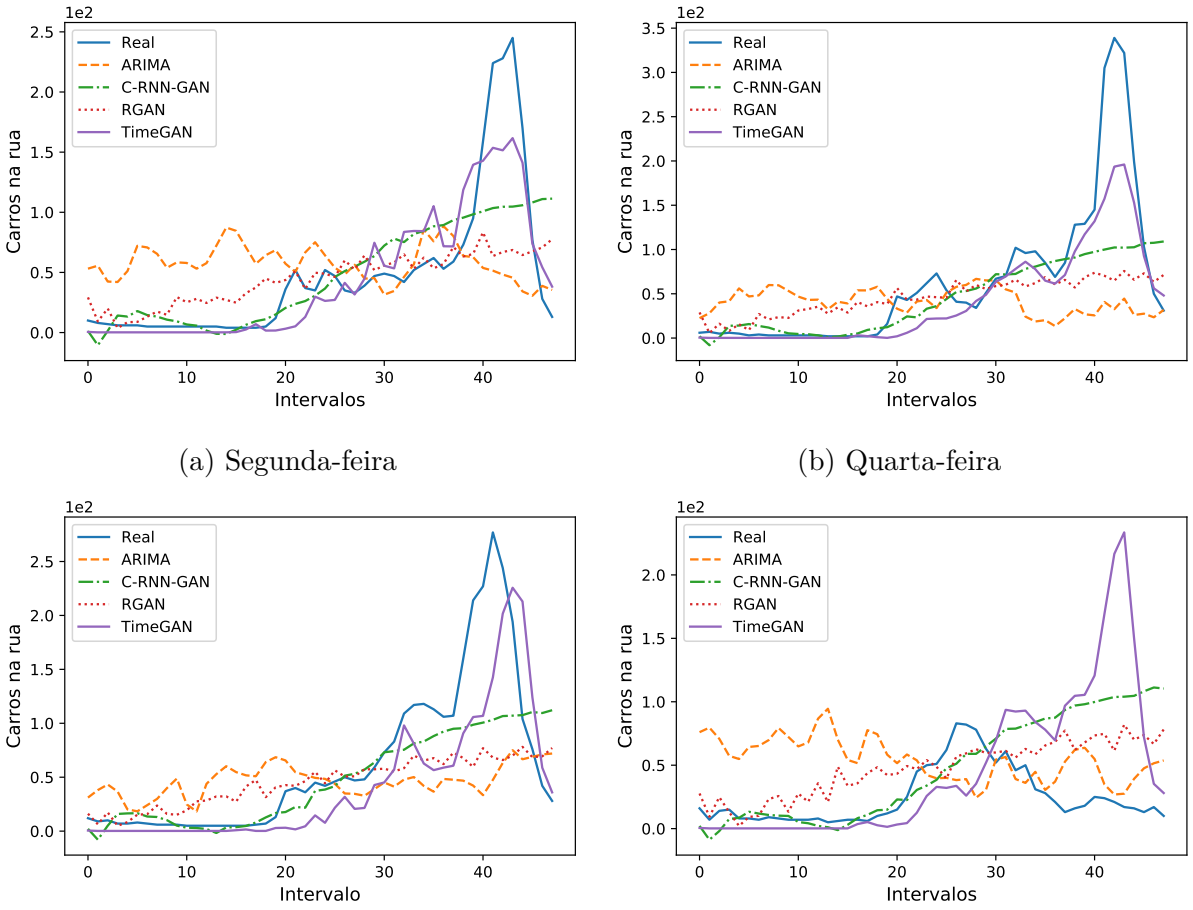
Figura 33 – Somas por intervalo dos dados sintéticos gerados comparados à base de dados *BikeSharing* para a Segunda (a), Quarta (b), Sexta (c) e Sábado (d). O TimeGAN foi o modelo que gerou dados cujas somas mais se aproximaram dos dados reais. No Sábado (d) nenhum modelo conseguiu modelar o comportamento dos dados.

O ARIMA e a C-RNN-GAN não conseguiram capturar corretamente as características dos dados, em nenhum dos dias observados. Os dados do ARIMA, como pode ser visto, comportam-se de maneira aproximadamente linear, nos quatro dias, com nenhum padrão claro que se destaque. De maneira parecida, nos quatro dias, a C-RNN-GAN apresenta

um valor menor no de bicicletas alugadas no início dos dias com uma leve tendência de. Diferente dos outros modelos, a C-RNN-GAN apresentou valores negativos.

Como mostrado pela Figura 33d, nenhum dos quatro modelos conseguiu modelar os dados no Sábado. Diferente dos dias úteis, os dados dos finais de semana apresentam um pico de observação entre as 12 e 13 horas. Referente aos modelos RGAN e TimeGAN, o desempenho inferior nos finais de semana pode ser justificado pelo fato de que, em um ano, a maioria dos dias são dias úteis. Dessa forma, os modelos tendem a aprender mais sobre as características desses dias. Para o caso da *BikeSharing*, por exemplo, entre de 2011 à 2019, há um total de 78888 dias, dos quais apenas $\approx 28\%$ (22416) são finais de semana.

6.4.1.2 Trânsito Vitória



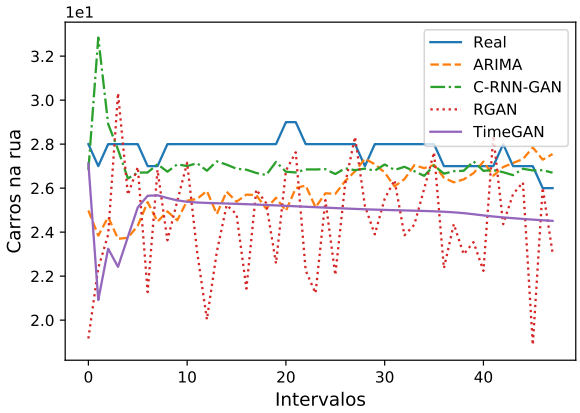
(a) Segunda-feira

(b) Quarta-feira

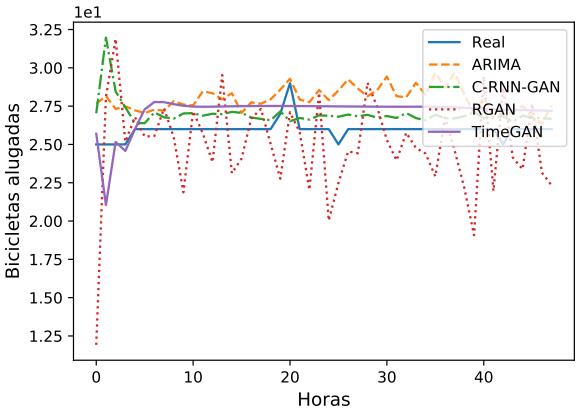
(c) Sexta-feira

(d) Sábado

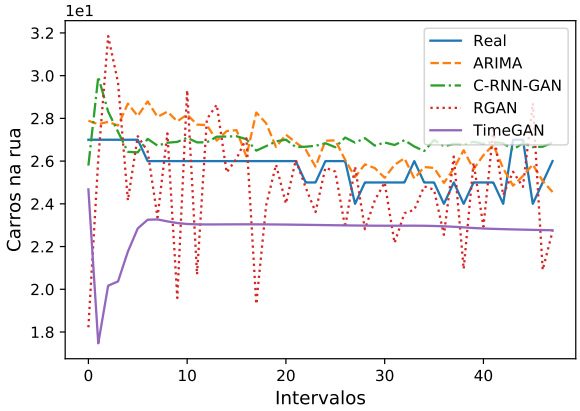
Figura 34 – Somas por intervalo dos dados sintéticos gerados comparados à base Rua I para a Segunda (a), Quarta (b), Sexta (c) e Sábado (d). A TimeGAN foi o modelo que gerou dados cujas somas mais se aproximaram dos dados reais. Assim como para a *BikeSharing* nenhum modelo conseguiu modelar o comportamento dos dados para o Sábado (d).



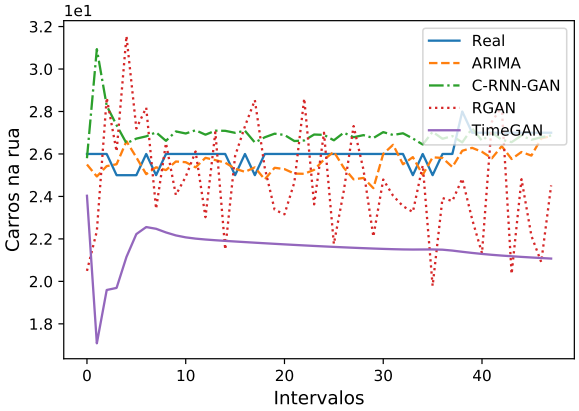
(a) Segunda-feira



(b) Quarta-feira



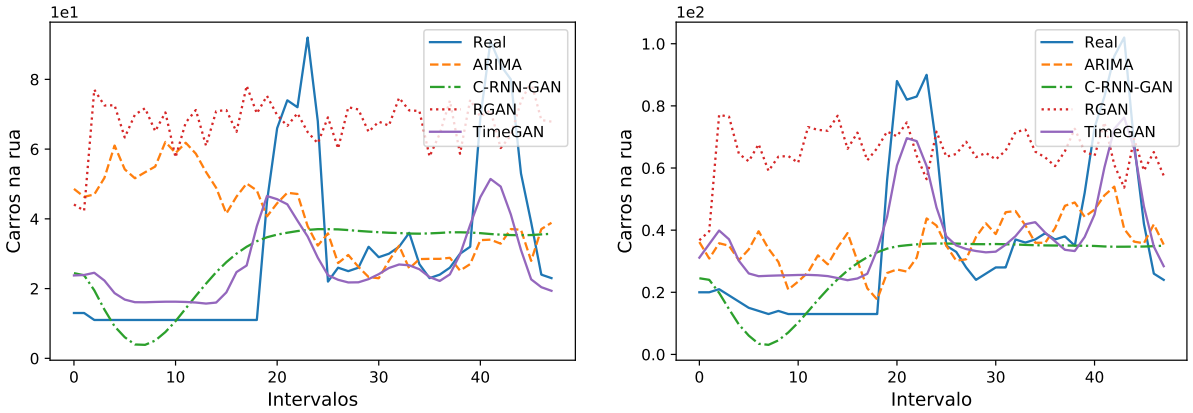
(c) Sexta-feira



(d) Sábado

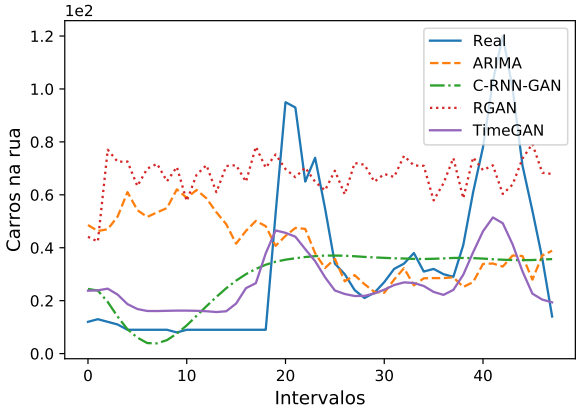
Figura 35 – Somas por intervalo dos modelos para a Rua II, considerando a Segunda (a), a Quarta (b), a Sexta (c) e o Sábado (d). A Rua II não apresenta um comportamento sazonal e possui uma variabilidade muito pequena. Os modelos que mais se aproximaram aos dados reais foram os que tendem a gerar dados também com pouca variabilidade, como a C-RNN-GAN e o ARIMA.

As somas por intervalo para a Rua I, para Segunda-feira, Quarta-feira, Sexta-feira e Sábado são mostradas na Figura 34. Assim como para *BikeSharing*, a TimeGAN foi o modelo que gerou dados sintéticos que mais se assemelham aos reais, capturando o comportamento do número de carros na rua durante a Segunda, Quarta e Sexta-feira, respectivamente, Figuras 34a, 34b, 34c. Pelas figuras, é possível notar que os dados se mantêm aproximadamente lineares antes do intervalo 20 e aumentam gradativamente até apresentarem um pico de carros, que se inicia próximo do intervalo 35. Nesse sentido, os dados gerados para as sextas-feiras foram o que mais se assemelharam aos reais, como mostra a Figura 34c. Nenhum dos outros modelos conseguiu reproduzir esse comportamento dos dados. O C-RNN-GAN, apresentou dados sintéticos com uma tendência de crescimento, mas sem o decrescimento após o intervalo 40. Assim como na *BikeSharing*, nenhum dos modelos aprendeu os comportamentos do dados para o Sábado, como pode ser visto na Figura 34d.

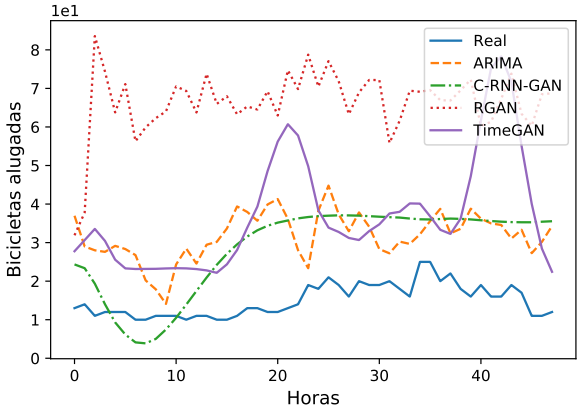


(a) Segunda-feira

(b) Quarta-feira



(c) Sexta-feira



(d) Sábado

Figura 36 – Somas por intervalo dos dados sintéticos gerados comparados à base de dados Rua III para a Segunda (a), a Quarta (b), a Sexta (c) e o Sábado (d). Essa base possui somas por intervalo muito similar à base *BikeSharing* e essa característica foi capturada pela TimeGAN. Assim como para bases anteriores, nenhum modelo capturou as propriedades dos dados durante o final de semana.

Como descrito no Capítulo anterior, a Rua II possui dados sem nenhum comportamento identificável e com baixíssima variabilidade, o que possivelmente prejudicaria o desempenho dos modelos. De fato, a Figura 35 evidencia que apenas o C-RNN-GAN gerou dados com características próximas às do reais, em todos os dias observados. Contudo, se olharmos os dados gerados pela C-RNN-GAN para as outras bases, veremos que, na maioria dos casos, esse modelo gerou dados com pouca variação entre os intevalos. Na Quarta-feira 35b, entretanto, a TimeGAN conseguiu gerar dados com propriedades similares aos reais. Em contrapartida, a RGAN gera dados com picos e vales que se alternam ao longo dos intervalos e o ARIMA apresenta dados levemente parecidos com os reais. Nesse último caso, assim como visto nas análises anteriores, isso pode ser justificado pelo fato de o ARIMA gerar dados com comportamentos aparentemente lineares.

As somas por intervalo da Rua III (Figura 36), mostram que, exceto pela TimeGAN, nenhum outro modelo conseguiu simular corretamente o comportamento dos dados na

Segunda, Quarta e Sexta-feira, que apresentam picos por volta do intervalo 20 e 40. A RGAN, assim como na Figura 35, gerou dados que oscilavam entre os intervalos, sem nenhum padrão aparente. A C-RNN-GAN acompanhou, levemente, o comportamento dos dados reais apenas entre os intervalos 0 e 20, onde o número de carros apresentam-se com os menores valores. Para os dados referentes ao Sábado, nenhum modelo conseguiu gerar dados com características semelhantes.

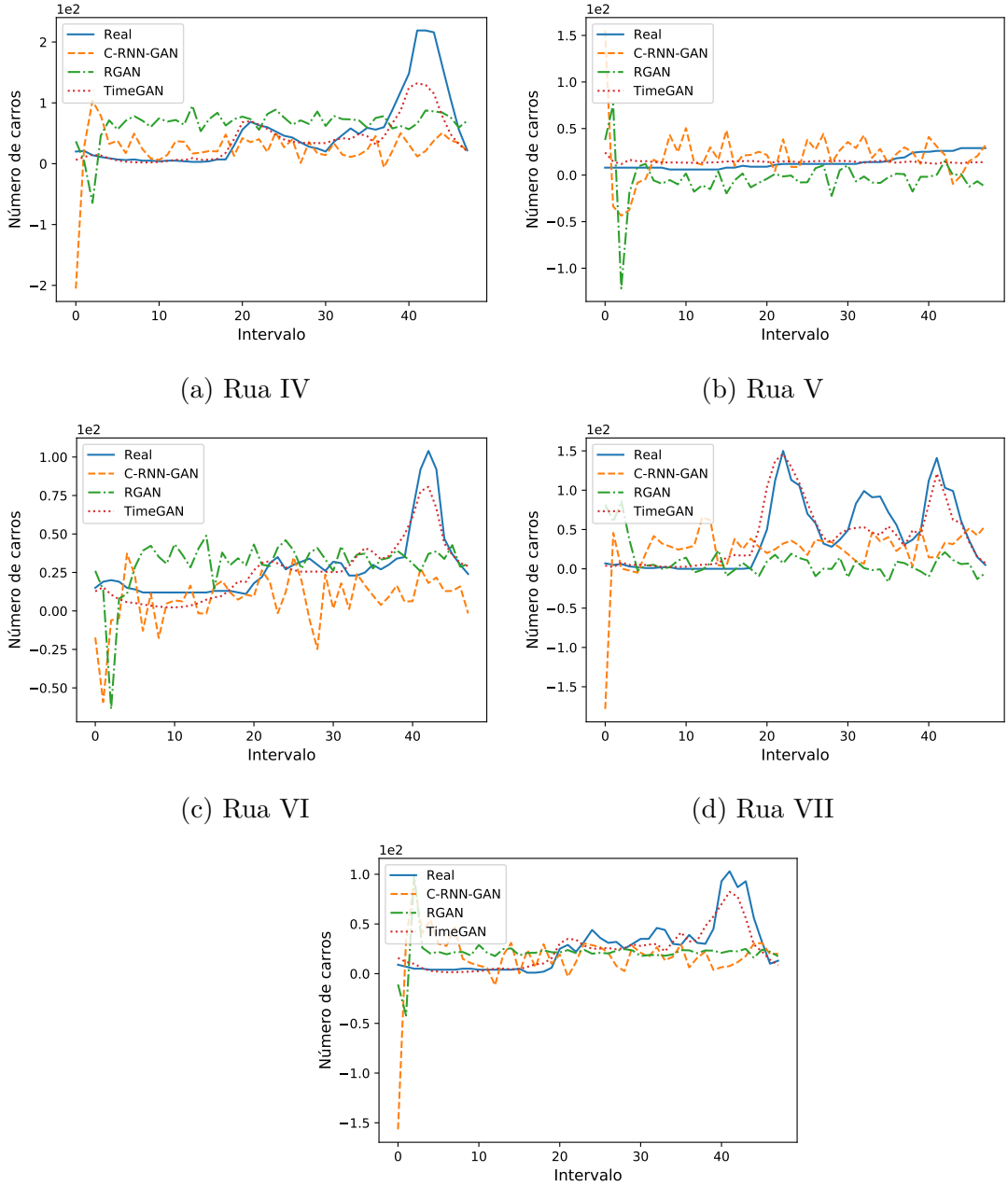
A Figura 37 apresenta as somas por intervalo para a base Ruas IV_VIII, para a Sexta-feira. Cada figura representa uma das 5 ruas presentes na base de dados. Dos modelos usados, apenas a TimeGAN conseguiu capturar bem as propriedades dos dados. Por exemplo, nas Ruas IV, VI e VIII, respectivamente, Figuras 37a, 37b e 37e, há um maior número de carros no final do dia, com um pico que se inicia pouco antes do intervalo 40. As respectivas figuras mostram que a TimeGAN modelou bem as oscilações dos dados, desde intervalo 0 até a tendência de crescimento apresentada. Para essas ruas, os outros modelos não apresentaram dados sintéticos com essas características.

Para a Rua VII, na Figura 37d, observamos que os dados reais apresentam três picos, por volta dos intervalos 20, 30 e 40. A figura mostra que a TimeGAN modelou os três picos observados, com uma melhor semelhança com os reais para os picos do intervalo 20 e 40. A Rua V, na Figura 37b, apresenta dados que comportam-se como uma reta, com um leve crescimento no número de observação no final do dia, por volta do intervalo 40. Para essa rua, a soma no intervalo 0 gerados pela TimeGAN são bem diferentes dos reais, com valores muito maiores que os apresentados pela soma dos dados reais.

Por fim, para todas as ruas, tanto a C-RNN-GAN quanto a RGAN, geraram somas com valores negativos, principalmente no início das Sextas-feiras. Apesar disso, a RGAN ainda teve um desempenho melhor que a C-RNN-GAN, com somas que oscilam menor. Esses resultados indicam que, evidentemente, a TimeGAN é o melhor modelo para a modelagem de bases multivariada. É importante ressaltar, além disso, que os parâmetros usados na TimeGAN para a base Rua IV_VIII foram os mesmo usados nas Ruas I, II e III, exceto pelo parâmetro *epochs* que, nesse caso, foi igual à 3000. Os outros modelos necessitaram que se fizesse uma variação dos parâmetros para obter-se modelos que gerassem dados com um mínimo de qualidade.

6.4.2 Análises Quantitativas

Os resíduos dos ajustes dos modelos podem ser vistos na Tabela 16, com médias à esquerda e desvios-padrão entre parênteses. Os melhores resultados são mostrados em negrito. Os resíduos de um modelo auxiliam na avaliação dos modelos, indicando se há informações nos dados que o modelo não conseguiu capturar, por exemplo, a influência que um dia de observação de dados pode ter no próximo dia, ou o comportamento dos dados em épocas diferentes. Em dados de mobilidade urbana, os dados podem variar



(a) Rua IV

(b) Rua V

(c) Rua VI

(d) Rua VII

(e) Rua VIII

Figura 37 – Somas por intervalo dos dados sintéticos gerados comparados as bases Ruas IV (a), V (b), VI (c), VII (d) e VIII (e) durante a Sexta-feira. Apenas a TimeGAN conseguiu reproduzir as propriedades dos dados reais. A C-RNN-GAN e a RGAN geraram dados negativos para todos as 5 ruas.

significativamente entre dias, mêses e anos. Mostramos anteriormente, por exemplo, que na base *BikeSharing* o número de bicicletas alugadas tende à ser menor no início e final de cada ano.

Para modelos puramente preditivos, as médias e desvios-padrão iguais a 0 significam que o modelo é ótimo, pois o que esperava-se que o modelo previsse (y), foi exatamente o que ele previu ($\hat{y}$), ou seja, $y - \hat{y} = 0$. Contudo, para o problema de geração de dados,

principalmente quando considera-se as questões de privacidade, médias e desvios-padrão iguais a 0 indicam que os dados sintéticos são praticamente uma cópia dos dados reais, ou seja, os dados gerados pelo modelo não possuem variabilidade e não são diferentes dos reais. Consequentemente, o modelo não provê nenhum tipo de privacidade aos dados. Assim, para o caso de geração de dados, espera-se resíduos com médias e desvios-padrão próximos de 0, porém não exatamente iguais.

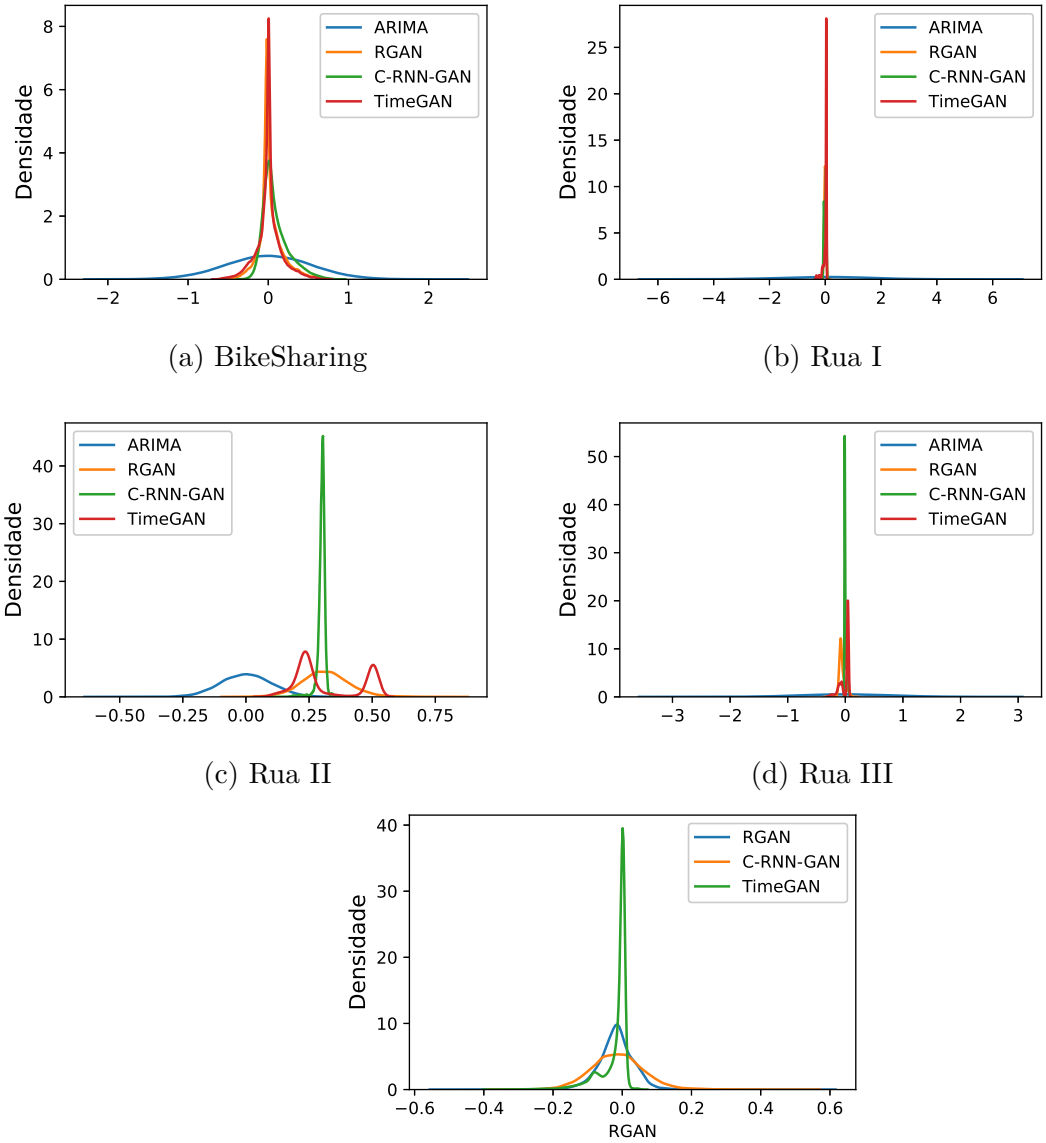
Tabela 16 – Análise dos resíduos dos modelos treinados, com as médias à esquerda e desvios padrão à direita. Os melhores resultados são mostrados em negrito.

	ARIMA	**TimeGAN**	**RGAN**	**C-RNN-GAN**
BikeSharing	.000 (.530)	**.007 (.158)**	.025 (.154)	.078 (.185)
Rua I	.000 (1.534)	.314 (.131)	**.005 (.032)**	.005 (.037)
Rua II	.000 (.100)	**.002 (.072)**	.316 (.094)	.300 (.018)
Rua III	.000 (.748)	.014 (.081)	.056 (.050)	**.001 (.019)**
Rua IV_VIII	-	**-.021 (.044)**	-.017 (.053)	-.013 (.078)

Dessa forma, a Tabela 16 demonstra que as GANs apresentaram os melhores resultados. A TimeGAN teve melhor desempenho nas bases *BikeSharing*, Rua II e Ruas IV_VIII. Na *BikeSharing* esse resultado era esperado, tendo em vista as somas por intervalo gerados pelos dados sintéticos da TimeGAN. O desvio padrão de 0.158, nesse caso, pode ser justificado pelo fato do modelo gerar, para os finais de semana, somas com propriedades similares aos dos dias úteis. A Rua II, como mostrado anteriormente, é uma base com pouca variabilidade e que não apresenta uma sazonalidade evidente. A soma por intervalo dessa base mostrou que a TimeGAN, apesar de não gerar dados tão próximos dos reais, gerou dados também com pouca variabilidade, justificando o seu melhor desempenho na análise dos resíduos.

A base Ruas IV_VIII possui propriedades interessantes, como, uma rua com somas por intervalo similar à soma das bases *BikeSharing* (Rua V), Ruas I, III (Ruas IV, VI e VIII) e uma rua (Rua V) com com somas parecidas com a Rua II. Como visto nas análises qualitativas anteriores, a TimeGAN gerou somas por intervalo bem similares aos reais, com uma qualidade bem superior aos outros modelos, para a base Ruas IV_VIII. Isso justifica o seu melhor desempenho nas análises dos resíduos.

Para a Rua I, a análise dos resíduos mostra que a RGAN foi o modelo que melhor capturou as propriedades dos dados. Contudo, nas análises qualitativas vimos que as somas por intervalo da TimeGAN são mais parecidas com as somas dos dados reais do que as somas geradas pela RGAN. Nesse sentido, a TimeGAN não ser o melhor modelo pode ser explicado por dois fatores principais: os dados gerados pela TimeGAN oscilam muito mais que os dados gerados pelos outros modelos, incluindo a RGAN (apesar de serem similares aos reais) e; os dados sintéticos da TimeGAN para os finais de semana são muito distintos dos dados reais, fazendo aumentar o valor da média dos erros e dos desvios padrão.



(a) BikeSharing

(b) Rua I

(c) Rua II

(d) Rua III

(e) Ruas IV_VIII

Figura 38 – Distribuição dos resíduos para as bases *BikeSharing* (a), Rua I (b), Rua II (c), Rua III (d) e Rua IV_VIII (e). No geral, as distribuições dos resíduos das GANs se assemelham mais uma distribuição normal do que o ARIMA, principalmente para a *BikeSharing*. Em todas as bases onde foi usado, o ARIMA apresentou um alto desvio padrão tanto para os resíduos quanto para as distribuições dos resíduos

A C-RNN-GAN obteve o melhor desempenho nos resíduos para a base Rua III. Assim como a situação anteriormente explicada, outros modelos, como a TimeGAN, possuem dados que, visualmente, se parecem mais com os reais. Novamente, os fatores decisivos para as análises dos resíduos são as oscilações dos modelos (a C-RNN-GAN gera dados mais suaves) e os dados que são gerados para os finais de semana. Nesse último caso o ARIMA, a RGAN e a TimeGAN geram dados que produzem somas com valores muito maiores que os dados reais.

Vale ressaltar também os resíduos gerados pelo ARIMA. Apesar das médias 0, a tabela mostra que os desvios padrão gerados pelo ARIMA para cada base são os maiores em comparação aos modelos de GANs, indicando que o modelo teve desempenho inferior aos outros, com relações temporais entre os dados que ele não conseguiu modelar. Isso corrobora os resultados mostrados nas avaliações qualitativas, em que o modelo ARIMA não conseguiu gerar dados com propriedades similares aos reais.

Considerando que um bom modelo precisa ter média e desvio padrão próximos de 0, era esperado que as distribuições dos resíduos, caso os modelos tivessem aprendido corretamente as propriedades dos dados, fossem aproximadamente normais. Para verificar essa situação, a Figura 38 apresenta as distribuições dos resíduos para cada modelo. Na base *BikeSharing*, na Figura 38a, notamos que apenas as GANs apresentam distribuições próximas das normais e com média 0. O ARIMA, apesar de ter média 0 possui um alto desvio padrão, como mostramos na Tabela 16.

Para as bases Ruas I, II e III, nas Figuras 38b, 38c e38d, respectivamente, notamos que as GANs apresenta distribuições próximas de uma distribuição normal apenas nas Rua I, e III. Para a Rua II, nenhum modelo de GAN apresentou uma distribuição com média 0. Entretanto, nas três Ruas a desvio padrão do ARIMA foi o maior apresentando, indicando que há propriedades nos dados que o modelo não conseguiu aprender.

Por fim, na base Ruas IV_VIII, como é uma base multivariada, não possui resíduos para o ARIMA. Pra essa base todos os modelos possuem distribuição com média 0, entretanto, a RGAN e a C-RNN-GAN possuem distribuições com maior desvio padrão. Como visto nas análises qualitativas, esses modelos não conseguiram gerar dados com propriedades similares aos reais.

Por fim, ressaltamos que, apesar dos resíduos serem uma métrica que pode ser feita automaticamente e fornece informações valiosas sobre o desempenho dos modelos, os resultados dos resíduos evidenciam que a avaliação qualitativa é importante para se entender como os dados sintéticos estão sendo gerados e se, de fato, condizem com o esperado.

6.5 Considerações Sobre o Capítulo

Neste capítulo apresentamos avaliação do MobDeep. Descrevemos as métricas utilizadas para a avaliação da qualidade dos dados sintéticos gerados, os hiper-parâmetros mais importantes de cada modelo e como eles podem influenciar o processo de treinamento e os resultados dos experimentos realizados.

Os resultados obtidos dos experimentos mostram que o arcabouço proposto consegue modelar com eficiência e variabilidade as propriedades das bases de dados. Neste

sentido, entre os modelos utilizados, a TimeGAN foi o que obteve os melhores resultados, qualitativamente e quantitativamente. Como o foco dos modelos experimentados é a geração de séries temporais, o resultados também sugerem que dados de outros cenários de mobilidade também podem ser gerados com eficiência. É importante ressaltar, entretanto, que apesar de as análises serem feitas de maneira automática, a análise qualitativa fornece informações valiosas para a avaliação do desempenho dos modelos.

7 Conclusões

O estudo da mobilidade urbana é essencial para diversas tecnologias e protocolos baseados em redes móveis, como redes móveis sem fio, veiculares ou *ad hoc*. Contudo, a realização desses estudos sofre restrições devido à informações privadas, que são comuns nesse tipo de base de dados. As técnicas existentes para solucionar esse problema, como utilização de modelos geradores sintéticos, enfrentam o desafio de conseguir gerar dados úteis (com as mesmas propriedades dos reais) enquanto mantém a privacidade dos dados. Nesse sentido, modelos generativos baseados em aprendizado profundo, como as Redes Generativas Adversárias, são uma alternativa promissora para a geração de dados de qualidade e com privacidade.

Desta forma, propomos o MobDeep, um arcabouço para a geração e avaliação de modelos geradores de dados de mobilidade urbana baseado em aprendizado profundo. Como etapas para construção do arcabouçou proposto, listamos os seguintes resultados alcançados nesta dissertação: (i) propomos a modelagem de dados mobilidade urbana como séries temporais, permitindo que uma maior gama de dados de mobilidade possam ser gerados; (ii) propomos a utilização de modelos baseados em aprendizado profundo para a geração da séries temporais de mobilidade. Especificamente neste trabalho, utilizamos as GANs que conseguem gerar dados com as propriedades dos reais ao mesmo tempo que mantém a privacidade e; (iii) propomos as avaliações quantitativas e qualitativas dos modelos gerado.

Neste trabalho, o MobDeep é avaliado com dois tipos de modelos: um clássico (ARIMA) e modelos baseados em aprendizado profundo (GANs). Os modelos são treinados utilizando uma base de dados aberta, que contém informações sobre o compartilhamento de bicicletas em 7 cidades dos Estados Unidos e uma base privada, com informações sobre o trânsito da cidade de Vitória-ES. Os modelos treinados são avaliados quantitativamente, por meio da análise dos resíduos e qualitativamente, por meio de análises visuais que comparam os dados sintéticos com os reais. Os resultados mostram que os modelos baseados em aprendizado profundo conseguem modelar com eficiências as séries temporais que representam os dados.

As limitações da solução proposta referem-se ao tipo de informação presente nos dados que serão modelados (precisam ser numéricos) e que representem a sumarização de algum tipo de informação de mobilidade e a forma como as avaliações qualitativas são feitas (o arcabouço irá gerar uma avaliação para cada variável dos dados). Com relação à primeira limitação alguns estudos sobre o desempenho dos modelos utilizados para a geração de dados de outros tipos, como localização GPS ou categóricos, precisam ser feitos.

7.1 Trabalhos Futuros

À seguir apresentamos algumas possíveis pesquisas futuras que podem ser feitas à partir dos conhecimentos construídos neste trabalho:

- **Adição de novos modelos**: Nos experimentos, utilizamos o ARIMA e Redes Generativas Adversárias, como modelos generativos internos do MobDeep. O ARIMA possui a limitação de não poder ser usado em bases de dados multivariadas, por isso a adição de um modelo ARIMAX torna-se interessante. Além disso, outros modelos baseados em aprendizado profundo, como os Auto-encoders variacionais (VAEs, na sigla em Inglês), podem ser adicionados.
- **Novas bases de dados**: As bases usadas neste trabalho possuem características interessantes e seu uso foi fundamental para entender o comportamento dos modelos utilizados. Em trabalhos futuros pretendemos utilizar bases de dados com outros tipos de informações, como localizações de usuários ou pontos de interesse e bases de dados com informações numéricas e categóricas;
- **Otimização dos modelos**: Os modelos de aprendizado profundo usados neste trabalho foram desenvolvidos para tratar de séries temporais de maneira geral. Uma possibilidade de pesquisa é estudar formas de otimizar os modelos usados para a geração de dados de mobilidade urbana

Referências

ABRAHAM, B.; LEDOLTER, J. *Statistical methods for forecasting*. [S.l.]: John Wiley & Sons, 2009. v. 234. Citado 2 vezes nas páginas 13 e 59.

AGGARWAL, C. C. et al. Neural networks and deep learning. *Springer*, Springer, v. 10, p. 978–3, 2018. Citado 3 vezes nas páginas 51, 53 e 55.

ALJERI, N.; BOUKERCHE, A. A performance evaluation of time-series mobility prediction for connected vehicular networks. In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. [S.l.: s.n.], 2020. p. 127–131. Citado na página 30.

BAEK, S.; KIM, K. I.; KIM, T.-K. Weakly-supervised domain adaptation via gan and mesh model for estimating 3d hand poses interacting objects. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 6121–6131. Citado na página 28.

BARBOSA, H. et al. Human mobility: Models and applications. *Physics Reports*, Elsevier, v. 734, p. 1–74, 2018. Citado na página 31.

BAUCH, C. T. et al. Dynamically modeling sars and other newly emerging respiratory illnesses: past, present, and future. *Epidemiology*, JSTOR, p. 791–801, 2005. Citado na página 32.

BEERY, S. et al. Synthetic examples improve generalization for rare classes. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. [S.l.: s.n.], 2020. p. 863–873. Citado na página 28.

BHASKARAN, K. et al. Time series regression studies in environmental epidemiology. *International journal of epidemiology*, Oxford University Press, v. 42, n. 4, p. 1187–1195, 2013. Citado na página 35.

BLONDEL, V. D.; DECUYPER, A.; KRINGS, G. A survey of results on mobile phone datasets analysis. *EPJ data science*, Springer, v. 4, n. 1, p. 10, 2015. Citado 2 vezes nas páginas 27 e 29.

BÖCK, S.; SCHEDL, M. Polyphonic piano note transcription with recurrent neural networks. In: IEEE. *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. [S.l.], 2012. p. 121–124. Citado na página 52.

BOX, G. E. et al. *Time series analysis: forecasting and control*. [S.l.]: John Wiley & Sons, 2015. Citado na página 40.

BOX, G. E.; TIAO, G. C. Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical association*, Taylor & Francis, v. 70, n. 349, p. 70–79, 1975. Citado na página 70.

BRASIL. 2018. <https://www2.camara.leg.br/legin/fed/lei/2018/lei-13709-14-agosto-2018-787077-publicacaooriginal-156212-pl.html>. Acessado em: 13 Ago. 2021. Citado 2 vezes nas páginas 27 e 29.

BROCK, A.; DONAHUE, J.; SIMONYAN, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. Citado 4 vezes nas páginas 13, 28, 57 e 58.

BROCKWELL, P. J.; DAVIS, R. A. *Time series: theory and methods*. [S.l.]: Springer Science & Business Media, 2009. Citado 4 vezes nas páginas 13, 30, 35 e 36.

CCP. 2021. <https://www.waze.com/wazeforcities>. Acessado em : 2010-09-30. Citado na página 88.

CHAINTREAU, A. et al. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, IEEE, v. 6, n. 6, p. 606–620, 2007. Citado na página 64.

CHATFIELD, C. *The analysis of time series: an introduction*. [S.l.]: Chapman and Hall/CRC, 2003. Citado 4 vezes nas páginas 35, 37, 38 e 39.

CHEN, J. et al. Generative dynamic link prediction. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, AIP Publishing LLC, v. 29, n. 12, p. 123111, 2019. Citado 3 vezes nas páginas 64, 65 e 66.

CHOLLET, F. et al. *Deep learning with Python*. [S.l.]: Manning New York, 2018. v. 361. Citado 9 vezes nas páginas 13, 30, 42, 43, 47, 48, 49, 51 e 72.

CISCO. 2018. Https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Acessado em: 15 Ago. 2021. Citado na página 31.

CONNOR, J. T.; MARTIN, R. D.; ATLAS, L. E. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, IEEE, v. 5, n. 2, p. 240–254, 1994. Citado na página 52.

DOMENICO, M. D.; LIMA, A.; MUSOLESI, M. Interdependence and predictability of human mobility and social interactions. *Pervasive and Mobile Computing*, Elsevier, v. 9, n. 6, p. 798–807, 2013. Citado na página 30.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, v. 12, n. 7, 2011. Citado na página 49.

ESTEBAN, C.; HYLAND, S. L.; RÄTSCH, G. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017. Citado 2 vezes nas páginas 46 e 70.

FALBO, R. de A. Mapeamento sistemático. *Retrieved October*, v. 7, 2018. Citado na página 61.

FANAEE-T, H.; GAMA, J. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, Springer Berlin Heidelberg, p. 1–15, 2013. ISSN 2192-6352. Disponível em: <[WebLink]>. Citado na página 83.

FENG, J. et al. Learning to simulate human mobility. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2020. p. 3426–3433. Citado 3 vezes nas páginas 63, 65 e 66.

GERS, F. A.; SCHMIDHUBER, J. Recurrent nets that time and count. In: IEEE. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. [S.l.], 2000. v. 3, p. 189–194. Citado na página 53.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. *Neural computation*, MIT Press, v. 12, n. 10, p. 2451–2471, 2000. Citado 2 vezes nas páginas 53 e 54.

GILES, C. L.; LAWRENCE, S.; TSOI, A. C. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, Springer, v. 44, n. 1, p. 161–183, 2001. Citado na página 52.

GO, T. et al. Deep learning-based hologram generation using a white light source. *Scientific reports*, Nature Publishing Group, v. 10, n. 1, p. 1–12, 2020. Citado na página 28.

GOMES, M. F. et al. Assessing the international spreading risk associated with the 2014 west african ebola outbreak. *PLoS currents*, Public Library of Science, v. 6, 2014. Citado na página 32.

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680. Citado 3 vezes nas páginas 13, 28 e 56.

GOODFELLOW, I. et al. Generative adversarial nets. In: GHAHRAMANI, Z. et al. (Ed.). *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014. p. 2672–2680. Disponível em: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>. Citado 3 vezes nas páginas 13, 56 e 57.

GRANGER, C. W. J.; NEWBOLD, P. *Forecasting economic time series*. [S.l.]: Academic Press, 2014. Citado na página 35.

GRAVES, A.; JAITLY, N. Towards end-to-end speech recognition with recurrent neural networks. In: PMLR. *International conference on machine learning*. [S.l.], 2014. p. 1764–1772. Citado 2 vezes nas páginas 52 e 55.

GREFF, K. et al. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 28, n. 10, p. 2222–2232, 2016. Citado 2 vezes nas páginas 54 e 56.

GUPTA, A. et al. Social gan: Socially acceptable trajectories with generative adversarial networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 2255–2264. Citado 4 vezes nas páginas 29, 62, 65 e 66.

HAYKIN, S. S. *Neural Networks and Learning Machines*. Upper Saddle River, NJ: Pearson Education, 2009. Citado 6 vezes nas páginas 13, 43, 44, 45, 49 e 50.

HE, M. et al. Global traffic state recovery via local observations with generative adversarial networks. In: IEEE. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2020. p. 3767–3771. Citado 2 vezes nas páginas 65 e 66.

HELBING, D.; FARKAS, I.; VICSEK, T. Simulating dynamical features of escape panic. *Nature*, Nature Publishing Group, v. 407, n. 6803, p. 487–490, 2000. Citado na página 32.

HELBING, D.; JOHANSSON, A.; AL-ABIDEEN, H. Z. Dynamics of crowd disasters: An empirical study. *Physical review E*, APS, v. 75, n. 4, p. 046109, 2007. Citado na página 32.

HENDERSON, M.; THOMSON, B.; YOUNG, S. Word-based dialog state tracking with recurrent neural networks. In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. [S.l.: s.n.], 2014. p. 292–299. Citado na página 52.

HILLIER, B. et al. Metric and topo-geometric properties of urban street networks: some convergences, divergences and new results. *Journal of Space Syntax Studies*, 2009. Citado na página 32.

HINTON, G.; TIELEMAN, T. *Lecture 6.5 - RMSProp*. [S.l.]: COURSERA: Neural Networks for Machine Learning, 2012. Citado na página 49.

HINTON, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, MIT Press, v. 14, n. 8, p. 1771–1800, 2002. Citado na página 28.

HOCHREITER, S. et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. [S.l.]: A field guide to dynamical recurrent neural networks. IEEE Press, 2001. Citado na página 53.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado 2 vezes nas páginas 28 e 53.

HUANG, Z.; TATEM, A. J. Global malaria connectivity through air travel. *Malaria journal*, Springer, v. 12, n. 1, p. 1–11, 2013. Citado na página 32.

ISELLA, L. et al. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, Elsevier, v. 271, n. 1, p. 166–180, 2011. Citado na página 64.

ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1125–1134. Citado na página 28.

JAUHRI, A. et al. Generating realistic ride-hailing datasets using gans. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, ACM New York, NY, USA, v. 6, n. 3, p. 1–14, 2020. Citado 5 vezes nas páginas 29, 30, 63, 65 e 66.

JOHANSSON, A. et al. From crowd dynamics to crowd safety: a video-based analysis. *Advances in Complex Systems*, World Scientific, v. 11, n. 04, p. 497–527, 2008. Citado na página 32.

JONES, K. E. et al. Global trends in emerging infectious diseases. *Nature*, Nature Publishing Group, v. 451, n. 7181, p. 990–993, 2008. Citado na página 32.

KAHOU, S. E. et al. Recurrent neural networks for emotion recognition in video. In: *Proceedings of the 2015 ACM on international conference on multimodal interaction*. [S.l.: s.n.], 2015. p. 467–474. Citado na página 52.

KARIM, F. et al. Multivariate lstm-fcns for time series classification. *Neural Networks*, Elsevier, v. 116, p. 237–245, 2019. Citado na página 55.

KARRAS, T. et al. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. Citado 2 vezes nas páginas 13 e 58.

KESKAR, N. S. et al. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. Citado na página 106.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 49.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. Citado na página 70.

KITAMURA, R. et al. Micro-simulation of daily activity-travel patterns for travel demand forecasting. *Transportation*, Springer, v. 27, n. 1, p. 25–51, 2000. Citado na página 32.

KIUKKONEN, N. et al. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin*, v. 68, p. 7, 2010. Citado na página 62.

KRAEMER, M. U. et al. The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, American Association for the Advancement of Science, v. 368, n. 6490, p. 493–497, 2020. Citado na página 32.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, v. 3361, n. 10, p. 1995, 1995. Citado na página 28.

LECUN, Y.; CORTES, C.; BURGES, C. J. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, v. 10, n. 34, p. 14, 1998. Citado na página 71.

LEDIG, C. et al. Photo-realistic single image super-resolution using a generative adversarial network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4681–4690. Citado na página 28.

LEI, K. et al. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In: IEEE. *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. [S.l.], 2019. p. 388–396. Citado 4 vezes nas páginas 29, 64, 65 e 66.

LENKEI, Z. *Crowdsourced traffic information in traffic management: Evaluation of traffic information from Waze*. 2018. Citado 2 vezes nas páginas 88 e 89.

MA, Q. et al. Learning to dress 3d people in generative clothing. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 6469–6478. Citado na página 28.

MACHADO, J. C.; NETO, E. R. D. Privacidade de dados de localização: Modelos, técnicas e mecanismos. *Sociedade Brasileira de Computação*, 2021. Citado 3 vezes nas páginas 28, 29 e 30.

Malandrino, F.; Chiasserini, C.; Kirkpatrick, S. Cellular network traces towards 5g: Usage, analysis and generation. *IEEE Transactions on Mobile Computing*, v. 17, n. 3, p. 529–542, 2018. Citado na página 27.

MNIH, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint*

arXiv:1312.5602, 2013. Citado na página 47.

MOGREN, O. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016. Citado 2 vezes nas páginas 46 e 70.

MOKHAYERI, F.; KAMALI, K.; GRANGER, E. Cross-domain face synthesis using a controllable gan. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. [S.l.: s.n.], 2020. p. 252–260. Citado na página 28.

MONTGOMERY, D. C.; HINES, W. W. *Probability and statistics in engineering and management science*. [S.l.]: John Wiley & Sons, 1980. Citado na página 41.

MONTGOMERY, D. C.; JENNINGS, C. L.; KULAHCI, M. *Introduction to time series analysis and forecasting*. [S.l.]: John Wiley & Sons, 2015. Citado na página 71.

MOTA, V. F. et al. Protocols, mobility models and tools in opportunistic networks: A survey. *Computer Communications*, v. 48, p. 5 – 19, 2014. ISSN 0140-3664. Opportunistic networks. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366414001054>. Citado na página 27.

NAYEBI, A.; VITELLI, M. Gruv: Algorithmic music generation using recurrent neural networks. *Course CS224D: Deep Learning for Natural Language Processing (Stanford)*, 2015. Citado na página 52.

OUYANG, K. et al. A non-parametric generative model for human trajectories. In: *IJCAI*. [S.l.: s.n.], 2018. p. 3812–3817. Citado 3 vezes nas páginas 62, 65 e 66.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, v. 2, n. 417, p. 1, 2012. Citado na página 53.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: PMLR. *International conference on machine learning*. [S.l.], 2013. p. 1310–1318. Citado na página 53.

PATTERSON, J.; GIBSON, A. *Deep learning: A practitioner's approach*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado 14 vezes nas páginas 13, 42, 43, 44, 45, 46, 48, 49, 51, 52, 53, 54, 55 e 56.

PETRY, L. M. et al. Marc: a robust method for multiple-aspect trajectory classification via space, time, and semantic embeddings. *International Journal of Geographical Information Science*, Taylor & Francis, v. 34, n. 7, p. 1428–1450, 2020. Citado na página 63.

PINCUS, S. M.; GOLDBERGER, A. L. Physiological time-series analysis: what does regularity quantify? *American Journal of Physiology-Heart and Circulatory Physiology*, American Physiological Society Bethesda, MD, v. 266, n. 4, p. H1643–H1656, 1994. Citado na página 35.

QU, Y. et al. Gan-driven personalized spatial-temporal private data sharing in cyber-physical social systems. *IEEE Transactions on Network Science and Engineering*, IEEE, 2020. Citado 4 vezes nas páginas 29, 64, 65 e 66.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with

deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. Citado 2 vezes nas páginas 46 e 57.

RAO, J. et al. Lstm-trajgan: A deep learning approach to trajectory privacy protection. *arXiv preprint arXiv:2006.10521*, 2020. Citado 3 vezes nas páginas 63, 65 e 66.

REZENDE, D. J.; MOHAMED, S.; WIERSTRA, D. Stochastic backpropagation and approximate inference in deep generative models. In: PMLR. *International conference on machine learning*. [S.l.], 2014. p. 1278–1286. Citado na página 70.

RIBEIRO, I. et al. Caracterização de mobilidade e detecção de comunidades baseadas em tópicos de interesse. In: *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2020. p. 603–616. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/12312>. Citado na página 31.

RIBEIRO, I. et al. Mobility and community detection based on topics of interest. In: IEEE. *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. [S.l.], 2021. p. 1–6. Citado na página 32.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 43.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado na página 28.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 49.

SALEHYAN, I.; GLEDITSCH, K. S. Refugees and the spread of civil war. *International organization*, Cambridge University Press, v. 60, n. 2, p. 335–366, 2006. Citado na página 35.

SALIMANS, T. et al. Improved techniques for training gans. *Advances in neural information processing systems*, v. 29, p. 2234–2242, 2016. Citado na página 57.

SERBAN, I. et al. Multiresolution recurrent neural networks: An application to dialogue response generation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2017. v. 31, n. 1. Citado na página 52.

SHUMWAY, R. H.; STOFFER, D. S. *Time series analysis and its applications*. [S.l.]: Springer, 2000. v. 3. Citado na página 37.

SONG, C. et al. Limits of predictability in human mobility. *Science*, American Association for the Advancement of Science, v. 327, n. 5968, p. 1018–1021, 2010. Citado na página 30.

SONG, H. Y.; BAEK, M. S.; SUNG, M. Generating human mobility route based on generative adversarial network. In: IEEE. *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. [S.l.], 2019. p. 91–99. Citado 5 vezes nas páginas 29, 30, 62, 65 e 66.

SUSSKIND, J.; ANDERSON, A.; HINTON, G. E. *The Toronto face dataset*. [S.l.], 2010. Citado 2 vezes nas páginas 13 e 57.

SUTSKEVER, I.; MARTENS, J.; HINTON, G. E. Generating text with recurrent neural networks. In: *ICML*. [S.l.: s.n.], 2011. Citado na página 52.

TERAMOTO, A. et al. Deep learning approach to classification of lung cytological images: Two-step training using actual and synthesized images by progressive growing of generative adversarial networks. *PloS one*, Public Library of Science San Francisco, CA USA, v. 15, n. 3, p. e0229951, 2020. Citado na página 28.

THOMÉ, M. et al. Um arcabouço para detecção e alerta de anomalias de mobilidade urbana em tempo real. In: SBC. *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.], 2020. p. 784–797. Citado 2 vezes nas páginas 30 e 89.

VALLENDER, S. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, SIAM, v. 18, n. 4, p. 784–786, 1974. Citado na página 64.

VANNONI, M. et al. Using volunteered geographic information to assess mobility in the early phases of the covid-19 pandemic: a cross-city time series analysis of 41 cities in 22 countries from march 2nd to 26th 2020. *Globalization and health*, BioMed Central, v. 16, n. 1, p. 1–9, 2020. Citado na página 30.

VENUGOPALAN, S. et al. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014. Citado na página 52.

WANG, C. et al. Image captioning with deep bidirectional lstms. In: *Proceedings of the 24th ACM international conference on Multimedia*. [S.l.: s.n.], 2016. p. 988–997. Citado na página 55.

WANG, J. et al. Cnn-rnn: A unified framework for multi-label image classification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 2285–2294. Citado na página 46.

WANG, Y. et al. Attention-based lstm for aspect-level sentiment classification. In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. [S.l.: s.n.], 2016. p. 606–615. Citado na página 55.

WEN, T.-H. et al. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*, 2015. Citado na página 52.

WU, Y. et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. Citado na página 55.

XU, D. et al. Ge-gan: A novel deep learning framework for road traffic state estimation. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 117, p. 102635, 2020. Citado na página 28.

YIN, D.; YANG, Q. Gans based density distribution privacy-preservation on mobility data. *Security and Communication Networks*, Hindawi, v. 2018, 2018. Citado 3 vezes nas páginas 63, 65 e 66.

YOON, J.; JARRETT, D.; SCHAAR, M. van der. Time-series generative adversarial networks. 2019. Citado 2 vezes nas páginas 46 e 71.

YOU, Q. et al. Image captioning with semantic attention. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 4651–4659. Citado na página 52.

YU, H. et al. Taxi-based mobility demand formulation and prediction using conditional generative adversarial network-driven learning approaches. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 20, n. 10, p. 3888–3899, 2019. Citado 3 vezes nas páginas 63, 64 e 66.

YU, H. et al. Extracting and predicting taxi hotspots in spatiotemporal dimensions using conditional generative adversarial neural networks. *IEEE Transactions on Vehicular Technology*, IEEE, v. 69, n. 4, p. 3680–3692, 2020. Citado 3 vezes nas páginas 63, 64 e 66.

YU, Y. et al. Point encoder gan: A deep learning model for 3d point cloud inpainting. *Neurocomputing*, Elsevier, v. 384, p. 192–199, 2020. Citado na página 28.

ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, Elsevier, v. 50, p. 159–175, 2003. Citado na página 39.

ZHANG, H. et al. Understanding and modeling urban mobility dynamics via disentangled representation learning. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2020. Citado 5 vezes nas páginas 29, 30, 62, 65 e 66.

ZHANG, L. Stggan: Spatial-temporal graph generation. In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. [S.l.: s.n.], 2019. p. 608–609. Citado na página 64.

ZHAO, J. et al. 3d-aided dual-agent gans for unconstrained face recognition. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 41, n. 10, p. 2380–2394, 2018. Citado na página 28.

ZHENG, Y. et al. Mining interesting locations and travel sequences from gps trajectories. In: ACM. *World wide web*. [S.l.], 2009. p. 791–800. Citado na página 27.

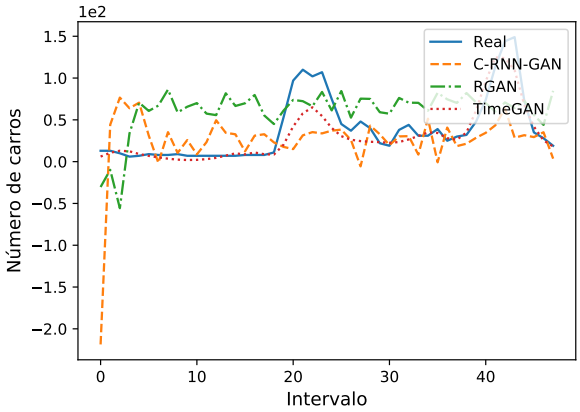
ZHOU, H. et al. 3d high resolution generative deep-learning network for fluorescence microscopy imaging. *Optics letters*, Optical Society of America, v. 45, n. 7, p. 1695–1698, 2020. Citado na página 28.

ZHU, J.-Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2223–2232. Citado na página 28.

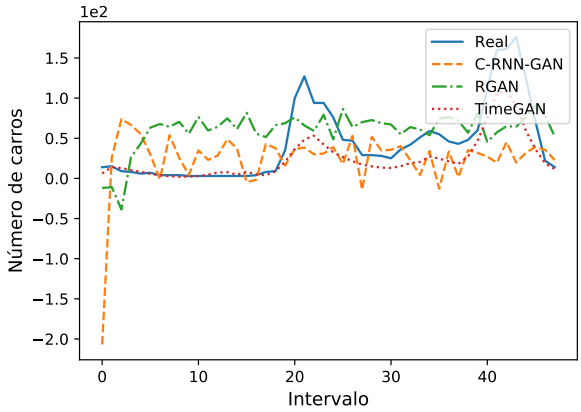
Apêndices

APÊNDICE A – Somas por intervalo

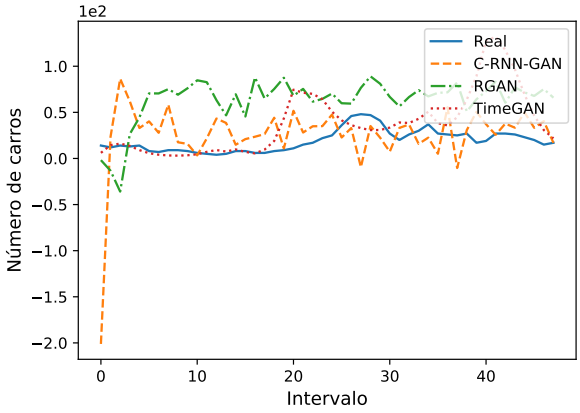
À seguir, as Figura 39, 40, 41, 42 e 43, apresentam, respectivamente, as somas por intervalo das Ruas IV, V, VI, VII e VIII. As somas mostradas consideram os dias de Segunda-feira, Quarta-feira e Sábado. Para todas as 5 bases, a TimeGAN foi o modeloe que gerou dados com somas por intervalo que mais se assemelham aos reais, exceto no somas representando os Sábados. Os modelos C-RNN-GAN e RGAN, na maioria das vezes, geraram somas com valores negativos.



(a) Segunda-feira

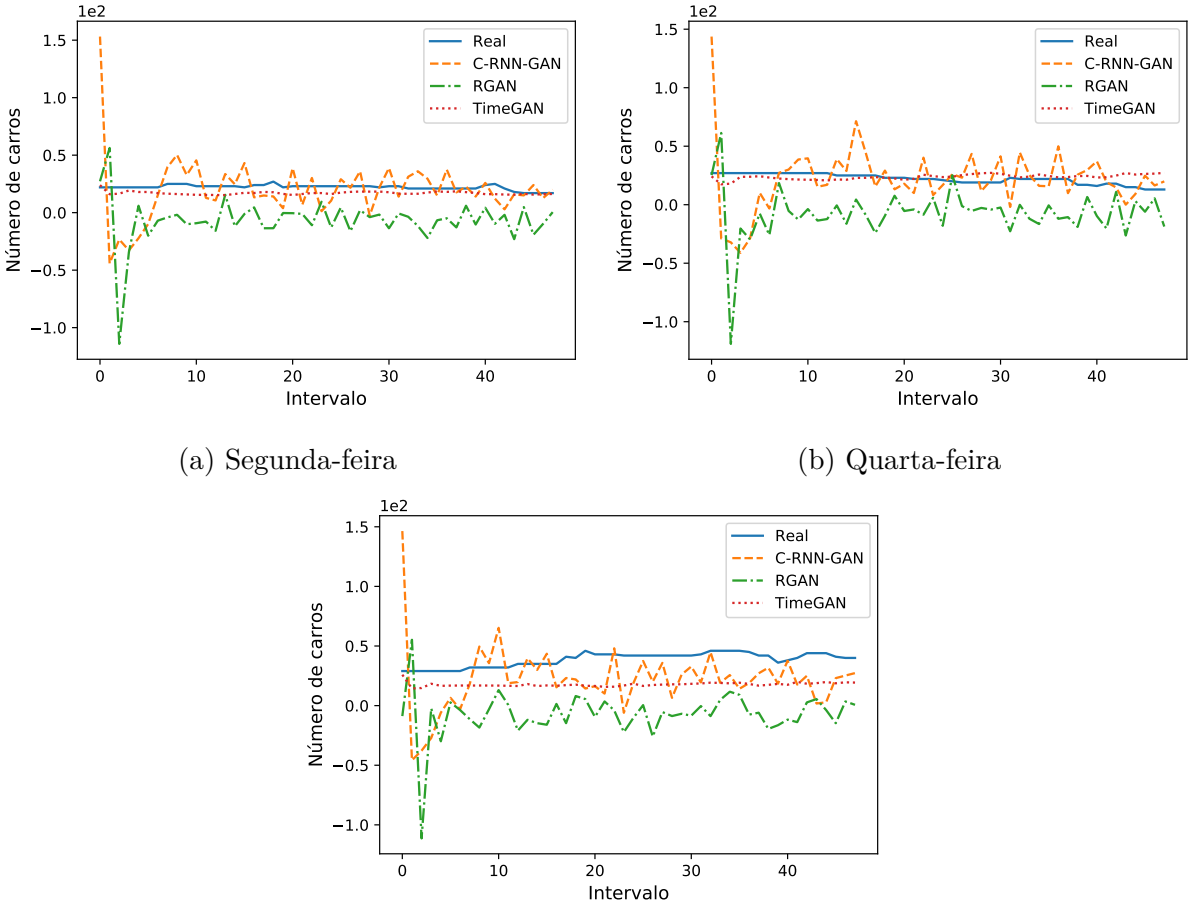


(b) Quarta-feira



(c) Sábado

Figura 39 – Somas por intervalo dos modelos para a Rua IV, considerando a Segunda (a), a Quarta (b) e o Sábado (c). Exceto pelo final de semana, a TimeGAN foi o modelo que gerou dados com características mais similares aos reais. Contudo, nenhum modelo conseguiu gerar dados de qualidade durante o final de semana e os C-RNN-GAN e a RGAN geraram dados negativos.

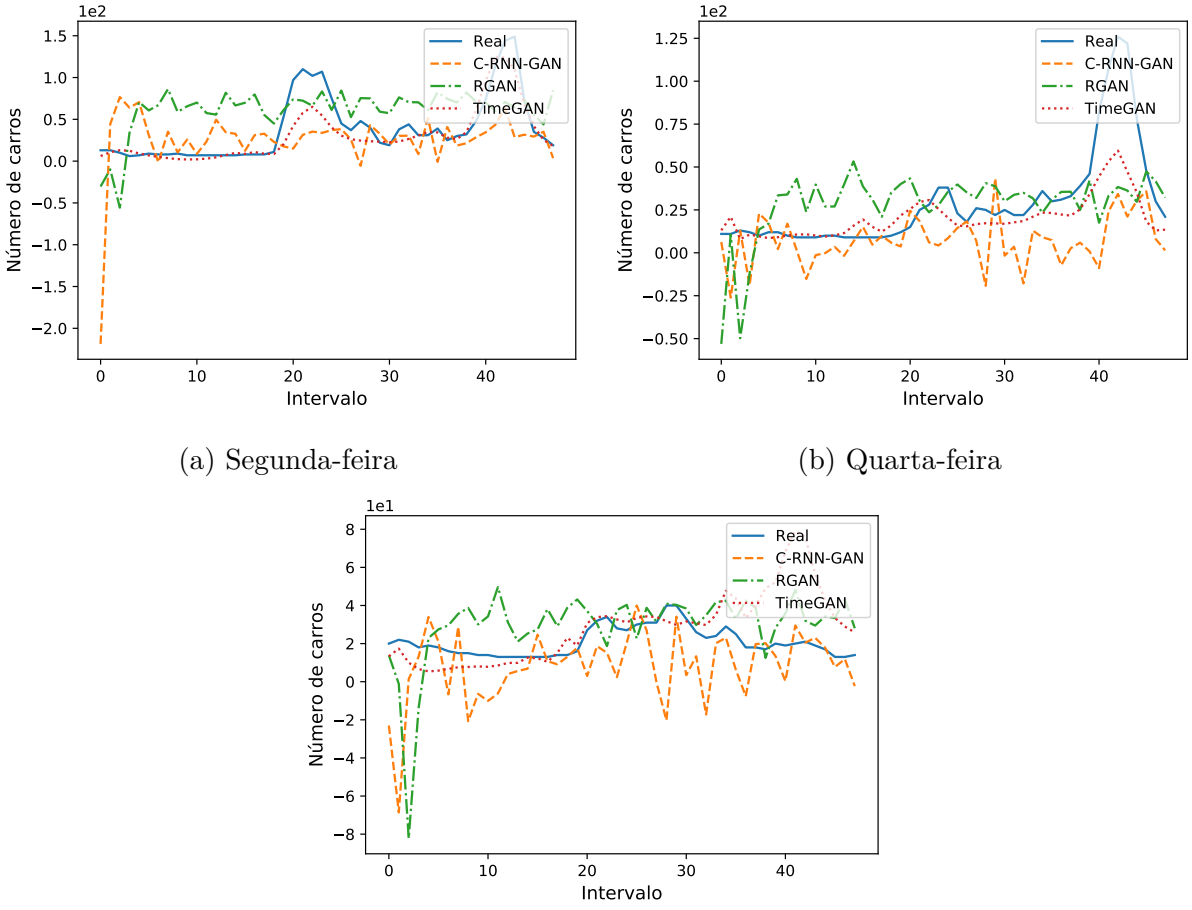


(a) Segunda-feira

(b) Quarta-feira

(c) Sábado

Figura 40 – Somas por intervalo dos modelos para a Rua V, para a Segunda (a), a Quarta (b) e o Sábado (c). A TimeGAN foi o modelo que gerou dados com características mais similares aos reais, exceto pelo final de semana, em que nenhum modelo conseguiu gerar dados de qualidade. A C-RNN-GAN e a RGAN geraram somas por intervalo com valores negativos.

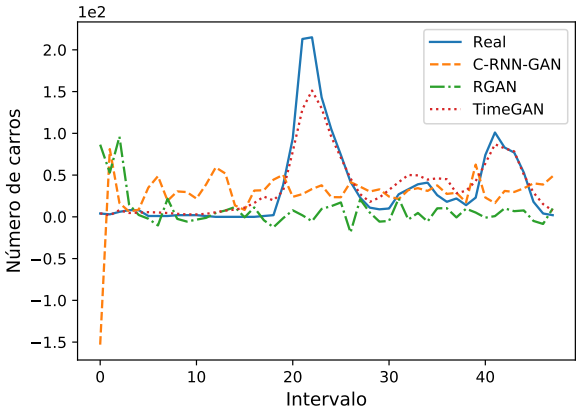


(a) Segunda-feira

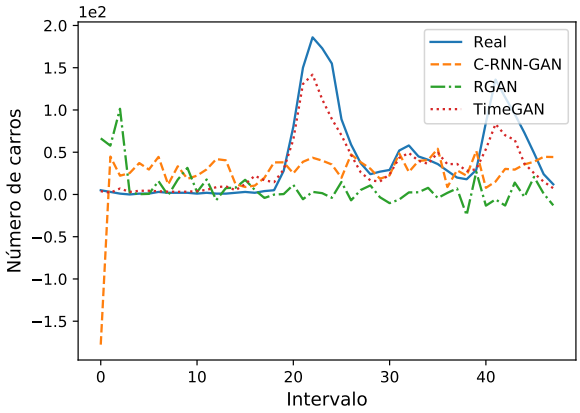
(b) Quarta-feira

(c) Sábado

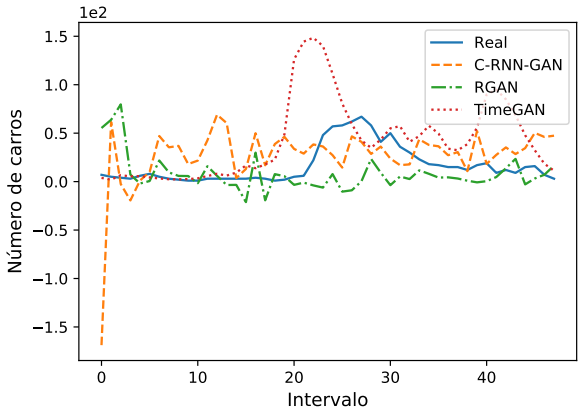
Figura 41 – Somas por intervalo dos modelos para a Rua VI, para dados agrupados entre a Segunda (a), a Quarta (b) e o Sábado (c). A TimeGAN conseguiu capturar as características dos dados e, durante boa parte do Sábado (c), gerou dados próximos dos reais. Mais uma vez a C-RNN-GAN e a RGAN geraram dados negativos.



(a) Segunda-feira

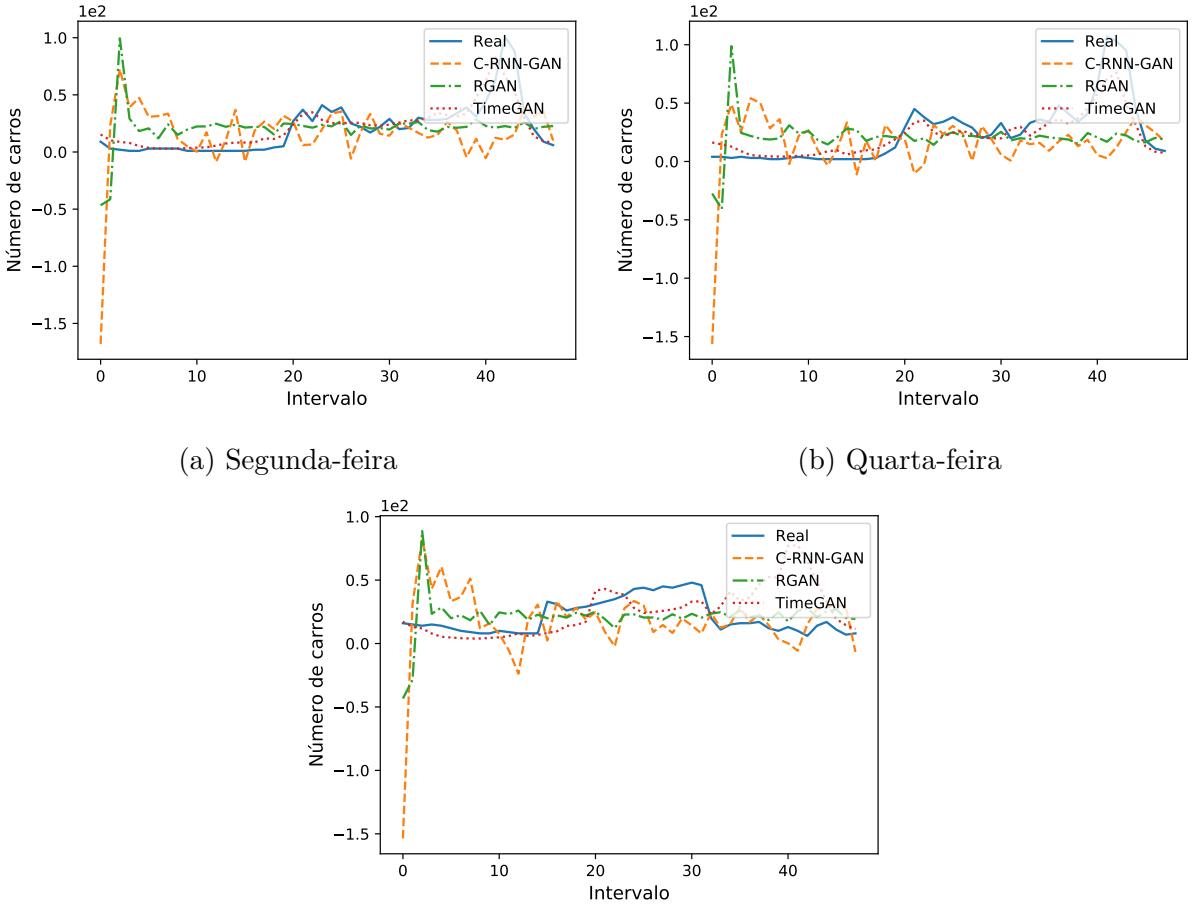


(b) Quarta-feira



(c) Sábado

Figura 42 – Somas por intervalo dos modelos para a Rua VII, para a Segunda (a), a Quarta (b) e o Sábado (c). Durante os dias úteis, a TimeGAN gerou dados muito parecidos com os reais. No Sábado (c), a TimeGAN gerou dados próximos dos reais apenas nas primeiras horas. Diferente das situações anteriores, a RGAN, no Sábado (c) gerou poucos dados negativos. A C-RNN-GAN, contudo, não conseguiu modelar as características dos dados reais.



(a) Segunda-feira

(b) Quarta-feira

(c) Sábado

Figura 43 – Somas por intervalo dos modelos para a Rua VIII, para a Segunda (a), a Quarta (b) e o Sábado (c). A TimeGAN gerou dados com características similares aos reais, durante os dias úteis. No Sábado (c) a TimeGAN acompanhou, levemente, o comportamento apresentando pelos dados reais. Novamente, a C-RNN-GAN e a RGAN geraram dados negativos.

APÊNDICE B – Parâmetros

A Tabela 17 apresenta a variação dos principais parâmetros dos modelos baseados em aprendizado profundo, C-RNN-GAN, RGAN e TimeGAN. Ressalta-se que as bases de Rua I à VIII são levemente diferentes entre si, entretanto os mesmos parâmetros puderam ser aplicados. O objetivo da variação é, além de encontrar o melhor conjunto de parâmetros, identificar como os modelos se comportam com cada combinação.

Tabela 17 – Variação dos principais parâmetros para os modelos C-RNN-GAN, RGAN e TimeGAN, considerando as base *BikeSharing* e as bases Ruas I à VIII. Para simplificar essas bases são representadas pela base *Trânsito Vitória*

Modelo	**Parâmetros**	**BikeSharing**	**Trânsito Vitória**
C-RNN-GAN	bs	28, 56, 84, 168	28, 50, 75
	lr	.001, .0001, .0005	.001, .0001, .0005
	hd	50, 75, 100, 125	50, 75, 100, 125
	ep	50, 100, 200	100, 200, 300
RGAN	bs	28, 56, 84, 168	28, 50, 75
	lr	.01, .1, .2	.01, .1, .2
	hd	25, 50, 75, 100	16, 32, 64
	ep	300, 1000, 2000, 3000	300, 1000, 2000, 3000
TimeGAN	bs	28, 56, 84, 168	25, 50, 75
	lr	.0004, .0005, .0006	.0004, .0005, .0006
	hd	24, 48, 72	24, 48, 72
	ep	2000, 3000, 4000, 5000	2000, 3000, 4000, 5000