UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DOUTORADO EM ENGENHARIA ELÉTRICA

PEDRO HENRIQUE FEIJÓ DE SOUSA

LIDAR3DNET - ABORDAGEM INTELIGENTE PARA CLASSIFICAR OBJETOS 3D COM BASE EM NUVENS DE PONTOS REAIS E SINTÉTICAS

FORTALEZA

2022

PEDRO HENRIQUE FEIJÓ DE SOUSA

LIDAR3DNET - ABORDAGEM INTELIGENTE PARA CLASSIFICAR OBJETOS 3D COM BASE EM NUVENS DE PONTOS REAIS E SINTÉTICAS

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia Elétrica. Área de Concentração: Sistema de Energia Elétrica

Orientador: Prof. Dr. Pedro Pedrosa Rebouças Filho

Coorientador: Prof. Dr. Bismark Claure Torrico

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S698 Sousa, Pedro Henrique Feijó de.
LIDAR3DNET - ABORDAGEM INTELIGENTE PARA CLASSIFICAR OBJETOS 3D COM BASE EM NUVENS DE PONTOS REAIS E SINTÉTICAS / Pedro Henrique Feijó de Sousa. – 2022.
84 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2022.
Orientação: Prof. Dr. Pedro Pedrosa Rebouças Filho.
Coorientação: Prof. Dr. Bismark Claure Torrico.

1. Nuvem de pontos. 2. Aprendizado de Máquinas. 3. Classificação de objetos 3D. 4. Lidar. I. Título.

CDD 621.3

ATA DA SESSÃO DE DEFESA DE TESE DE DOUTORADO

UNIVERSIDADE FEDERAL DO CEARÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Como parte das exigências para concessão do grau de doutor, às 09:00 horas do dia 02 de Dezembro de 2022, realizou-se a sessão pública da defesa de tese de doutorado do aluno PEDRO HENRIQUE FEIJÓ DE SOUSA. O trabalho tinha como título: "Abordagem inteligente para classificar objetos 3D com base em nuvens de pontos reais e sintéticas".

Compunham a banca examinadora os professores(as) doutores(as) PEDRO PEDROSA REBOUCAS FILHO, orientador, BISMARK CLAURE TORRICO, coorientador, FABRICIO GONZALEZ NOGUEIRA, ANTONIO CARLOS DA SILVA BARROS e LEANDRO BEZERRA MARINHO. O candidato expôs oralmente a tese, em seguida os membros da banca procederam à arguição, e a sessão foi finalizada com a APROVAÇÃO, por parte da banca examinadora, do trabalho sem ressalvas.

Foi lavrada a presente ata que é abaixo assinada pelos membros da referida banca:

PEDRO PEDROSA REBOUCAS FILHO
UFC - Orientador

BISMARK CLAURE TORRICO
UFC - Coorientador

FABRICIO GONZALEZ NOGUEIRA
UFC - Examinador Interno

ANTONIO CARLOS DA SILVA BARROS
UNILAB - Examinador Externo à Instituição

LEANDRO BEZERRA MARINHO
IFCE - Examinador Externo à Instituição

Fortaleza, 02 de Dezembro de 2022

RESUMO

Utilizar nuvem de pontos de *datasets* da literatura e geradas em simuladores e evitam custos de tempo de coleta. Dessa forma, há acesso a uma quantidade ilimitada e organizada de nuvens de pontos, um cenário ideal para redes de aprendizado profundo. No entanto, modelos que utilizam redes neurais convolucionais e dados sintéticos apresentam limitações quando aplicados a dados do mundo real. Este trabalho propõe um método baseado em perceptron multicamadas (MLP) para classificação de objetos 3D com base em nuvens de pontos reais, aquisitadas com o sensor Light Detection and Ranging (LiDAR), e uma etapa de pré-processamento. Além disso, são propostos dois *dataset*, com dados reais e sintéticos, e a pesquisa utiliza o *dataset* ModelNet para fomentar as comparações. A rede neural proposta, Lidar3DNet, possui arquitetura baseada em MLP, pois suporta a elevada quantidade de dados como entrada. Com o objetivo de proporcionar o melhor desempenho das redes neurais selecionadas, é prosposta uma etapa de pré-processamento com duas tratativas, normalização dos pontos e ajuste da nuvem de pontos no plano cartesiano 3D, para contornar discrepâncias na distribuição dos pontos das nuvens de pontos. A pesquisa é composta por quatro etapas: (1) são selecionadas as nove redes com melhor desempenho no *dataset* ModelNet40 e todas as arquiteturas são configuradas e executadas. em seguida, (2) o uso do dataset ModelNet40 é substituído pelo *dataset* com nuvens de pontos proposto, variando a aplicação das nuvens de pontos originais e pré-processados. Na sequência, (3) o *dataset* com nuvens de pontos reais é substituído pelo *dataset* com nuvens de pontos sintéticas, também com nuvens de pontos originais e pré-processados. Nessas etapas, cinco métricas de avaliação foram aplicadas em conjunto com dois métodos de avaliação estatística. Por fim, (4) uma variação na quantidade de pontos das nuvens de pontos é implementada para avaliar a interferência da quantidade de pontos no desempenho da rede Lidar3DNet, com testes em CPU e GPU e com dados reais e sintéticos. A etapa de pré-processamento proporcionou aumento no desempenho das redes neurais selecionadas com ganhos de 25% no melhor caso. A rede proposta alcançou 98,33% de acurácia e um tempo de teste de 88 μs com dados sintético e com dados reais atingiu 98,47% e 125 μs em acurácia e tempo de teste, respectivamente. Avaliação da variação da quantidade de pontos das nuvens de pontos para classificação de objetos 3D indicou que o uso de GPU é válido a partir de nuvens de pontos com mais de 1024 pontos.

Palavras-chave: Nuvem de pontos, aprendizado de máquina, classificação de objetos 3D e LiDAR

ABSTRACT

Use point clouds of *datasets* from the literature and generated in simulators and avoid collection time costs. In this way, there is access to an unlimited and organized amount of point clouds, an ideal scenario for deep learning networks. However, models that use convolutional neural networks and synthetic data have limitations when applied to real-world data. This work proposes a multilayer perceptron (MLP)-based method for classifying 3D objects based on real point clouds, acquired with the Light Detection and Ranging (LiDAR) sensor, and a pre-processing step. In addition, two *dataset* are proposed, with real and synthetic data, and the research uses *dataset* ModelNet to encourage comparisons. The proposed neural network, Lidar3DNet, has an architecture based on MLP, as it supports a high amount of data as input. To provide the best performance of the selected neural networks, a pre-processing step is proposed with two treatments, normalization of points and adjustment of the point cloud in the 3D Cartesian plane, to overcome discrepancies in the distribution of points in the point clouds. The research consists of four steps: (1) the nine networks with the best performance in the *dataset* ModelNet40 are selected and all architectures are configured and executed. then (2) the use of the ModelNet40 dataset is replaced by the proposed *dataset* with point clouds, varying the application of the original and pre-processed point clouds. Next, (3) the *dataset* with real point clouds is replaced by the *dataset* with synthetic point clouds, also with original and pre-processed point clouds. In these stages, five evaluation metrics were applied together with two statistical evaluation methods. Finally, (4) a variation in the number of points of the point clouds is implemented to evaluate the interference of the number of points in the performance of the Lidar3DNet network, with CPU and GPU tests and with real and synthetic data. The pre-processing step provided an increase in the performance of the selected neural networks with gains of 25% in the best case. The proposed network reached 98.33% accuracy and a test time of 88 μs with synthetic data and with real data it reached 98.47% and 125 μs in accuracy and test time, respectively. Evaluation of the variation in the number of points of point clouds for the classification of 3D objects indicated that the use of GPU is valid from point clouds with more than 1024 points.

Keywords: Point cloud, machine Learning, 3D object classification and LiDAR.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A forma de visualização mais comum de conteúdo digital é a partir de *displays* com duas dimensões (2D). A câmera é o dispositivo tradicional que captura vídeos e imagens. No entanto, a ascensão de aplicações em três dimensões (3D) desperta a necessidade de visualizações mais detalhadas e de diferentes pontos de vista. Um meio utilizado para visualizar e manipular dados 3D é com malhas, que são formadas por conjunto de faces, vértices e arestas e correspondem a um polígono (QUEIROZ; CHOU, 2016). E o arranjo de polígonos geram o objeto 3D, como ilustra a figura 1 . Porém, a manipulação e compressão das malhas exigem cálculos complexos e necessidade de grande recurso computacional, por esses motivos suas aplicações são limitadas. Nesse contexto, as nuvens de pontos se destacam por oferecer mais liberdade de visualização e manipulação de seus dados.

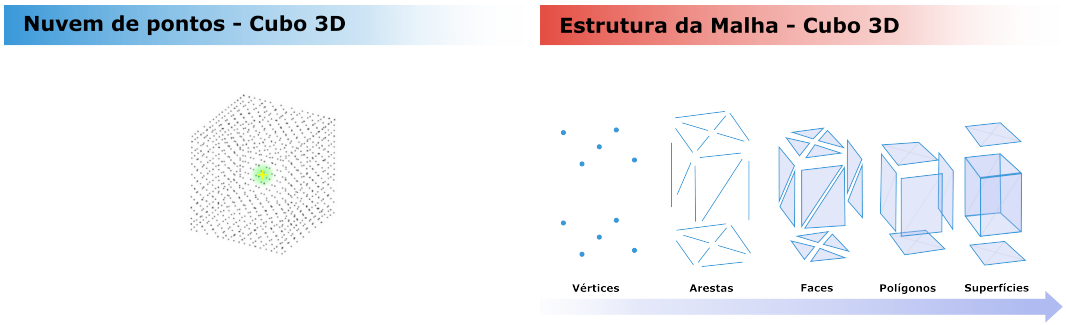


Figura 1 – Diferença entre as estrutura de uma malha 3D e da nuvem de pontos 3D.
Fonte: Elaborado pelo autor.

Nuvem de pontos são conjuntos de dados que representam objetos, formas ou cenas em um determinado espaço. Cada ponto consiste em coordenadas geográficas nos eixos X, Y e Z (RUSU; COUSINS, 2011). Nuvem de pontos é um tipo comum de dados geométricos e pode auxiliar os robôs a entender os ambientes de forma robusta. O voxel é um termo utilizado no contexto 3D e representa uma unidade de informação gráfica que define um ponto no espaço tridimensional. Assim como o pixel (elemento de imagem) define um ponto no espaço bidimensional com suas coordenadas X e Y, o voxel define a unidade no espaço tridimensional adicionando a coordenada chamada Z.

A descrição de objetos, presentes nas nuvem de pontos, incorporados em um rede de veículos conectados gera um fluxo robusto de informações, fornecimento de segurança e transporte inteligente, importantes conceitos na Rede Ad hoc Veicular (VANET) (WANG *et al.*, 2019). Dessa forma, é importante pesquisas no tratamento da informação, segmentação

e classificação de nuvens de pontos. Essas áreas estudos estão presentes em aplicativos que auxiliam na navegação (MILIOTO *et al.*, 2019; MILIOTO *et al.*, 2020; LANGER *et al.*, 2020) , congestionamento de tráfego detecção (MOHANTY *et al.*, 2019), detecção de colisão (ANADU *et al.*, 2018) e redução de pacotes perdas na comunicação de carros autônomos e VANETS (WANG *et al.*, 2019).

Entre as várias representações de dados 3D, as nuvens de pontos são amplamente utilizadas em computação gráfica e visão computacional devido à sua manipulação e visualização amigável (UY *et al.*, 2019). As nuvens de pontos de espaços 3D são aplicados para melhorar a experiência do usuário, pois permite o o usuário navegar e se orientar dentro do modelo 3D com níveis de detalhes que possibilita identificar características locais de interesse (GUO; WANG, 2021). O uso de nuvens de pontos está presentes em aplicações que auxiliam na navegação de veículos (SUN *et al.*, 2020; MEI *et al.*, 2022; CAESAR *et al.*, 2020), especificamente na detecção de espaços livres (FAN *et al.*, ; NAGY; ONIGA, 2021), detecção e segmentação de estradas (FAN *et al.*, 2021; CHEN *et al.*, 2019; RAVISHANKAR *et al.*, 2022) detecção de anomalias nas estradas (WANG *et al.*, 2020).

Com o surgimento de recursos computacionais poderosos, como unidades de processamento gráfico (graphics processing unit, GPUs) e a disponibilidade de dados 3D, a partir de sensores de profundidade, o uso de aprendizado profundo em dados 3D verticalizou. De forma incisiva, pesquisas utilizam processamento para segmentar e classificar objetos 3D em tempo real (CHEN; CHANG, 2021; MOHAPATRA *et al.*, 2021; ALNAGGAR *et al.*, 2021). No entanto, as nuvens de pontos são escassas, não estruturadas e desordenadas (CHANG *et al.*, 2015). Nesse contexto, alguns trabalhos propõem *datasets*, além de desenvolverem métodos para manipular, segmentar e classificar nuvens de pontos, por exemplo ShapeNet (WU *et al.*, 2015), ModelNet40 e nuScenes (CAESAR *et al.*, 2020). Neste trabalho utilizamos nuvens de pontos reais e sintéticas, geradas durante a pesquisa, para classificar objetos 3D utilizando técnicas de aprendizado de máquinas.

1.1 Estado da Arte

Os dispositivos mais comuns na geração de nuvens de pontos são os sensores de Detecção e Alcance de Luz (*light detection and ranging*, LiDAR) e o Kinect. Estudos mostram que o sensor *Kinect* tem sido usado para reconstruir virtualmente objetos (FERREIRA; SILVA, 2021), fornecer dados morfológicos 3D de plantas (SUN; WANG, 2019), e auxiliar na avaliação

motora de pacientes idosos para prevenção de quedas (ONO *et al.*, 2020). No entanto, estes trabalhos apresentam uma limitação comum, o alcance de varredura do sensor que é de 8 metros. Além disso, o sensor *Kinect* possui qualidade de medições comprometidas por altos níveis de iluminação externa (ROSELL-POLO *et al.*, 2017). Dessa forma, o *kinect* é limitado a ambientes internos e não indicado em aplicações com veículos autônomos em vias aéreas e terrestres. (ASSAD-UZ-ZAMAN *et al.*, 2021). O sensor LiDAR é uma tecnologia que gera modelos 3D de superfícies, a partir da integração de um sistema de posicionamento global (global positioning system, GPS), um sistema de navegação inercial (inertial navigation system, INS), e um scanner a laser. Após sua varredura, o sensor LiDAR gera uma nuvem de pontos do cenário escaneado. Dessa forma é possível mensurar e descrever objetos ou até propriedades de edifícios de forma não invasiva, preservando o nível de detalhe, precisão e visualização 3D (IZATT *et al.*, 2017).

A segmentação e classificação de nuvens de pontos 3D englobam, entre outros, desafios de escala e estrutura dos dados. As imagens têm arranjos de pixels que fornecem uma organização regular, mas em nuvens de pontos 3D, os pontos organização é um desafio (LI, 2017). Na literatura, os autores propõem abordagens de redes neurais artificiais que usam estratégias para resolver este problema, modelando ou agrupando pontos na nuvem. A rede SO-Net, por exemplo, modela a distribuição espacial da nuvem de pontos criando um Mapa Auto-Organizável (LI *et al.*, 2018). O SEGCloud usa nuvens de pontos de locais urbanos e classifica-os usando a grade de voxels 3D e os recursos de bairros 3D (multiescala). Essa técnica aglomera um conjunto de pontos e gera um voxel, análogo ao pixel presente nas imagens (TCHAPMI *et al.*, 2017). Laudrieu e Simonovsky (LANDRIEU; SIMONOVSKY, 2018) organizam a nuvem de pontos 3D a partir de um aprendizado profundo de geometria, que é um dos desafios, especialmente para objetos 3D (Ahmed et al., 2019; Bronstein et al., 2017; Cao et al., 2020). Devido ao rápido desenvolvimento em aprendizado profundo, esses modelos podem realizar tarefas de classificação e reconhecimento de forma para diferentes tipos de geometrías 3D (Ahmed et al.,2019; Bronstein et al., 2017; Cao et al., 2020; Denk et al., 2019).

Especialmente o uso de redes neuronais convolucionais (convolutional neural network, CNN), permite um alto grau de paralelização. As camadas da CNN são normalmente usadas para estruturas de dados euclidianas, como um pixel 2D ou voxel 3D. Embora os métodos de aprendizado profundo aplicados em estruturas de dados euclidianas, como imagens 3D (voxels) ou imagens 2D (pixels), sejam amplamente estabelecidos, os campos de pesquisa atuais resumidos em (KOTB *et al.*, 2018) (AHMED *et al.*, 2019; BRONSTEIN *et al.*, 2017; CAO *et al.*, 2020)

usam CNN para dados não euclidianos, como grafos, superfícies poligonais e nuvens de pontos. Propriedades como invariância de deslocamento e uma parametrização global na imagem podem falhar em formas não euclidianas(BRONSTEIN *et al.*, 2017), de modo que o processamento em dados não euclidianos é bastante desafiador (BRONSTEIN *et al.*, 2017) (DENK; REISSWIG, 2019).

Com base em diferentes representações de dados, os métodos 3D de segmentação semântica e classificação são baseados em diferentes representações de dados e podem ser divididos em três categorias: baseado em imagem *multiview*, baseado em voxel e baseado em ponto (HU *et al.*, 2020). Nosso método se enquadra na categoria baseada em pontos, utilizando a nuvem de pontos bruta e sem nenhum método de agrupamento de pontos ou concatenação com imagens. Embora a precisão com dados gerados por desenho assistido por computador (computer-aided design, CAD), como o *dataset* ModalNet40, consigam atingir um alto grau de detalhes, aprender a classificar um conjunto de dados de objetos 3D do mundo real ainda é uma tarefa bem desafiadora.

Na literatura alguns alguns *dataset* se destacam, por exemplo, ShapeNet (WU *et al.*, 2015), ModelNet40 , nuScenes (CAESAR *et al.*, 2020), SUN RGB-D (SONG *et al.*, 2015), Stanford 3D Indoor Scene Dataset (S3DIS) (ARMENI *et al.*,), Berkeley Segmentation Dataset 500 (BSDS500) (ARBELAEZ *et al.*, 2010). Em comum, todos possuem mais de 200 *papers* utilizando seus dados e cada um mais de 6 *benchmarks*. A diferença entre eles, além do método de aquisição, são o número de classes, os tipos de classes, a quantidade de números de pontos dos objetos, ambientes que foram aquisitados entre outros. Na tabela 1 estão organizados trabalhos em relação ao tipo de sensor para aquisição dos dados e o tipo de dados gerados. Nesta pesquisa o *dataset* ModelNet40 é utilizado para comparar os resultados com o *dataset* proposto, aplicando a arquitetura das redes com melhores performance no ModelNet40. Por ser o conjunto de dados mais utilizado para análise de nuvem de pontos, 897 *papers* publicados, diversas categorias de objetos e mais de 12 mil malhas geradas, o ModelNet40 ofereceu mais recursos e foi escolhido.

Tabela 1 – Resumo de trabalhos relacionados aos sensores e tipos de dados.

Tipo de SenSor	**Tipos de Dados**	**Artigos**
Câmeras	Imagens RGB-D	(WANG *et al.*, 2018; KAZHDAN *et al.*, 2003, 2003; WU *et al.*, 2015) (MATURANA; SCHERER, 2015; SU *et al.*, 2015; QI *et al.*, 2016; LI, 2017; SONG *et al.*, 2015)
Cameras/ 3D scans	Imagem RGB-D + Nuvem de Pontos	(CHEN *et al.*, 2016; CHEN *et al.*, 2017; QI *et al.*, 2018; HAO *et al.*, 2018) (BOULCH *et al.*, 2017; PASZKE *et al.*, 2016; RUKHOVICH *et al.*, 2021)
LiDAR(3D CAD)	Dados Sintéticos	(WU *et al.*, 2019; MOHAMMADI *et al.*, 2021; RAN *et al.*, 2022) (ZHANG *et al.*, 2021; REIZENSTEIN *et al.*, 2021)

Ao analisar os resultados do *benchmark*, é importante discutir os modelos de classi-

ficação treinados em dados sintéticos geralmente não generalizam bem para dados do mundo real, como nuvens de pontos reconstruídas a partir de varreduras RGB-D (DAI *et al.*, 2017; HUA *et al.*, 2016), e vice-versa. Além disso, observações no contexto parcial de objetos do mundo real são comuns devido a oclusões e erros de reconstrução, por exemplo, eles podem ser encontrados em detectores de objetos baseados em janelas que auxiliam em aplicações de robótica ou veículos autônomos (SONG; XIAO, 2016). Por fim, as nuvens de pontos de dados sintéticos são fechadas por serem, em sua maioria, geradas por CAD. A utilização de nuvens de pontos para classificação, segmentação de objetos é destaque em diversas pesquisas, novos *datasets*, métodos e arquiteturas foram propostos no decorrer das pesquisas. Esses fatos apresentam a relevância do tema, principalmente no contexto de veículos autônomos. Dessa forma, avaliar as nuances das nuvens de pontos e propor métodos de classificação de objetos 3D são contribuições relevantes para a evolução do tema.

1.2 Objetivos

O principal objetivo deste trabalho é propor uma rede neural artificial, baseada em MLP, para classificação de objetos 3D, com base em nuvens de pontos reais e sintéticas aquisitadas pelo sistema de aquisição móvel desenvolvido.

1.2.1 Objetivos Específicos

Este trabalho possui os seguintes objetivos específico :

- Projetar uma arquitetura, baseada em MLP, para classificar objetos 3D com base em nuvem de pontos;
- Gerar dois *dataset* de nuvens de pontos obtidas com o sensor *LiDAR* no ambiente real e simulado;
- Configurar e analisar diferentes arquiteturas de CNN presentes no estado da arte com objetivo de reconhecimento de objetos em 3D;
- Analisar o desempenho das redes com as nuvens de pontos dos dois *dataset* propostos e com *bechmarker* ModelNet40;
- Propor a etapa de pré-processamento para normalizar e ajustar no plano cartesiano 3D as nuvens de pontos de entrada das CNNs;
- Comparar os resultados das CNNs usando as nuvens de pontos com e sem a etapa de

pré-processamento e avaliar o desempenho de cada configuração utilizando métricas de avaliação, tais como Acurácia, Precisão, Sensibilidade, F1-*Score* e Especificidade, além do tempo de processamento e teste estatístico;

- Identificar o comportamento das redes configuradas utilizando dados sintéticos e dados reais e dimensionar a interferência da quantidade de pontos nas nuvens de pontos na classificação do objeto 3D.

1.3 Produção

A produção científica durante o doutorado resultou na submissão e publicação de artigos científicos a periódicos e congressos. A seguir estão listadas as publicações relacionadas à pesquisa descrita neste documento:

DE SOUSA, Pedro Henrique Feijó et al. **Intelligent 3d objects classification for vehicular ad hoc networks based on lidar and deep learning approaches.** IEEE Transactions on Intelligent Transportation Systems, 2021 (SOUSA *et al.*, 2021).

XU, Yongzhao et al. **Multi-sensor edge computing architecture for identification of failures short-circuits in wind turbine generators.** Applied Soft Computing, v. 101, p. 107053, 2021 (XU *et al.*, 2021b).

REBOUÇAS FILHO, Pedro Pedrosa et al. **A New Strategy for Mobile Robots Localization based on Omnidirectional Sonar Images and Machine Learning.** In: Anais Estendidos da XXXII Conference on Graphics, Patterns and Images. SBC, 2019. p. 168-171 (FILHO *et al.*, 2019).

1.4 Organização do Texto

A organização do texto obedece a sequência cronológica de desenvolvimento da pesquisa. Dessa forma, o texto está organizado da seguinte maneira:

Capítulo 2 são expostas as fundamentações teóricas necessárias para o desenvolvimento do trabalho;

Capítulo 3 é explorada a metodologia para execução dos experimentos e aquisição dos dados.

No Capítulo 4 é apresentado e discutido quais redes obtiveram os melhores resultados na classificação de objetos 3D e o impacto do uso de dados reais e sintéticos, assim como a

interferência da quantidade de pontos nas nuvens de pontos.

Capítulo 5, é exposta a conclusão, baseado nos objetivos idealizados, de qual método é mais indicado na classificação de objetos 3D.

2 FUNDAMENTAÇÃO TEÓRICA

Na seção 2.1 são descritos os detalhes de como as nuvens de pontos são geradas. Em seguida, a seção 2.2 apresenta os principais métodos de aprendizado de máquinas para a classificação de objetos a partir de nuvens de pontos, com destaque para as rede neurais artificial *Multilayer Perceptron* (*MLP*) e *Convolutional Neural Network* (CNN). E por fim, na seção 2.4 são abordados trabalhos destaques na classificação de objetos 3D com base em nuvens de pontos.

2.1 Geração de Nuvem de Pontos

É possível gerar nuvens de pontos a partir de imagens RGB-D e scans 3D. No uso de scans 3D, é necessário o uso de sensores a laser, por exemplo o Lidar. O sensor Lidar emite um feixe luminosos que coleta diversos pontos vizinhos presente em seu campo de visão, produzindo assim uma varredura em um eixo cartesiano. Vetores com os dados de distância são gerados. Com novas leituras, que capturam seções vizinhas entre si e concatenam os vetores de distância gerados aos vetores gerados na varredura passada, é possível assim formar uma nuvem de pontos. Para unir essas varreduras do lidar e formar nuvens de pontos é necessário modificar o processo de varredura por passo angular ou fixar o eixo cartesiano de varredura e deslocar o sensor sobre ele, modo este escolhido para o sistema de aquisição móvel (SAM) e com eixo Y fixado. É importante que o eixo fixado seja diferente do eixo cartesiano da varredura.

Os sensores inerciais são capazes de medir os três eixos, referente ao local do sensor, devido ao uso do acelerômetro e giroscópio que são os instrumentos que, geralmente, geram esse conjunto de informações. O acelerômetro mensura a aceleração do corpo relativa a queda livre, ou seja, informação da orientação no eixo Y. Essa orientação fixa, do eixo Y, permite o enquadramento constante enquanto o Lidar executa a rotação de varredura. O giroscópio é responsável por medir a velocidade angular do corpo em relação a um quadro móvel. Caso o sensor sofra movimento de translação na varredura, os vetores de distância não serão precisos. Algumas limitações estão presentes no IMU devido a ruídos, vieses e não linearidade que possam interferir diretamente na geração dos dados. Nesse contexto, a teoria de fusão sensorial incide para reduzir erros nos dados. Entre as técnicas existentes, o filtro de Kalman se destaca na fusão de dados pois a sua abordagem estocástica inclui características dos ruídos dos sensores, envolvidos no processo, na modelagem do filtro. Porém o filtro de Kalman é orientado para sistemas lineares (LEE *et al.*, 2009; LUO; KAY, 1989). Envolvendo situações não-lineares o filtro

de Kalman foi base e outras versões surgiram, entre elas, a filtro de Kalman estendido (Extended Kalman Filter, EKF)(WAN; MERWE, 2000)), e o filtro de Kalman Unscented (Unscented Kalman Filter, UKF)(WAN; MERWE, 2000).

Munido dos dados dos vetores e com as informações da unidade de medida inercial (inertial measurement unit, IMU) é necessário fundir os dados para formar as coordenadas dos pontos. A fusão de dados é feita combinando a estimação de orientação dos dados da IMU sincronizados com o vetor de varredura laser do LiDAR. A combinação dos dados é feita através do sistema operacional do robô (robot operating system, ROS), que possui bibliotecas para conversão e transformações dos dados do LiDAR. Duas transformações são aplicadas sobre a leitura dos dados, uma de rotação e outra de translação. A transformação de translação é descrita pelas coordenadas cardinais do sensor lidar e a transformação de rotação é descrita por um quatérnion. Com a combinação dos dados pelo ROS, a nuvem de pontos é gerada. O arquivo gerado possui linhas, cada linha representa um ponto, e três colunas que representam as coordenadas x,y,z de cada ponto. No sistema de aquisição móvel (SAM) proposto existem apenas movimentos rotacionais tornando assim importante essa combinação confiável da orientação do sensor e consequentemente na geração dos dados.

2.2 *Rede Neurais Artificiais*

O aprendizado de máquinas (*machine learning*) é uma técnica proposta para resolver tarefas complexas para os humanos. A partir do fornecimento de uma grande quantidade de dados, o algorítimo de aprendizado explora e busca modelos que atinjam um determinado objetivo. Redes neurais são técnicas computacionais que apresentam modelos matemáticos baseado na estrutura neural interconectados que funcionam análogos aos neurônios do cérebro humano. Usando algoritmos é capaz de reconhecer padrões e encontrar correlações em dados brutos, agrupar e classificar dados e com o tempo aprender e melhorar continuamente (KOVÁCS, 2002).

2.2.1 Multilayer Perceptron

O Perceptron Multicamadas (multilayer perceptron, MLP), é um algoritmo de aprendizado de máquina supervisionado utilizado em aplicações de classificação e de regressão. Este método utiliza múltiplos modelo do Perceptron simples interconectados, como ilustra a figura

2, caracterizando nós, formando uma rede neural artificial para a solução de problemas não lineares (HAYKIN, 2001).

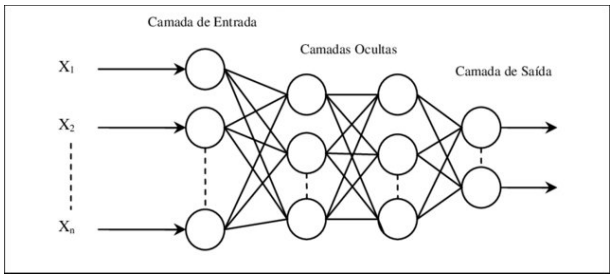


Figura 2 – Perceptron multicamadas

O Perceptron simples foi proposto por Rosenblatt (1958) que reproduz uma combinação linear das suas entradas com pesos específicos. O Perceptron é um algoritmo para solucionar problemas lineares e sua saída pode ser expressa como visto na Equação 2.1,

$$\hat{y} = \phi\left(\sum_{i=1}^{p} x_i w_i + b\right) \tag{2.1}$$

em que ϕ é a função de ativação, x_i é o vetor de entrada e w_i é o vetor de pesos que ponderam a entrada, ambos de tamanho p. O viés b, assim como o vetor w_i são parâmetros que se ajustam durante o treinamento do Perceptron. Este treinamento é realizado de maneira iterativa e dado pela Equação 2.2,

$$w_i(t+1) = w_i(t) + \eta(y - \hat{y}(t))x_i, \ \forall \ i = 1, ..., p \tag{2.2}$$

na qual t é a iteração do treinamento, y é classe da amostra e $\hat{y}(t)$ é a predição realizada para esta amostra naquela iteração. Além disso, o η é taxa de aprendizagem, responsável por ponderar o erro de predição. Sabendo que o Perceptron não era capaz de solucionar problemas não lineares, o MLP foi proposto como um arranjo de Perceptrons, agora chamados de neurônios, em camadas conectadas adiante, as quais utilizam uma função de ativação sigmoide, responsável por introduzir a não linearidade no vetor de atributos. Cada neurônio tem sua saída a depender da camada em que está inserido, como visto na Equação 2.3,

$$\hat{y}_{l_k} = \begin{cases} \phi_{0_k}\left(\sum_{i=1}^{p} x_i w_{0_{k_i}} + b_{0_k}\right) & se \ \ l = 0 \\ \phi_{l_k}\left(\sum_{i=1}^{D} \hat{y}_{l-1} w_{l_{k_i}} + b_{l_k}\right) & se \ \ l \neq 0 \end{cases} \tag{2.3}$$

em que l é o número da camada e k é a posição do neurônio nessa camada, sendo x um vetor de atributos de dimensão p, quando $l = 0$ o número de neurônios é igual ao tamanho de x e quando $l \neq 0$ o número de neurônios é D. Ainda, é ϕ é a função de ativação não linear. Durante o

treinamento, o MLP utiliza um método de otimização chamado *Backpropagation* (RUMELHART *et al.*, 1995) que via ajustar os pesos do modelo a cada iteração de maneira a reduzir o erro de predição. Para tal, é necessário usar a Equação 2.4 para que estes pesos possam ser utilizados,

$$\hat{w}_{l_{k_i}}(t+1) = \begin{cases} w_{0_{k_i}}(t) + \eta \cdot \delta_{l_i}(t) \cdot x_k(t) & se\ \ l = 0 \\ w_{l_{k_i}}(t) + \eta \cdot \delta_{(l+1)_i}(t) \cdot \hat{y}_{l_k}(t) & se\ \ l \neq 0 \end{cases} \tag{2.4}$$

na qual $\delta_{l_k}(t)$ é o gradiente local na camada l-ésima do k-ésimo neurônio, e pode ser expresso pela Equação 2.5:

$$\delta_{l_k} = \begin{cases} \phi'_{l_k}[u_{l_k}(t)] \frac{1}{2M} \sum_{m=1}^{M} [y_{l_m} - \hat{y}_{l_m}(t)]^2 & se\ \ l = z \\ \phi'_{l_k}[u_{l_k}(t)] \sum_{j=1}^{J} w_{l_{k_i}}(t) - \delta_{(l+1)_j}(t) & se\ \ l \neq 0 \end{cases} \tag{2.5}$$

em que z é a camada de saída da rede, a qual tem M neurônios e l é qualquer outra camada com J neurônios. Além disso, ϕ' é a derivada da função de ativação utilizada, dentre sigmoide e tangente hiperbólica, a derivada será como na Equação 2.6,

$$\phi'_{l_k}[u_{l_k}(t)] = \begin{cases} y_{l_k}(t)[1 - y_{l_k}(t)] & se\ \ logistica\ ou\ sigmóide \\ \frac{1 - y^2_{l_k}(t)}{2} & se\ \ tangente\ hiperbólica \end{cases} \tag{2.6}$$

De acordo com as limitações deste tipo de modelo e devido a necessidade de se processar dados em dimensões maiores, como imagens, novas redes neurais foram propostas, tais como as *Convolutional Neural Networks* (CNN). Na próxima Seção estarão dispostas algumas definições importantes sobre as CNNs.

2.2.2 Convolutional Neural Networks

A rede neural convolucional é uma rede neural artificial utilizada para aprendizado profundo. O algoritmo capta um dado de entrada, atribui importância a características, relacionando pesos às características,

métodos, operações e funções de ativação que são utilizados na implementação, treinamento e validação de CNN estão descritas nesta seção com o objetivo de auxiliar no entendimento sobre o funcionamento das redes deste tipo.

2.2.2.1 Convolução

Convolução é a aplicação de um filtro com objetivo de extrair o maior número de informações relevantes da imagens ou dado de entrada ou de camada anteriores (ALBAWI *et*

al., 2017). De acordo com a estrutura da CNN, são aplicados diferentes filtros com objetivo de extrair características da borda, contraste, textura gradiente e outros, quando aplicados em imagens.

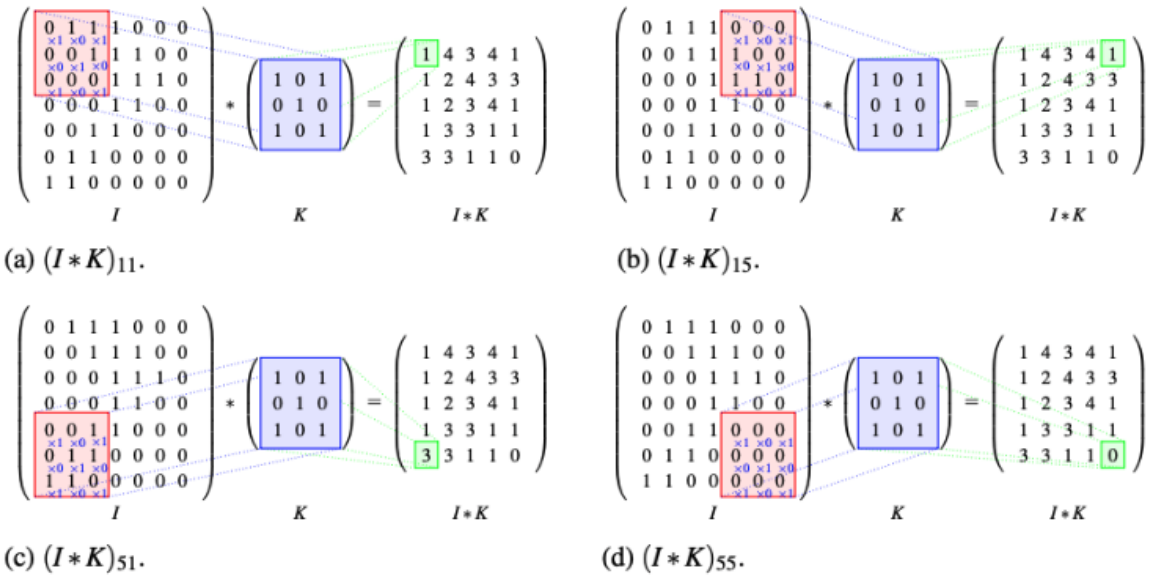
A convolução do filtro K em uma imagem I, o filtro percorre toda imagem. E quando existir sobreposição entre ambos, haverá um somatório dos produtos de elementos por elemento, de acordo com a equação 2.7.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.7}$$

onde S é o mapa de características originado pela convolução do filtro K na imagem I. Na figura 2.7, é representada o fluxo das quatro iterações da convolução, no exemplo um filtro 3x3 aplicado a uma matriz 7x7, sem *padding* e com *stride*, passo com o qual o filtro se desloca, de um. A equação 2.8 representa a ordem da matriz de saída, G.

$$G = \frac{N-F}{S} + 1 \tag{2.8}$$

na qual N é a ordem da matriz, F é a ordem do filtro da convolução e S é o tamanho do *stride*. Na Figura 2a observa-se a iteração que resulta no elemento $(1,1)$ da matriz resultante, através do somatório dos produtos de elemento por elemento do filtro com a matriz de entrada. Já na Figura 2b apresenta a iteração que resulta no elemento $(1,5)$ da matriz de saída $I * K$. Enquanto as Figuras 2c e 2d ilustram a mesma operação para os elementos $(5,1)$ e $(5,5)$ de $I * K$, respectivamente.



(a) $(I * K)_{11}$. (b) $(I * K)_{15}$.

(c) $(I * K)_{51}$. (d) $(I * K)_{55}$.

Figura 3 – Sequência de interação da operação de Convolução.

Fonte: Elaborada pelo autor.

2.2.2.2 Batch Normalization

Uma limitação presente em treinamentos usando redes neurais de aprendizado profundo é a distribuição das entradas de cada camada muda durante o treinamento à medida que os parâmetros das camadas anteriores alteram. Esse fato impacta na velocidade do treinamento (IOFFE; SZEGEDY, 2015).

Proposta por Ioffe e Szegedy (2015), a *Batch Normalization* é um método de otimização indicado para acelerar e estabilizar treinamentos de redes neurais profundas (LUO *et al.*, 2018). Neste processo ocorre uma normalização das entradas de uma camada por lote em que a rede é dividida previamente. Esta normalização é feita pela média e desvio padrão, como descrito na Equação 2.9:

$$H' = \frac{H - \mu}{\sigma} \tag{2.9}$$

Com isso, ao estabilizar o processo de aprendizagem da rede neural, a *Batch Normalization*, reduz o número de épocas necessárias para a etapa de treinamento, uma vez que os pesos são atualizados ao final do processamento de cada lote e não da rede completa como é feito tradicionalmente.

2.2.2.3 Pooling

Em algumas situações a dimensão dos dados de entrada na CNN pode interferir em seu funcionamento. Para contornar esse problema, é aplicado a operação de *pooling* na saída da camada convolucional. Esse método usa campos receptivos sobre a matriz de entrada para reduzir a quantidade de informação, reduzindo assim a dimensão desses dados (SCHERER *et al.*, 2010).

Na figura 4 é ilustrada as duas formas mais usadas de pooling, *max pooling* e *average pooling*, onde os filtros receptivos possuem tamanho 2x2 e varrem a matriz 4x4. No fim do processo, existe a redução da dimensionalidade da matriz. No exemplo, a matriz resultante é de 2x2 e os valores são de acordo com a operação de *pooling* aplicada.

A operação de *max pooling* escolhe o maior valor do campo receptivo e seu valor é atribuído à matriz de saída. Na operação de *average pooling* o valor atribuído à matriz de saída é resultante do cálculo da média aritmética dos valores presentes no campo receptivo (GOODFELLOW *et al.*, 2016).

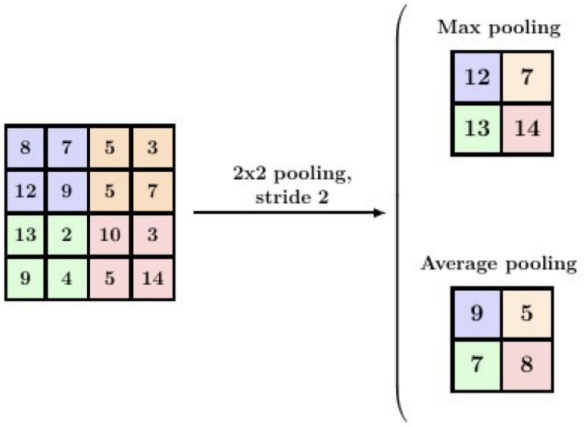


Figura 4 – Ilustração da operação de *Pooling*.
Fonte: Elaborada pelo autor.

2.2.2.4 Camada Densa

Camada densa ou totalmente conectada é um conjunto de neurônios que operam propagando adiante as informações que pondera na sua entrada. Quanto posicionada na saída da rede, ela será a camada responsável pela classificação do vetor de características ao final das operações de convolução e *pooling* dentro de uma rede neural (LECUN *et al.*, 1989). Ao longo de uma rede neural os dados de entrada são filtrados através das convoluções e tem a dimensão reduzida após a operação de *pooling*, ao final, no topo da rede, antes da camada densa, apenas um vetor de características restará, e a partir dele ocorrerá a predição, dentre as classes existentes no conjunto de treino.

2.2.2.5 Dropout

Proposto por Srivastava *et al.* (2014), o dropout é uma técnica utilizada na regularização de uma rede neural. Especificamente, essa técnica é aplicada para evitar o sobreajuste do aprendizado de uma rede neural no subconjunto de treinamento, também chamado de *overfitting*. A figura 5 ilustra o funcionamento do método.

O método desativa uma taxa fixa de neurônios em cada camada aplicada. No exemplo acima, dos cinco neurônios dois estão desativados, representados pela cor vermelha, o que representa 40% dos neurônios. Isso significa que 40% dos neurônios não contribuem na alimentação direta, assim como não têm pesos atualizados pelo algoritmo de *backpropagation*.

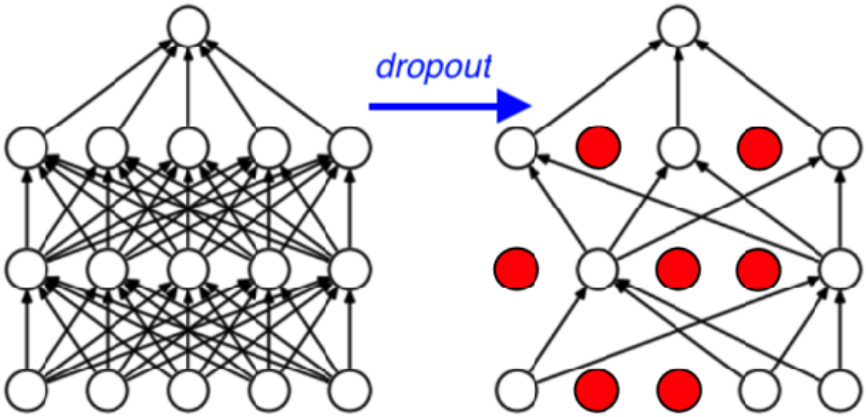


Figura 5 – Ilustração da aplicação do método do dropout.
Fonte: Elaborada pelo Autor

2.2.2.6 *Funções de Ativação*

A função de ativação é o elemento de tomada de decisão que define o limite de decisão no espaço de entrada, definindo um limite no campo local induzido. Sem uma função de ativação, o sinal de saída se torna uma função linear simples. As arquiteturas de aprendizagem profunda usam funções de ativação, para realizar diversos cálculos entre as camadas ocultas e as camadas de saída da arquitetura (SHEN *et al.*, 2017). A função de ativação pode ser linear, sigmoide, softmax, ReLU e Leaky ReLU. Na rede Lidar3DNet é utilizada as funções de ativação ReLU nas camadas intermediárias e a função softmax na camada de saída.

A *relu*, ou unidade linear retificada é uma das funções de ativação mais utilizadas atualmente. É uma função não linear (AGARAP, 2018) representada pela Equação 2.10:

$$f(x) = max(0,x) \tag{2.10}$$

E ao observar a figura 6 percebe-se que para qualquer valor menor que 0, a saída da função é zerada, não ativando o neurônio, desta forma, em cada iteração de treinamento, dificilmente todos os neurônios serão ativados simultaneamente. Essa característica influencia no custo computacional durante o tempo de treino. Por contemplar problemas não lineares e por um custo operacional menor que as demais função de ativação ReLu é utilizada na ativação das camadas intermediárias da Lidar3DNet.

A função de ativação softmax é responsável por transformar o espaço do vetor de saída da rede neural para um espaço probabilístico, onde os valores são apresentados entre 0 e 1, e a soma dos valores resulta em 1. Normalmente, a *softmax* é usada em camadas de classificação

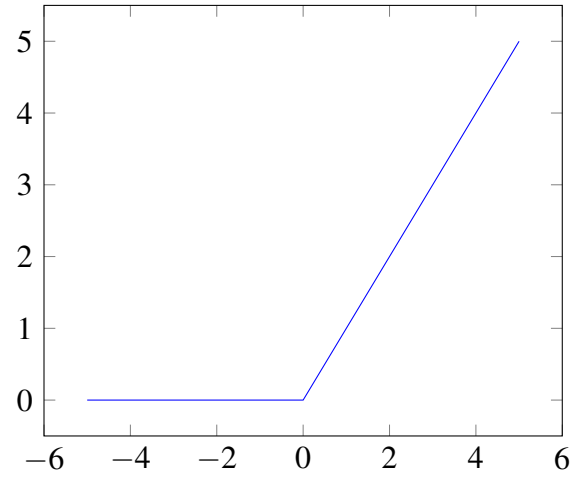


Figura 6 – Gráfico da funcão de ativação *relu*.
Fonte: Elaborada pelo autor.

das redes neurais e pode ser descrita pela Equação 2.11:

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{2.11}$$

em que $i = 1, ..., K$ e $z_i = (z_1, ..., z_k) \in \mathrm{R}^K$, sendo K a quantidade de classes do problema e z_i é a saída do neurônio i da última camada da rede. Indicada para problemas não binários e aplicada na saída de problemas de classificação multi classe, a função *softmax* é aplicada na camada de saída da rede Lidar3DNet.

2.3 Descrição do Sensor LiDAR

Os primeiros passos para os veículos autônomos foram iniciados a mais de uma década. Os sensores de detecção de luz e varredura (LiDAR) surgiu inicialmente para levantamento topográfico mas também contribuiu para a evolução da navegação de veículos autônomos e aplicações industriais, sendo assim um instrumento de destaque e robustez (RORIZ *et al.*, 2021).

O sensor LiDAR da marca SICK da família LMS5xx, utilizado nesta pesquisa, operam com luz infravermelha a 905nm dessa forma conseguem detectar características do ambiente e dos objetos com detalhe(SICK, 2015). Esses sensores são um sistema de medição de distância por luz emissão com base no princípio de tempo de voo de emissões. Este sensor é considerado um dos mais apropriados até para aplicações que exigem precisão nos dados espaciais para a percepção de objetos e superfícies (KOTB *et al.*, 2018). Motivado por sua relevância na área, este trabalho utilizou o LiDAR no sistema móvel de aquisição (SAM) para geração de nuvens pontos dos objetos.

O funcionamento do sensor LiDAR assemelha-se ao de um sensor de distância a laser convencional. No entanto, os sensores LiDAR adquirem diferentes feixes de luz que coletam vários pontos em seu campo de visão digitalizando um eixo cartesiano desejado. A formação de nuvens de pontos 3D requer que a varredura de cada sensor seja associada à orientação, rotação e matriz de transformação fornecido por uma primeira unidade de medição ou um Computador Numérico Controlado.

O dataset desenvolvido nesta pesquisa utilizou dois tipos de LiDAR, são eles:
I) LMS 291: Este sensor foi utilizado no simulador *Webots*. É um sensor que pode ser usado em ambientes externos. O sensor tem um campo de visão de 180 ○ e um alcance de até 80 metros em condições ideais. A configuração da resolução angular pode ser ajustada em três níveis: 0,25 ○ , 0,5 ○ e 1 ○.
II) LMS 511: Este sensor foi usado em ambientereal Ambiente. É um sensor adequado para uso externo, que tem um campo de visão de 190 ○ , um alcance de até 80 metros em condições ideais e um ângulo de abertura de 190 ○ . A resolução angular pode ser ajustada em três níveis: 0,25 ○ , 0,5 ○ , e 1 ○ . Devido às suas características, o LMS 511 é frequentemente utilizado em aplicações industriais.

2.4 Estado da Arte

Nesta seção serão explorados trabalhos presentes na literatura que propuseram métodos e contribuíram para a tarefa de classificação de objetos com nuvens de pontos. Informações sobre suas configurações, *dataset*, métodos propostos e tecnologias utilizadas para preparar e executar o ambiente serão destacadas com o objetivo de detalhar e comparar os métodos. A escolha desses trabalhos é baseada na métrica de acurácia

2.4.1 Relation-Shape-CNN

A rede RS-CNN (LIU *et al.*, 2019) utiliza a base da rede neural convolucional e se estende para a análise de configurações irregulares para análise de nuvem de pontos. O método aprende através da restrição de topologia geométrica entre os pontos. Em particular, um conjunto local de pontos possui um peso convolucional resultante de uma expressão de alto nível a partir de dados geométricos predefinidos. De modo que a RS-CNN define uma relação hierárquica para aprendizado contextual da forma contextual de uma nuvem de pontos. O autor propõe a

técnica de aumento de dados e uso de mais filtros convolucionais para reduzir o impacto da limitação presente na CNN quando aplicada em nuvem de pontos. O peso convencional de cada região aprende essa relação baseado em propriedades geométricas predefinidas. Um valor x, denominado centroide, é escolhido de forma aleatória em cada região, dessa forma o aprendizado do peso convolucional é induzido a gerar expressões que relacionam a centroide X com seus N pontos vizinhos de acordo com as restrições geométricas predefinidas.

2.4.2 LDGCNN

O trabalho propõe uma metodologia chamada de *Dual-Scale* com arquiteturas padrões baseada em nuvens de pontos locais e *voxel*. Com o objetivo de relacionar estrutura global com partes específicas da nuvem de pontos, o processamento global é realizado por *voxels* e identifica relações entre as estruturas com um alcance maior (global) e o processamento local é realizado nos pontos da região para identificar características mais específicas. O tempo de processamento da metodologia proposta e a duplicação de informações entre os dois métodos não foram avaliadas no estudo (ZHANG *et al.*, 2021).

2.4.3 PointNet

A PointNet (QI *et al.*, 2017) é uma rede neural projetada para receber de forma direta os dados das nuvens de pontos sem nenhum agrupamento ou coleção de imagens. Proposta para classificação de objetos, segmentação de regiões determinadas na nuvem de pontos e análise semântica da cena. Dessa forma a rede é treinada com parâmetros globais e a nuvem de pontos apresenta as coordenadas x,y, e z de cada ponto, possibilitando desprezar algum dado importante de alguma região. Além disso, a rede neural também pode ser treinada com parâmetros locais, porém informações extras devem ser adicionadas além das coordenadas dos pontos. O alto volume de informações e o processamento total da nuvem de pontos podem comprometer o tempo de desempenho da rede PointNet nas tarefas de classificação e segmentação. No estudo, o autor citou métricas de tempo de processamento empíricas.

2.4.4 PVT

A rede Point-Voxel Transformer (PVT) é desenvolvida para contribuir na problemática de custo de processamento de nuvens de pontos. Para isso o autor propõe dois módulos o

Sparse Window Attention (SWA) e módulo de atenção relativa (RA), sua combinação resulta na arquitetura neural PVT para análise de nuvens de pontos. O módulo SWA captura informações locais de *voxels* não vazios, contornando a complexidade computacional envolvida na resolução dos *voxels*. Considerando o contexto global da nuvem de pontos, o módulo RA refinar as estruturas com transformações rígidas de objetos. A redundância de informações ou perda de informações no processamento dos dois módulos não foram claras na escrita do autor (ZHANG *et al.*, 2022).

2.4.5 CurveNet

A rede CurveNet propõe um método que agrega curvas hipotéticas na nuvem de pontos dos objetos, necessitando assim apenas informações x,y,z da nuvem de pontos. Essa sequência de pontos conectados desempenham um papel semelhante como uma caminhada guiada interligando os pontos e em seguida o caminho de volta é feito para aumentar as características pontuais do caminho. Para isso propôs um novo operador de agrupamento. O melhor resultado, aplicado no *dataset*, 94,2% com tempo de processamento de 146 ms e foram programadas 200 épocas para o treinamento com taxa de aprendizado inicial de 0.05 (XIANG *et al.*, 2021).

2.4.6 SimpleView

Em sua pesquisa, consideraram fatores auxiliares para melhorar o desempenho, como diferentes esquemas de avaliação, estratégias de aumento de dados e funções de perda, que são independentes da arquitetura do modelo de aprendizado. Dessa forma o SimpleView tem seu método baseado em projeções e o que acarreta uma melhor generalização entre os conjuntos de dados. (GOYAL *et al.*, 2021)

2.4.7 PAConv

A Position Adaptive Convolution (PAConv) utiliza convolução genérica para processamento em nuvem de pontos 3D. Um kernel de convolução é projetado e monta dinamicamente matrizes de peso básico que são salvos em um banco. Os coeficientes das matrizes são auto adaptativos e são orientados baseado na posição através do ScorteNet. O kernel é adaptado de acordo com os dados, dessa forma sua aplicação é indicada a nuvens de pontos irregulares e

desordenadas. O tempo de aprendizagem é reduzido devido ao processo de relação de matrizes e não de processamento direto com os pontos e suas posições, porém seu nível de detalhamento em regiões específicas podem ser reduzidas e intervir na classificação do objeto 3D. Considerada uma rede simples, pelo autor, porque esse método é combinado com pipelines baseados em MLP sem nenhuma configuração de rede. (XU *et al.*, 2021a)

2.4.8 GBNet

Com uma abordagem de classificar nuvens de pontos de acordo com as características geométricas das nuvens de pontos a rede GBNet adiciona informações geométricas dos pontos em espaço 3D de baixo nível e em locais de alto nível é aplicado estruturadas de aprendizagem, baseada em CNN, para aprender o contexto geométrico local. No entanto, esse método pode gerar redundâncias no aprendizado da rede sabendo que em nuvens de pontos reais os pontos não são bem distribuídos e a quantidade de pontos, em determinada região, pode ser insuficiente para a extração da característica geométrica da nuvem. (QIU *et al.*, 2021)

2.4.9 3DMedPT

Na literatura, diversas redes neurais utilizam informações das nuvens de pontos para segmentar, classificar e soluções aplicadas ao auxílio de veículos autônomos. A rede neural transformador de pontos 3D (*3D medical point Transformer*, 3DMedPT) propõe o uso de nuvens de pontos na área médica para examinar estruturas biológicas complexas e assim auxiliar na detecção e tratamento de doenças. Com foco na necessidade de otimizar as informações contextuais e resumir respostas em regiões específicas, a rede propõe soluções locais e globais na nuvem de pontos. Uma limitação encontrada pelos autores é o escasso número de amostras de treinamento. Por esse motivo, utilizaram técnicas de incorporar posições dos pontos para melhor a geometria Local e assim enriquecer o conhecimento local e global da estrutura da nuvem de pontos. Como em outros trabalhos, o uso de *data augmentation* pode provocar redundâncias nas informações locais da nuvem pontos e assim interferir no desempenho da rede neural (YU *et al.*, 2021).

3 METODOLOGIA

Nesta seção é abordada a metodologia para o desenvolvimento da pesquisa. A seção 3.1 é ilustrado e explicado o fluxo da metodologia proposta para classificação de objetos 3D. Em seguida, na seção **??** é destacada as configurações da arquitetura da rede neural. Na seção 3.3 e subseções 3.3.1 é exposta a forma de aquisição do *dataset* e sua estruturação baseado em trabalhos da literatura. Em seguida, na seção 3.5.1 e 3.5.2 são exploradas as etapas de pré-processamento propostas. E por fim, nas seções 3.7 e 3.8 são abordadas as métricas de avaliação utilizadas nas discussões dos resultados.

3.1 Fluxo do Método de Classificação de Objetos 3D.

O fluxo da metodologia adotada para classificação de objetos 3D está ilustrada na figura 7. A primeira etapa é a aquisição e organização dos dois *dataset* de nuvens de pontos, propostos neste trabalho. Um aquisitado em ambiente real e o outro em ambiente virtual, porém ambos contém as mesmas classe e em diferentes perspectivas que serão explicadas com maior profundidade na seção 3.3. Na etapa 2, é explorada as etapas de normalização e transformação da nuvem de pontos que formam o pré-processamento proposto nas nuvens de pontos dos objetos de entrada da rede e em seguida a arquitetura da rede *Lidar3DNet*. A topologia da rede proposta *Lidar3DNet* é composta pela biblioteca Tensorflow versão 1.14 e Keras para implementação. Com esta biblioteca, é construída uma rede usando uma camada de entrada, camadas intermediárias e camada de saída. Em segundo lugar, definimos as camadas densas usando 512 neurônios ocultos e 256 neurônios ocultos na segunda camada, como ilustra a figura 7. O dropout de 20% é utilizado para reduzir o overfitting no treinamento, desabilitando parte dos neurônios nas camadas intermediárias, descrito na seção 2.2.2.5. A função de ativação utilizada foi Relu nas camadas intermediárias e softmax na camada de saída, abordadas na seção 2.2.2.6.

Ainda na etapa 2, após a etapa de normalização, é ilustrado a variação do número de pontos do objeto e sua respectiva figura. Essa variação será tema de argumento na discussão **??**. Na terceira etapa são gerados os resultados da saída da rede, baseado nas métricas expostas nas seções 3.8. Em seguida, na etapa 4, são realizadas comparações entre os resultados com e sem pré-processamento das nuvens pontos e dentro dessa comparação são realizados testes estatísticos, que são detalhados na seção 3.7.

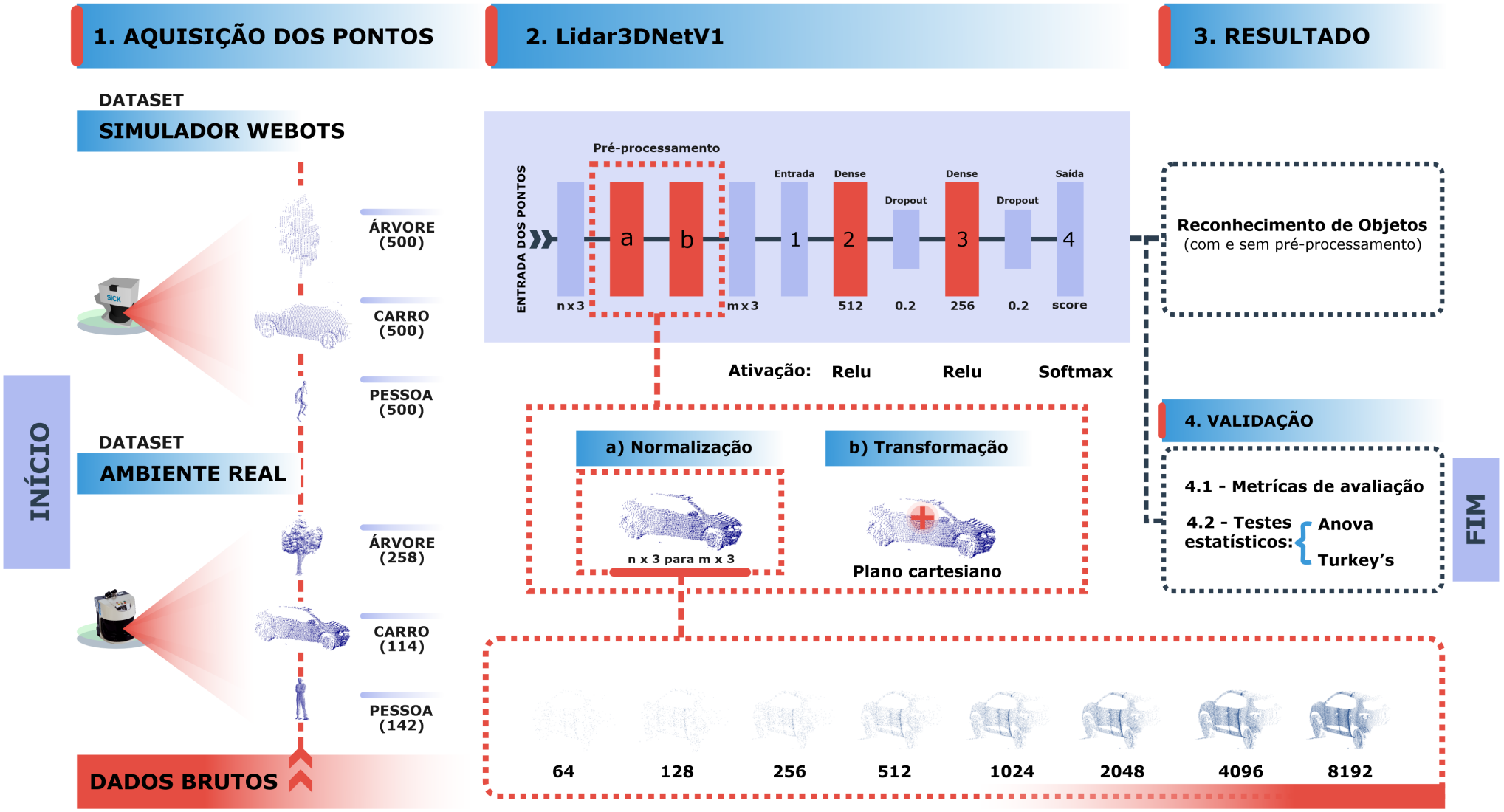


Figura 7 – Fluxo da metodologia proposta.

Fonte: Elaborada pelo autor.

3.2 Lidar3DNet

A rede Lidar3DNet é projetada para classificar nuvens de pontos com a arquitetura baseada no perceptron multicamadas, MLP. Ao contrário das redes neurais artificiais do estado da artes que usam métodos baseados em redes neurais convolucionais que, por sua vez, utilizam uma série de operações e funções para classificação e assim necessitam de alto poder de processamento. Essa abordagem com MLP tem como objetivo propor uma arquitetura menos complexa e assim reduzir o tempo de processamento de classificação das nuvens de pontos e com relevantes taxas de acertos, contribuindo para o bom desempenho em aplicações em sistemas embarcados.

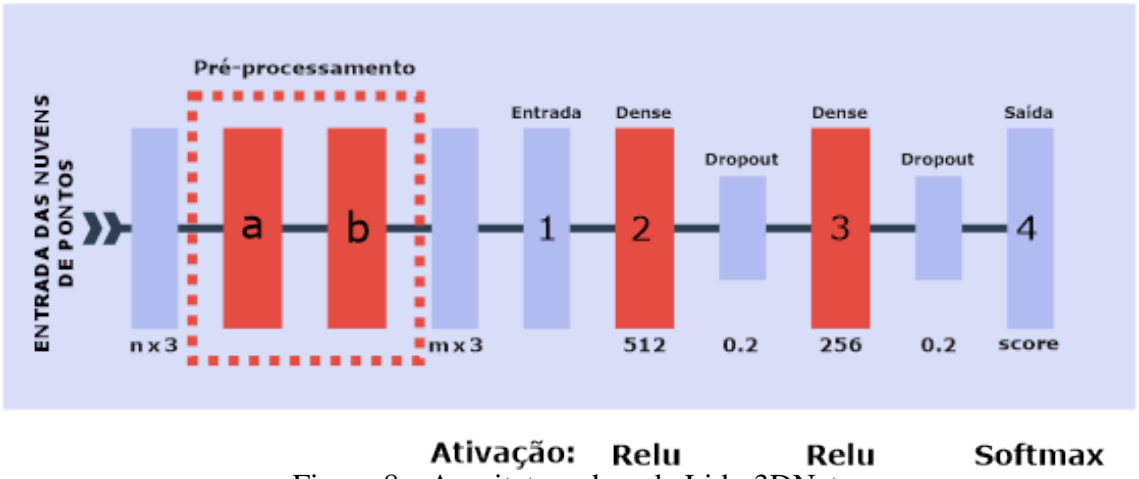


Figura 8 – Arquitetura da rede Lidar3DNet.
Fonte: Elaborada pelo autor.

Usamos a biblioteca Tensorflow versão 1.14 e Keras para implementar a topologia Lidar3DNet. Com esta biblioteca, construímos uma rede usando uma camada de entrada, camadas intermediárias e camada de saída. Em segundo lugar, definimos as camadas densas usando 512 neurônios ocultos e 256 neurônios ocultos na segunda camada. O dropout de 20% é utilizado para reduzir o overfitting no treinamento, desabilitando parte dos neurônios nas camadas intermediárias, como exposta na 2.2.2.5. A função de ativação utilizada foi *relu* nas camadas intermediárias e *softmax* na camada de saída, como ilustra a etapa 2 da figura 7. A definição do número de neurônios é motivada pela quantidade grande de pontos na camada de entrada, e ao longo da rede a quantidade de *features* reduz, consequentemente a camada seguinte reduz o número de neurônios, e por fim, formar o vetor final para classificação.

A nuvem de pontos do objeto 3D bruta contém três colunas com os valores das coordenadas x, y e z. Cada linha do arquivo da nuvem de pontos é a coordenada de um ponto. A primeira etapa da arquitetura da rede Lidar3DNet é normalizar as nuvens de pontos dos objetos

para 2048 pontos e em seguida centralizar a nuvem de pontos para origem.

Para isso, a nuvens de pontos é achatada, onde todas as informações dos pontos estão presente num único vetor, para isso é realizado o *flatten*. Onde um objeto 3D "R"é representado por [[x_1, y_1, z_1], [x_2, y_2, z_2] ... [x_n, y_n, z_n]], onde a coordenada de ponto representa o vetor do ponto. E após o achatamento o vetor "R"é representado por um único vetor, da seguinte forma por [x_1, y_1, z_1, x_2, y_2, z_2 ... x_n, y_n, z_n], onde "n"é o número da linha da coordenado do ponto. Após a etapa de *flatten*, é identificado a quantidade de pontos e aplicado uma interpolação baseada no vizinha mais próximo. Dessa forma é possível redimensionar para mais ou menos a quantidade de pontos do objeto 3D e assim atingir a quantidade de 2048 pontos.

Após a normalização para 2048 pontos, a nuvem de pontos do objeto 3D é ajustada para a origem do plano cartesiano 3D, através da trnaformação abaixo.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t^x \\ 0 & 1 & 0 & t^y \\ 0 & 0 & 1 & t^z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

Onde x', y', z' são as coordenadas dos pontos do objeto 3D ajustados para a origem, "t"é a medida de translação nos eixos. Após os métodos de *flatten*, normalização e ajuste da origem a nuvem de pontos de cada objeto 3D é inserida com na rede Lidar3DNet.

Na figura 9, é apresentada o fluxo com a manipulação das nuvens de pontos como um vetor. Na camada de entrada a nuvens de pontos com 2048 e 3 classes é apresentada como um vetor de entrada . Em seguida, uma camada de *flatten* achata a matriz pra um vetor, seguida da camada *dense* com função de ativação *relu* e 512 neurônios. Após a segunda camada o *dropout* de 20% é inclusa. Na sequência, outra camada *dense* com 256 neurônicos é adicionada com função de ativação *relu* e com *dropout* de 20% novamente. Uma camada dense com 3 neurônios e função de ativação *softmax*, configura a saída da rede.

A rede Lidar3DNet é baseado na arquitetura de uma MLP capaz de solucionar problemas não lineares. A saída de cada neurônio pode ser expressa pela Equação 3.2. A saída de cada neurônio é realizado de acordo com a camada.

$$\hat{y}_{l_k} = \phi_{l_k} \left(\sum_{i=1}^{D} \hat{y}_{l-1} w_{l_{k_i}} + b_{l_k} \right) \tag{3.2}$$

onde l é o número da camada, no caso duas camadas, e k é a posição do neurônio nessa camada, sendo x o vetor de atributos de cada nuvem de ponto de dimensão p. A equação 3.3, representa

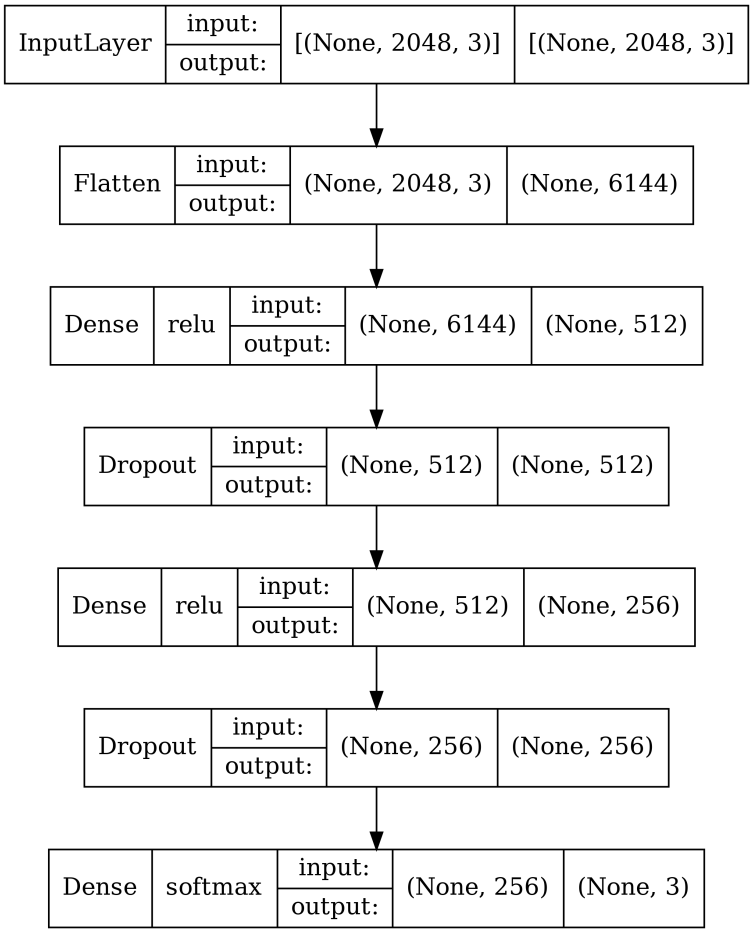


Figura 9 – Modelo da rede Lidar3DNet.

Fonte: Elaborada pelo autor.

matematicamente o ajuste dos pesos do modelo a cada iteração, de maneira a reduzir o erro de predição da classe do objeto 3D.

$$\hat{w}_{l_{k_i}}(t+1) = w_{l_{k_i}}(t) + \eta \cdot \delta_{(l+1)_i}(t) \cdot \hat{y}_{l_k}(t) \tag{3.3}$$

onde o gradiente local ($\delta_{l_k}(t)$) na camada l-ésima do k-ésimo neurônio e pode ser expresso pela Equação 3.4:

$$\delta_{l_k} = \phi'_{l_k}[u_{l_k}(t)] \sum_{j=1}^{J} w_{l_{k_i}}(t) - \delta_{(l+1)_j}(t) \tag{3.4}$$

onde z é a camada de saída da rede com três neurônios. Além disso, ϕ' é a função de ativação utilizada. Nas camadas intermediárias com 512 e 256 neurônios é utilizado a função de ativação *relu*, expressa na equação 3.5. No camada de saída a função de ativação utilizado é a *softmax*, expressa na equação 3.6.

$$f(x) = max(0,x) \tag{3.5}$$

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{3.6}$$

onde $i = 1, ..., K$ e $z_i = (z_1, ..., z_k) \in \mathrm{R}^K$, sendo K a quantidade de classes do problema, nos testes são três classes do *dataset* proposto e 40 classes no caso do *dataset* ModelNet40 e z_i é a saída do neurônio i da última camada da rede.

3.3 Descrição do Dataset

Na literatura existem alguns *dataset* de nuvem de pontos disponíveis, por exemplo ModelNet (WU *et al.*, 2015), ShapeNet (CHANG *et al.*, 2015), ScanNet (DAI *et al.*, 2017) e Semantic3D (HACKEL *et al.*, 2017). Suas formações são realizadas por objetos na forma de nuvens de pontos geradas em ambientes reais ou em ambientes simulados, utilizando softwares específicos. O *dataset* utilizado neste trabalho é formado por nuvens de pontos de objetos aquisitados a partir de um sistema móvel de aquisição (SAM) baseado no sensor *Light Detection and Ranging* (LIDAR) e do simulador Webots (MICHEL, 2004). Os dados reais foram aquisitados em ambiente real por meio de um sistema de aquisição móvel (SAM), representado na figura 10 e os dados sintéticos gerados no software Webots que é um simulador de robô 3D. A geração do *dataset* é uma das contribuições deste trabalho. As aquisições foram realizadas na área externa da Universidade Federal do Ceará no campus de Fortaleza e no ambiente interno do laboratório de processamento de imagens, sinais e computação aplicada(LAPISCO) localizado no Instituto Federal do Ceará de Educação, Ciência e Tecnologia do Ceará, campus Fortaleza.

Figura 10 – Estrutura do sistema de aquisição móvel(SAM).

Fonte: Elaborada pelo autor

Tabela 2 – Número de amostra por classe

Classe	***Dataset***	
	Real	**Sintético**
Carro	114	500
Pessoa	142	500
Árvore	258	500

Fonte: Elaborada pelo autor.

A construção de um *dataset* contribui com a sociedade científica tanto com a aquisição de pontos utilizando objetos reais quanto com objetos sintéticos. A tabela 2 apresenta a distribuição das amostras dos *datasets* propostos. O total de dados sintéticos são de 1500 amostras distribuídas de forma igual entre as classes carro, pessoa e árvore. Nos dados reais, o total de amostras são 484 e estão divididas em números diferentes de amostras: 258 árvores, 114 carros e 142 pessoas em várias poses. Além do número relevante de amostras por classe, existe também uma variação na forma das nuvens de pontos dentro das classes. A figura 11 apresenta as imagens das amostras e a variação no formato das nuvens de pontos reais e sintéticas. A classe carro, por exemplo, apresenta três perspectivas do carro: lateral, frontal e traseira. E essa variação no formato da nuvem de pontos está presente nos dois *dataset* propostos. Na classe de pessoas, as variações das amostras incluem pessoas sentadas, em movimento, em pé e de perfil. As amostras de carros também se alternam nas perspectivas: laterais, frontais e posteriores. A variação do conjunto das árvores é mais significativa do que em outras classes, uma vez que capturamos diferentes tipos de árvores. A segmentação da nuvem de pontes de cada objetos da nuvens de pontos bruta gerada pelo SAM foi realizada de forma manual através do *software CloudCompare*.

A variação dos dados contribui de forma significativa para o bom desempenho do classificador em situações reais, por exemplo na navegação de veículos em ruas e avenidas dentro da cidade. A escolha das classes carro, árvore e pessoa obedeceu o critério de serem objetos comuns em cidades, especificamente em ruas e avenidas. É importante ratificar que a distribuição e o formato da pontos nos dois *dataset* são bem semelhantes e embasa a relevância da construção do *dataset*.

O *dataset* ModalNet40 é um benchmark presente na literatura que contém nuvens de pontos de objetos sintéticos. Formado por 12311 malhas geradas por desenho assistido por computador (CAD), divididas em 9843 para treino e 2468 para testes, o ModelNet possui 40 classes de objetos 3D e é bastante utilizado para análise de nuvens de pontos. As redes

Figura 11 – Exemplo das amostras de cada classe dos conjuntos de dados:(1) árvore, (2) carro e (3) pessoa.
Fonte: Elaborada pelo autor.

escolhidas nesta pesquisa, citadas na seção 2.4 foram escolhidas por serem *benchmarks* do *dataset* Modelnet40 (WU *et al.*, 2015). Por esses motivos esse estudo optou por esse *dataset*. Além disso, suas nuvens de pontos possuem alta precisão por serem formados a partir de modelos de desenho assistidos por computador (CAD).

3.3.1 Aquisição dos Pontos

As nuvens de pontos possuem três classes: carro, árvore e pessoas. Os conjuntos de dados foram gerados em duas situações diferentes a partir do LIDAR: um ambiente simulado e um ambiente do mundo real. O ambiente simulado foi criado usando o software chamado Webots (versão R2020a-rev1). Software de código aberto e disponível em *https://cyberbotics.com/*. O software utiliza a linguagem Python para programar e executar os comandos do robô no simulador. O Lidar LMS 291 é modelo disponível e escolhido para aquisitar as nuvens de pontos dos objetos inseridos no ambiente do Webots.

Figura 12 – Ambiente Webots.
Fonte: Elaborada pelo autor.

Na sequência, a figura 12 ilustra o Robô Pioneer 3 equipado com o Lidar LMS 291 da SICK, o ambiente simulado e a respectiva nuvem de pontos do ambiente. Ao executar a

varredura robô gerou a nuvem de pontos dos objetos ao seu redor, especificamente uma árvore, que se assemelha a um coqueiro, e uma pessoa.

O LiDAR LMS511, especificamente, é um sensor 2D que gera um vetor de distâncias dos pontos das superfícies onde seu feixe luminoso reflete . Dessa forma, para realizar a varredura em uma sequência de profundidades diferentes, com o objetivo de montar um perfil tridimensional da cena na qual o sensor está inserido foi necessário desenvolver um algoritmo integrado ao software *Robot Operating System* (ROS). E assim foi possível gerar a nuvem de pontos a partir da fusão de uma série de vetores de distância. A aquisição dos dados reais foi realizada a partir de um Sistema de Aquisição Móvel (SAM). O SAM tem como componente principal o sensor LiDAR (modelo LMS511), baterias, suporte LiDAR (tripé), notebook e unidade de medição inercial(UMI). As especificações do sensor LiDAR estão descritas na seção 2.3. As aquisições foram realizadas em ambientes internos e ambientes externos, especificamente apenas a classe pessoa foi aquisitada nos dois ambientes. A distância entre o SAM e o objeto adquirido varia entre 5 e 30 metros. Essa variação da distância respeita o alcance indicado pelo fabricante do sensor LIDAR que garante uma faixa de operação de até 80 metros (AMARADI *et al.*, 2016). Na aquisição de nuvem de pontos de pessoas e carros, a variação da distância oscilou entre 5 e 10 metros. Na classe árvore, a distância de aquisição alternou de 10 a 30 metros, devido ao tamanho das árvores.

A distribuição de dados, o número de amostras presentes em cada classe e a divisão de acordo com o tipo de coleta de dados é apresentada na Tabela 2. O conjunto de dados construído com as aquisições do mundo real é nomeado de LLR514 (conjunto de dados do mundo real LAPISCO-LiDAR), enquanto o conjunto de dados que compreende as nuvens de pontos sintéticas é denominado LLS1500 (conjunto de dados sintéticos LAPISCO-LIDAR). As figuras 14 e 13 apresentam nuvens de pontos aquisitadas com o SAM em ambiente interno e externo.

3.4 ModalNet40

O *dataset* ModalNet40 é um *benchmark* presente na literatura que contém nuvens de pontos de objetos sintéticos. Formado por 12311 malhas geradas por desenho assistido por CAD, divididas em 9843 para treino e 2468 para testes, o ModelNet possui 40 classes de objetos 3D e é bastante utilizado para análise de nuvens de pontos. Por ter origem de CAD, seus objetos são sintéticos e fechados, por isso uma rede neural treinada com seus dados pode ter limitações

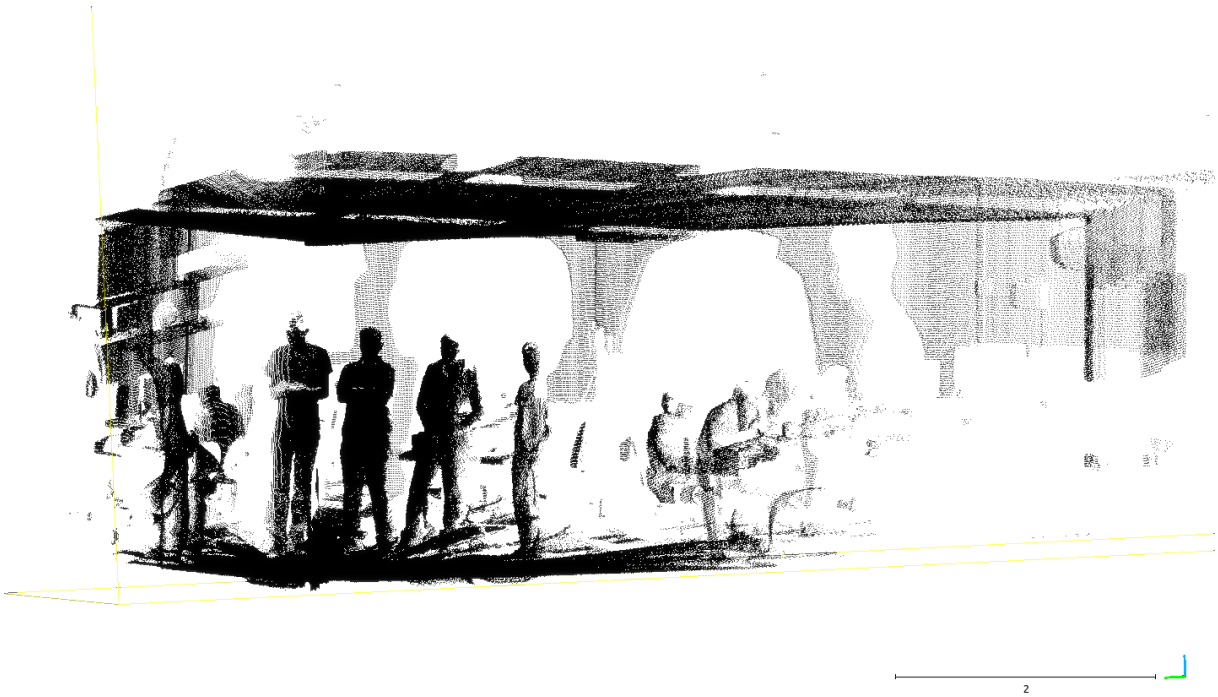


Figura 13 – Ambiente Interno.

Fonte: Elaborada pelo autor.

Figura 14 – Ambiente Externo.

Fonte: Elaborada pelo autor.

quando forem aplicadas para classificar objetos 3D aquisitadas em ambientes reais, pois não são objetos 3D fechados e a distribuição dos pontos é desordenada.

3.5 Técnicas de Pré-Processamento das Nuvens de Pontos

Métodos de pré-processamento em dados em sinais sinais tem como objetivo extrair atributos, padronizar dados ou parametrizar os dados com objetivo de melhorar a próxima etapa do processo de classificação, segmentação ou visualização (HAFEEZ; KATHIRISETTY, 2022). Nas seções a seguir são apresentadas as técnicas aplicadas nas nuvens que pontos dos objetos do dado proposto, essa etapa é uma das contribuições deste trabalho.

3.5.1 Normalização dos Pontos

A normalização da nuvem de pontos dos objetos é uma etapa de pré-processamento importante na abordagem proposta e uma das contribuições deste trabalho. A maioria das redes neurais utilizadas para classificação e reconhecimento de objetos, a partir de nuvens de pontos, utilizam a quantidade fixa de 2048 pontos como entrada, por exemplo a PointNet.(QI *et al.*, 2017)(AOKI *et al.*, 2019)

Considerando que a quantidade de pontos de um objeto varia de acordo com seu tamanho, por exemplo, a quantidade de pontos da nuvem de pontos de uma árvore é bem superior a nuvem de pontos de uma pessoa. A técnica de normalização da quantidade de pontos tem como objetivo padronizar a quantidade de pontos das nuvens sem interferir em suas características e estrutura e assim normalizar a nuvem de pontos de cada objeto presente no *dataset*. Nessa abordagem, o método de *upsampling* e *downsampling* são utilizados para remover e adicionar, respectivamente, amostras sintéticas (TORAL; CHAKRABARTI, 1993). Esse método é utilizado em processamento de sinais para aumentar a taxa de amostragem em X vezes, adicionando X-1 de zeros entre as amostras do sinal original da imagem. Após esse passo, o método aplica um filtro tipo passa-baixa para reconstrução do conjunto (LI *et al.*, 2019). No caso desta aplicação, o método *inter nearest* é aplicado nas nuvens de pontos baseado no conceito de vizinho mais próximo para interpolação trilinear (BOURKE, 1999) e assim redimensionar a nuvem de pontos do objeto através da combinação de oito pontos diferentes, o que implica na menor perda de informação no conjunto de pontos resultante.

Considerando que uma nuvem de pontos tenha, inicialmente, P pontos e ao fim dessa

etapa a nuvem de pontos esteja normalizada em N pontos, por exemplo 2048 pontos. O primeiro passo para que essa situação tenha sucesso e não modifique ou reduza a forma da nuvem de pontos é encontrar a proporção de pontos entre as dimensões de cada eixo cartesiano. Através da equação 3.7 temos as proporções.

$$s_X = s_Y = s_Z = \frac{P}{N} \tag{3.7}$$

Sabendo que que s_X é a razão para o eixo x, para o eixo y essa razão é s_Y e s_Z é a proporção para o eixo z. Dessa forma é possível calcular a nova posição de um ponto sobre cada eixos, através da Equação 3.8,

$$x_f = y_f = z_f = x's_X = y's_Y = z's_Z, \ \forall \ x = y = z = 1, ..., N \tag{3.8}$$

assim temos a variação em x dado por $\Delta_x = x_f - x$, a variação em y descrita por $\Delta_y = y_f - y$ e a variação em z sendo $\Delta_z = z_f - z$.

Dessa forma é calculada os pontos da nuvem pré-processado através da combinação de oito pontos que formam um cubo, expresso por (x,y,z), $(x+1,y,z)$, $(x,y+1,z)$, $(x,y,z+1)$, $(x+1,y,z+1)$, $(x,y+1,z+1)$, $(x+1,y+1,z)$ e $(x+1,y+1,z+1)$. E a equação 3.9 representa esse cálculo.

$$\begin{aligned} K(x',y',z') = {} & L(x,y,z)(1-\Delta_x)(1-\Delta_y)(1-\Delta_z) + L(x+1,y,z)\Delta_x(1-\Delta_y)(1-\Delta_z) + \\ & L(x,y+1,z)(1-\Delta_x)\Delta_y(1-\Delta_z) + L(x,y,z+1)(1-\Delta_x)(1-\Delta_y)\Delta_z + \\ & L(x+1,y,z+1)\Delta_x(1-\Delta_y)\Delta_z + L(x,y+1,z+1)(1-\Delta_x)\Delta_y\Delta_z + \\ & L(x+1,y+1,z)\Delta_x\Delta_y(1-\Delta_z) + L(x+1,y+1,z+1)\Delta_x\Delta_y\Delta_z \end{aligned} \tag{3.9}$$

sendo K o resultado do pré-processamento e L a nuvem de pontos original.

As figuras 15, 16 e 17 ilustram a técnica de normalização aplicadas nas três classes do *dataset* proposto. As etapas de normalização descritas no parágrafo anterior, foram aplicadas na nuvem de pontos de um cubo e de uma esfera, por serem objetos comuns que proporcionam uma boa visualização da técnica. Nas figuras 16, é nítido a variação da quantidade de pontos, consequentemente a densidade da nuvem, de acordo com o aumento da quantidade de pontos da nuvem de pontos do cubo. Da mesma forma é perceptível a técnica aplicada na nuvem de pontos da esfera, ilustrada nas figuras 17.

As imagens demonstram que os pontos adicionados ou eliminados não interferem, de forma incisiva, na forma do objeto porque os pontos manipulados são bem próximos entre si. E em algumas partes da nuvem de pontos é perceptível a impressão de sobreposição de dois ou

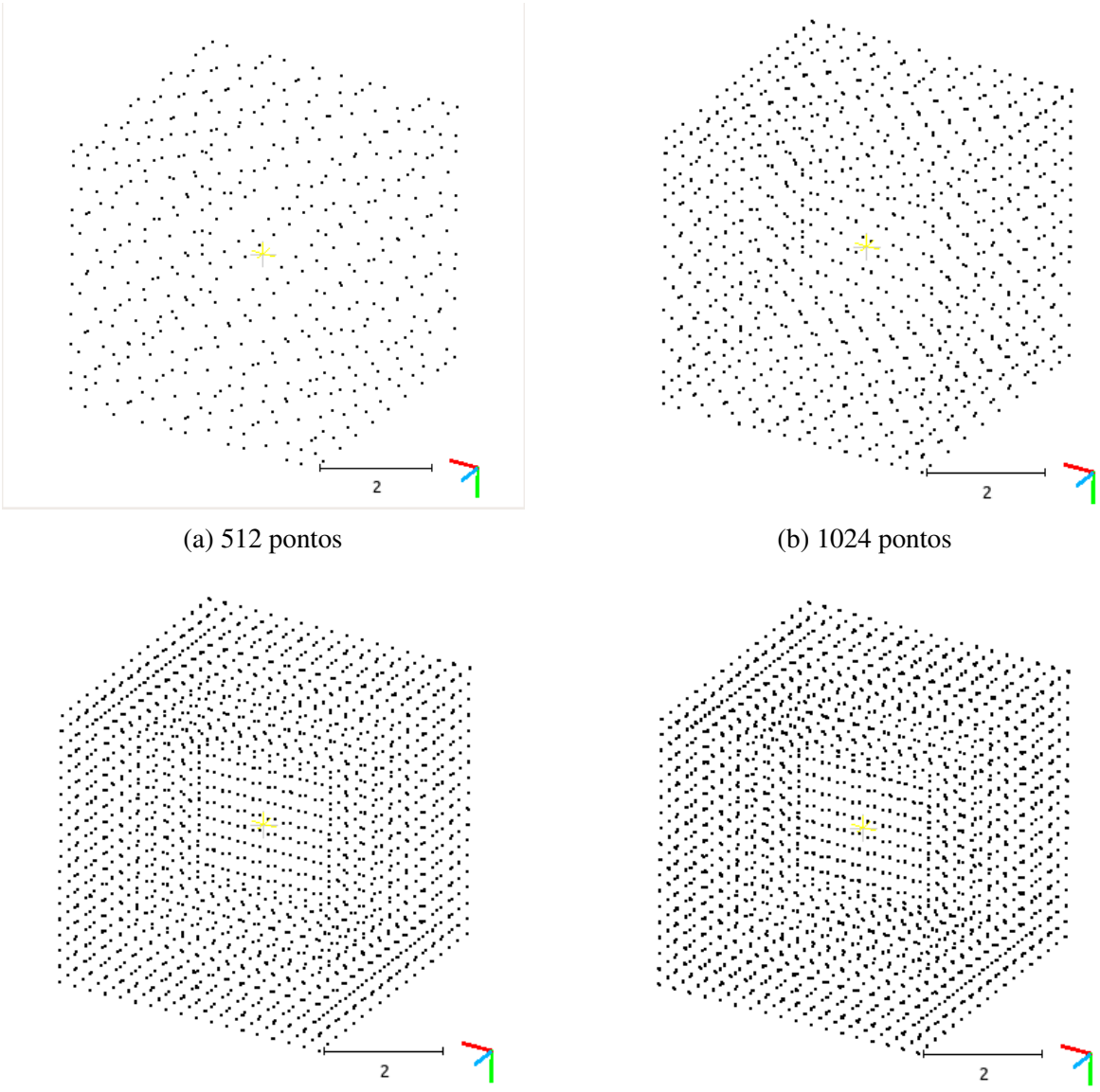
Figura 15 – Normalização da nuvem de pontos em cada classe do dataset.
Fonte: Elaborada pelo autor.

mais pontos porém eles possuem coordenadas diferentes e esse fato ocorre devido a configuração do tamanho ponto na geração da figura. O ponto pode ser reduzido no momento da geração da figura porém a qualidade de visualização da imagem reduz e impossibilita uma boa visualização.

A figura 15 apresenta a normalização da nuvem de pontos das classes árvores, carro e pessoa de 9961, 11189 e 2316 pontos, respectivamente, em 2048 pontos. Essas amostras estão presentes no *dataset* desenvolvido nesta pesquisa. A distância entre o sensor LiDAR e o objeto interfere na quantidade de pontos e concentração de pontos em determinadas regiões das nuvens de pontos dos objetos. No entanto, a etapa de pré processamento, especificamente de normalização, reduz essa limitação na aquisição das nuvens de pontos.

3.5.2 Ajuste no Plano Cartesiano

Após a etapa de normalização, o pré-processamento abordado é a de ajuste no plano cartesiano da nuvem de pontos do objeto 3D. Nesta etapa é aplicada uma transformação do ponto no espaço euclidiano 3D. Esta operação de translação move a nuvem de pontos do objeto para a origem do plano cartesiano 3D. Nessa etapa de pré-processamento é importante salientar que



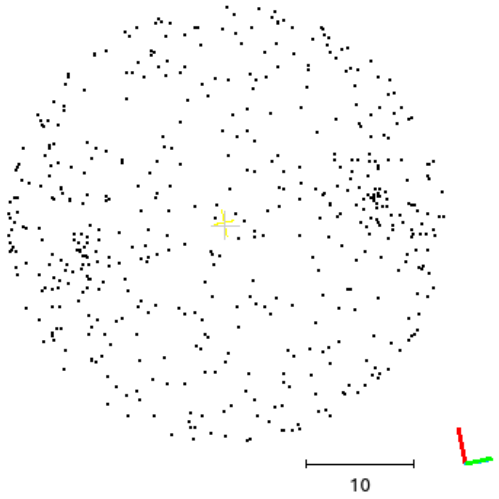
(a) 512 pontos (b) 1024 pontos

(c) 2048 pontos (d) 4096 pontos

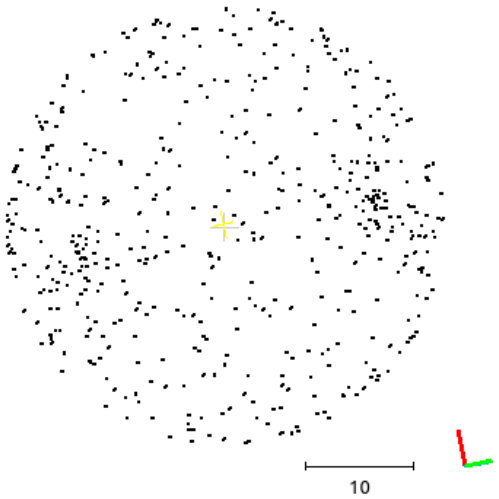
Figura 16 – Nuvens de pontos de um cubo com diferentes níveis de normalizações.
Fonte: Elaborada pelo autor.

a nuvem de pontos não sofre deformação, adição ou remoção de pontos e também mantém a origem padrão para todas as nuvens de pontos dos objetos. A operação de transformação pode ser expressa matematicamente pela Equação 3.10. Abordagens que utilizam a estratégia de conhecimento da origem da nuvem de pontos também são encontrados nos trabalhos (ZHOU; TUZEL, 2018; LANG *et al.*, 2019)

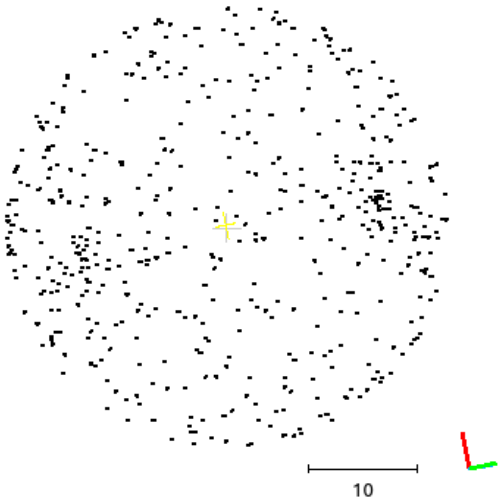
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t^x \\ 0 & 1 & 0 & t^y \\ 0 & 0 & 1 & t^z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.10}$$



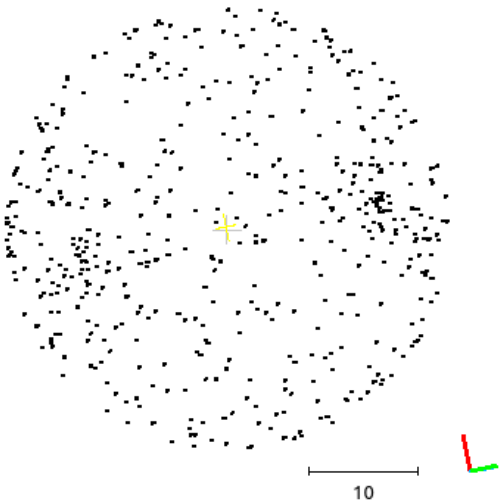
(a) 512 pontos



(b) 1024 pontos



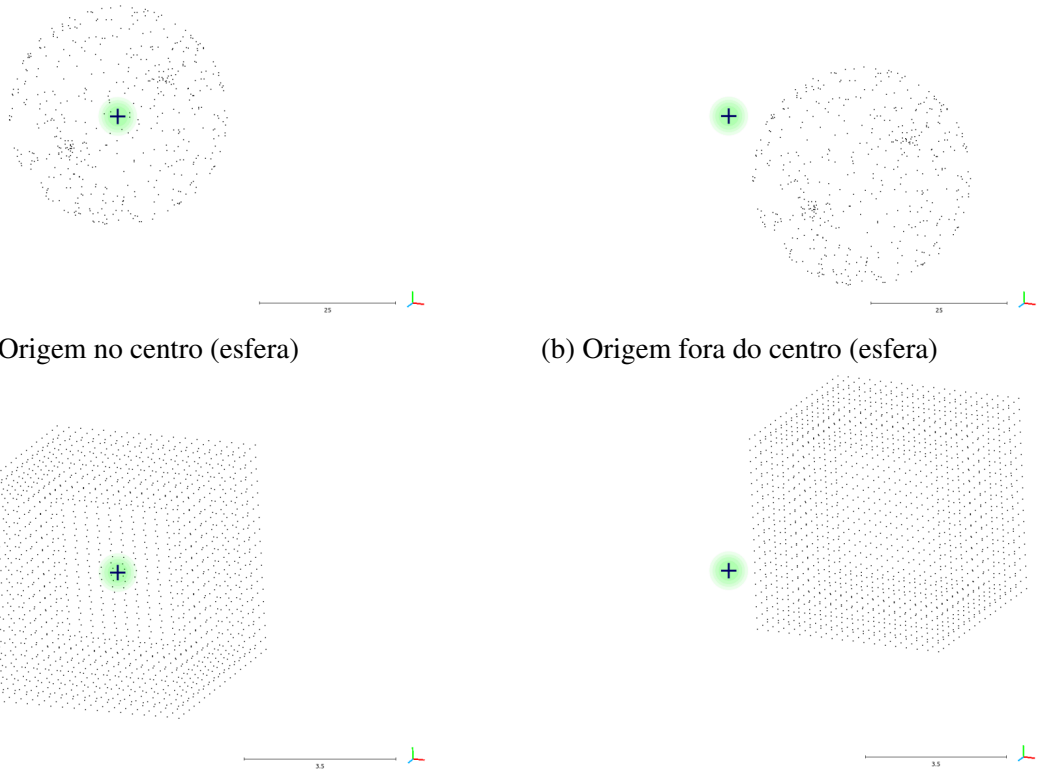
(c) 2048 pontos



(d) 4096 pontos

Figura 17 – Nuvens de pontos de uma esfera com diferentes níveis de normalizações.
Fonte: Elaborada pelo autor.

A figura 18 ilustra a etapa de pré-processamento para ajuste da origem aplicado em objetos com formatos espaciais diferentes. A nuvem de pontos de uma esfera e de um cubo foram utilizadas para exemplificar esta etapa. A origem da nuvem de pontos está destacada em todas as figuras com um símbolo de mais "+"e sombreado. A origem nas nuvens de pontos são geradas no momento da aquisição, dessa forma as posições são variadas. Essa etapa de ajuste no plano cartesiano é uma das contribuições da pesquisa e ocasionou uma interferência direta na melhor performance das redes neurais. Os resultados dessa etapa serão discutidos no capítulo 4.



(a) Origem no centro (esfera)

(b) Origem fora do centro (esfera)

(c) Origem no centro (cubo)

(d) Origem fora do centro (cubo)

Figura 18 – Etapa de ajuste da nuvens de pontos no plano cartesiano 3D para origem.
Fonte: Elaborada pelo autor.

3.6 Validação Cruzada

Após a aquisição e rotulação das nuvens de pontos, abordadas nas seções 3.3.1, conjunto de dados é embaralhado e dividido em três partes de tamanhos iguais. Essa partição é realizada com o objetivo de realizar a Validação Cruzada (REFAEILZADEH *et al.*, 2016) como forma de gerar resultados para cada modelo e assim mensurar o seu desempenho utilizando as métricas de avaliação.

O método de validação cruzada engloba treinamentos e testes de um modelo de aprendizagem de máquina em diferentes agrupamentos de um mesmo *dataset*. A figura 3.6 apresenta como o mesmo modelo é treinado e testado n vezes, onde n é o número de partições em que foi dividido o conjunto de dados, assim tem-se o subconjunto de treino formado por $n-1$ partições, em branco na Figura 3.6, e o subconjunto de teste formado pela partição restante, em azul. Em cada uma das n iterações o subconjunto de treino é formado por um arranjo diferente de partições, da mesma forma que o subconjunto de teste passa a ser a partição que sobra. Isto possibilita avaliar os modelos de Classificação de objetos 3D em diferentes arranjos de um

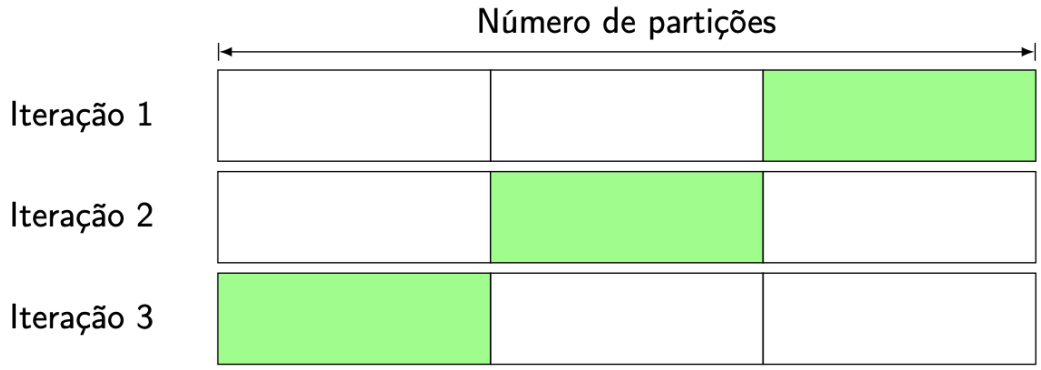


Figura 19 – Repartições da validação cruzada.
Fonte: Elaborada pelo autor.

mesmo conjunto de dados, tendo ao final desse processo *n* conjuntos de métricas de avaliação para analisar, ao invés de um, como na classificação tradicional.

Em cada uma das três iterações na Validação Cruzada, todas as redes foram treinadas em 50 épocas, mantendo as configurações originais de quando foram propostas. Apenas a quantidade de neurônios nas camadas de saída de 40 classes, quantidade de classes da ModelNet40, foram alteradas para 3 classes, número de classes do *dataset* proposto.

3.7 Métodos de Avaliação de Estatística

Para avaliar os resultados gerados na tarefa de classificação de objetos 3D a partir da nuvem de pontos, métodos estatísticos também foram aplicados nos resultados obtidos nos experimentos. Essa validação fortalece os resultados das métricas de avaliação e embasa a relevância dos métodos da etapa de pré-processamento propostos para classificação das nuvens de pontos dos objetos 3D.

Primeiro realizamos o teste estatístico de análise de variância (ANOVA) para identificar se os métodos de classificação de objetos 3D apresentam resultados significativamente diferentes entre si. Neste teste, o valor $p < 0{,}05$ representa a rejeição da hipótese nula H_0, tornando assim a hipótese alternativa H_1 verdadeira. O teste de confiança utilizado foi de 0,95 e as hipóteses são:

H_0: Todos os métodos de classificação são equivalentes estatísticos. Isso significa que a escolha da rede neural não importa para fins de classificação de objetos 3D.

H_1: Considera que ao menos uma rede neural difere de outra. Isso implica que os resultados entre as redes diferem, onde um resultado pode ser superior ou inferior que outro.

No *f-value*, cada relação F é calculada dividindo o valor da soma média de quadrados

entre os grupos (MSB) por soma média de erro de quadrados (MSE). Os valores DF são os graus de liberdade na fonte de variação dos dados. SS é a soma dos quadrados devido à fonte e a variação dos dados. MS é a soma média de quadrados devido à fonte e de variação dos dados. F significa "a estatística F"e P significa "o valor P". O valor de Quadrados Médios (Mean squares, MS) é calculado dividindo SS pelo valor de DF. Sendo a soma dos quadrados (sums of squares, SS) entre os meios do grupo e a grande média. O SS quantifica a variabilidade entres os grupos de interesse. E o grau de liberdade (degrees of freedom, DF) são: caso tenha *n* pontos de dados totais coletados, então há $n-1$ graus totais de liberdade; caso haja *m* grupos sendo comparados, então há *m - 1* graus de liberdade associados ao fator de interesse. E se houver *n* pontos de dados totais coletados e *m* grupos sendo comparados, então há *n-m* de graus de erro de liberdade.

Uma limitação do teste estatístico ANOVA é que ele não identifica qual ou quais redes diferem das outras, portanto não é possível destacar a melhor ou pior rede neste teste.

Dessa forma, é necessário realizar o teste de diferença honestamente significativa de Tukey (Turkey's Honestly Significant Difference, Turkey's HDS). Semelhante ao teste estatístico ANOVA, o valor $p < 0{,}05$ representa a rejeição da hipótese nula H_0, tornando assim a hipótese alternativa H_1 verdadeira. Porém, nesse confronta as redes individualmente com os mesmos parâmetros de aceitação ou rejeição das hipóteses abaixo:

H_0: A rede em destaque é equivalente à outra rede em comparação. Não existe diferença de significância estática no uso de uma rede ou outra para classificação de objetos 3D.

H_1: Pressupõe que as redes são diferentes entre si. Há uma diferença de significância estatística no uso de cada uma ou outra rede para a classificação de objetos 3D.

Uma amostra de cada métrica de avaliação obtidos da validação cruzada foi usada para realizar os testes estatísticos. Deste modo, as hipóteses destacadas anteriormente serão aceitas ou recusadas se, e somente se, os valores de *p* de cada métrica obedecerem, simultaneamente, a condição estabelecida anteriormente.

3.8 Métricas de Avaliação

Com o objetivo de verificar o desempenho das redes inseridas na pesquisa, cinco métricas de avaliação são utilizadas: Acurácia, Precisão, Sensibilidade, F1-Score, Especificidade e tempo de predição por amostra. O tempo de predição é calculado por amostra para identificar, de forma individual, o custo de tempo que cada rede neural utiliza para predizer a nuvem de pontos de cada classe. O calculo considera o tempo de predição de cada classe e é calculado a

média aritmética somando todos os tempos e dividindo pela quantidade de predições realizadas. As métricas para medir o desempenho são baseadas na matriz de confusão. O projeto é focado na classificação de objetos, a partir da sua nuvem de pontos. Portanto, os verdadeiros positivos (VP) são a predição correta da classe rotulada com a mesma classe, por exemplo, a amostra predita corresponde a classe pessoa e a amostra é rotulada com a classe pessoa. Os verdadeiros negativos (VN) é a predição correta da classe que a amostra não é rotulada, por exemplo, a predição indicou corretamente outra classe enquanto a amostra real é a árvore. Já o falso positivo corresponde a predição incorreta da classe que a amostra real é rotulada, por exemplo, predizer que a amostra é uma árvore enquanto que a amostra é uma pessoa. E, por último, o falso negativo é a predição incorreta da classe que a amostra não é rotulada, por exemplo, predizer incorretamente outra classe enquanto a amostra real é um carro.

Acurácia é a relação entre as predições corretas e todas as predições realizadas pela rede, determinada pela equação 3.8, onde VP é verdadeiro-positivo, VN é verdadeiro-negativo, FN é falso-negativo e FP é falso-positivo Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN). No contexto da pesquisa, a acurácia indica o percentual que o modelo classificou de forma correta, dentre todas as classificações. É a métrica mais indicada para a situação deste trabalho. É também a métrica comum entre todas as redes neurais comparadas no estado da arte.

$$Acurácia = \frac{VP+VN}{VP+FP+FN+VN}. \tag{3.11}$$

A métrica sensibilidade é obtida através da relação entre as predições positivas corretas e predições negativas erradas, a equação 3.12 ilustra o cálculo. Sua métrica avalia a capacidade do modelo indicar quando o objeto pertence à classe. Por exemplo, se a classe correta é carro, o valor da porcentagem da sensibilidade indica o grau de acerto que o modelo indicou que o objeto pertence a classe carro.

$$Sensibilidade = \frac{VP}{VP+FN}. \tag{3.12}$$

Na precisão, determinada pela equação 3.13, o valor é calculado baseado na relação entre as predições positivas corretas e todas as predições positivas encontradas. Dentre todas as classificações de classe positivo que o modelo fez, quantas estão corretas. Por exemplo, se a classe correta é pessoa, o valor da porcentagem da precisão indica, dentre todas as classificações de classe pessoa que o modelo fez, quantas estão corretas.

$$Precisão = \frac{VP}{VP + FP}. \tag{3.13}$$

De acordo com a equação 3.14 , é possível computar o valor F1-*Score* a partir do cálculo do balanceamento entre os valores de precisão e sensibilidade, é uma média harmônica entre as duas. Quando F1-*Score* é baixo significa que a precisão ou a sensibilidade está baixa.

$$F1 - Score = 2 \cdot \frac{Precisão \cdot Sensibilidade}{Precisão + Sensibilidade}. \tag{3.14}$$

Por fim, a métrica especificidade é calculada através da razão entre as todas predições corretas e predições positivas corretas, sua equação é expressa por:

$$Especificidade = \frac{VN}{VN + FP}. \tag{3.15}$$

Na métrica especificidade o valor da porcentagem indica quanto o modelo é capaz de dizer quando uma amostra não pertence a classe positiva. Por exemplo, se a classe positiva é a árvore, a especificidade indica o quanto o modelo é capaz de indicar que as amostras não pertencem à classe árvore.

Com os recursos das métricas de avaliação e avaliação estatística dos resultados é possível mensurar o desempenho das redes com fundamentação na tarefa de classificação de objetos 3D com base em nuvem de pontos. Sabendo que os resultados do *benchmarks* da *Modalnet40* avaliam apenas a métrica de acurácia, este trabalho contribui para uma análise mais ampla na incumbência de classificação de objetos 3D.

4 RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados, discussões e comparações entre as redes neurais do estado da arte e a rede proposta Lidar3DNet, considerando a metodologia e métricas de avaliações descritas na seção 3. Todos os ambientes foram configurados no hardware com sistema operacional *Ubuntu*, processador *Intel Core* i7, 8 GB de memória RAM, placa de vídeo *GeForce* GTX 1070. Além disso, bibliotecas como *python* 3.x, *scikit-learn*, *pytorch*, *tensorflow* e *keras* foram utilizadas com diferentes versões. Por esse motivo, o recurso da tecnologia *Docker* foi aplicada e cada rede neural possui um *container* específico. A primeira seção 4.1 discute a validação da configuração dos ambientes da redes do estado da arte. Na seção 4.2 é aplicado os dados reais, originais e com pré-processamento, na entrada das redes neurais. Na seção 4.3 é aplicado os dados sintéticos, originais e com pré-processamento, na entrada das redes neurais. Em seguida, a seção 4.4 analisa a variação da quantidade de pontos das nuvens na classificação dos objetos 3D. E por fim, a seção 4.5 compila uma análise geral dos resultados desta pesquisa.

4.1 Validação das Configurações das Redes com *dataset* ModelNet40

Na primeira etapa é configurada todas as redes de acordo com as instruções presentes nos repositórios indicados por cada trabalho para iniciar os testes. Para validar a configuração realizada, as arquiteturas foram executadas com o *dataset* ModelNet40 e as instruções de treinamento e testes informados pelos autores. A comparação entre os resultados de acurácia informados pelos autores e os resultados de acurácia obtidos na arquitetura configurada estão presentes na tabela 3, inclusive a rede Lidar3DNet, proposta neste trabalho.

A igualdade e, em alguns casos aproximação, entre as acurácias das redes é esperada pois a configuração e *dataset* são semelhantes aos que os autores divulgaram em seus estudos. O tempo de processamento não foi informado devido a diferença de recurso computacional entre as máquinas que foram realizados os testes. No entanto, é algo que não interfere no objetivo desta primeira etapa que é validar a configuração das redes. O resultado específico da rede Lidar3DNet, não obteve resultado relevante quando comparados com as demais. Porém, como especificado na seção 3, a rede Lidar3DNet é projetada baseada no *dataset* proposto que possui três classes e o *dataset* ModelNet40 possui 40 classes.

Outros trabalhos, por exemplo, DeepPano (SHI *et al.*, 2015) e 3DShapeNets (WU *et al.*, 2015) também usaram o *dataset* ModelNet40 e tiveram acurácia abaixo de 80%, por esse

Tabela 3 – Validação da arquitetura com as métricas de Acurácia (Acc), em porcentagem (%), de cada rede sobre o conjunto de dados ModelNet40.

Rede	***Acurácia(%)***	
	Artigo Original	**Replicação**
3DMedPT	93,40	93,20
Lidar3DNet	80,63	80,70
CurveNet	94,20	94,20
GBNet	91,04	91,04
LDGCNN	92,90	92,90
PAConv	93,90	93,90
PointNet	89,20	89,20
PVT	94,00	94,00
RSCNN	93,60	93,60
SimpleView	93,90	93,90

Fonte: Elaborada pelo autor.

motivo não foram adicionadas na tabela 3 e nem consideradas na pesquisa deste trabalho. A rede SimpleView obteve um dos melhores resultados, porém em sua configuração as nuvens de pontos dos objetos possuem 1024 pontos e esse fato deve ser observado na sequência das discussões.

4.2 Avaliação com *dataset* real como entrada nas redes do estado da arte, com dados originais e pré-processados.

Com as configurações validadas, a etapa seguinte é executar as redes configuradas alterando os dados de entrada das redes, substituindo os dados da ModelNet40 para o *datasets* com dados reais propostos neste trabalho. As tabelas 4 e 7 apresentam o desempenho das redes neurais do estado da arte e a rede proposta Lidar3DNet, com dados originais e com dados aplicados à etapa de pré-processamento, respectivamente. As tabelas são compostas por sete colunas que destacam a rede neural em análise, os subconjuntos criados na etapa de validação cruzada e as cinco colunas seguintes possuem as métricas de avaliação. A melhor acurácia de cada rede neural está destacada em azul e a melhor acurácia entre todas está destacada de verde.

A rede RSCNN obteve resultados expressivos na acurácia e caracteriza como a melhor rede neural no quesito acurácia. Ao aplicar os dados reais na entrada da RSCNN a acurácia atingiu 100% nas duas configurações. Pelo fato da RSCNN utilizar a arquitetura CNN

Tabela 4 – Métricas, em porcentagem (%), de cada Rede Neural e em cada subconjuntos de teste sobre os **originais** e **reais**

Rede	**Subconjunto**	**Acurácia**	**F1-*Score***	**Precisão**	**Sensib.**	**Especif.**
3DMedPT	1	58,72	47,54	55,31	54,86	54,86
	2	59,41	56,18	64,92	61,02	61,02
	3	**75,29**	73,04	73,49	75,32	75,32
Lidar3DNet	1	80,23	77,13	77,75	76,65	89,54
	2	**84,80**	83,06	82,22	84,87	93,00
	3	75,44	71,24	71,47	71,69	87,84
CurveNet	1	89,53	87,39	87,36	87,51	87,51
	2	90,06	88,61	87,77	89,83	89,83
	3	**90,64**	89,49	89,45	89,57	89,57
GBNet	1	40,70	29,42	26,21	33,88	33,88
	2	37,43	34,07	34,51	35,78	35,78
	3	**50,29**	22,31	16,76	33,33	33,33
LDGCNN	1	91,25	89,27	88,41	90,67	90,67
	2	92,46	91,32	90,41	92,66	92,66
	3	**92,50**	91,58	91,39	91,91	91,91
PAConv	1	**84,30**	80,61	81,16	80,78	80,78
	2	81,87	77,22	79,06	78,04	78,04
	3	78,94	72,89	74,45	74,31	74,31
PointNet	1	**87,79**	86,12	85,67	86,90	86,90
	2	82,74	81,73	80,88	84,10	84,10
	3	84,52	82,06	81,47	83,20	83,20
PVT	1	**93,60**	92,63	92,16	93,16	93,16
	2	90,64	90,09	88,94	91,84	91,84
	3	91,22	90,16	89,19	91,75	91,75
RSCNN	1	97,09	96,76	96,35	97,27	97,27
	2	98,83	98,82	98,92	98,74	98,74
	3	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**
SimpleView	1	86,63	85,66	84,68	88,08	88,08
	2	87,72	87,81	87,21	91,05	91,05
	3	**90,06**	89,30	88,75	91,15	91,15

Fonte: Elaborado pelo autor.

Tabela 5 – Resultados do teste *one-way* ANOVA para experimentos sobre os dados os **originais** e **reais**.

Métrica		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Acurácia	C(Redes)	9,00	0,72	0,08	45,85	0,00
	Residual	20,00	0,03	0,00	-	-
F1-*Score*	C(Redes)	9,00	1,13	0,12	46,75	0,00
	Residual	20,00	0,05	0,00	-	-
Precisão	C(Redes)	9,00	1,14	0,12	56,84	0,00
	Residual	20,00	0,04	0,00	-	-
Sensibilidade	C(Redes)	9,00	0,96	0,10	59,74	0,00
	Residual	20,00	0,03	0,00	-	-
Especificidade	C(Redes)	9,00	0,23	0,02	82,91	0,00
	Residual	20,00	0,00	0,00	-	-

Fonte: Elaborado pelo autor.

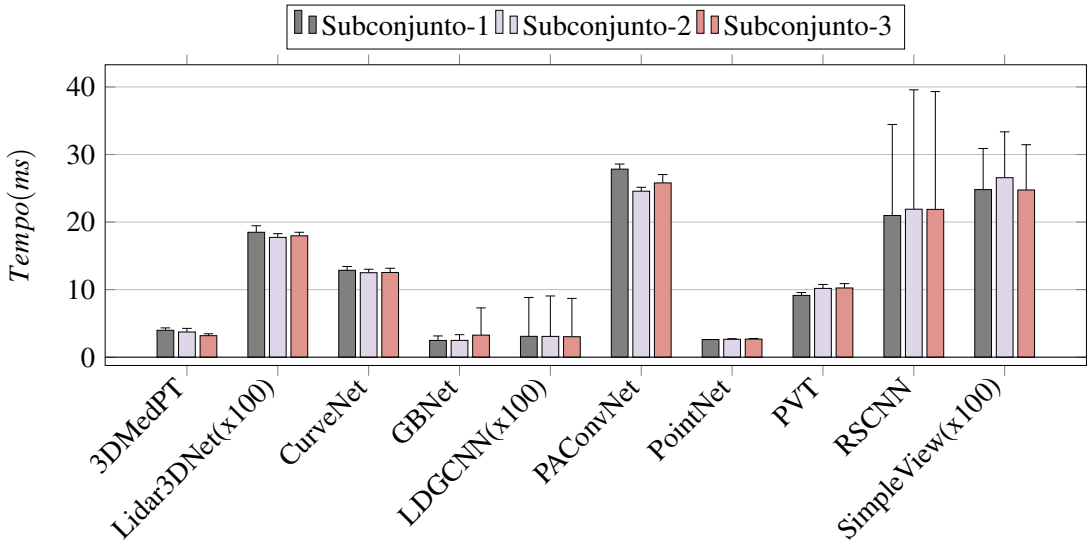


Figura 20 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento sobre os dados os **originais** e **reais**

.

Fonte: Elaborada pelo autor.

como base e estender para configurações irregulares de análise de nuvens de pontos, a etapa de pré-processamento proposta não interferiu de forma relevante nas métricas de avaliação. A RSCNN se adaptou aos dados originais e pré-processados. Além disso, o método aprende através da restrição de topologia geométrica entre os pontos, fato esse que o uso de dados reais contribuem para um bom desempenho. Considerando o tempo de processamento para classificação de cada amostra, ilustrado nas figuras 21 e 20, a RSCNN demanda um elevado tempo se comparado com as demais, figurando entre as três mais lentas. Aplicado a uma situação

Tabela 6 – Resultados teste Tukey HSD sobre os dadosos **originais** e **reais**.

Subconjunto	Subconjunto2	Acurácia	F1-*Score*	Precisão	Sensenb.	Especif.
3DMedPT *(Acc = 75,29%)*	Lidar3DNet	0,005	0,010	0,017	0,016	0,008
	CurveNet	0,001	0,001	0,001	0,001	0,001
	GBNet	0,001	0,001	0,001	0,001	0,001
	LDGCNN	0,001	0,001	0,001	0,001	0,001
	PAConv	0,007	0,005	0,002	0,03	0,002
	PointNet	0,001	0,001	0,004	0,003	0,001
	PVT	0,001	0,001	0,001	0,001	0,001
	RSCNN	0,001	0,001	0,001	0,001	0,001
	SimpleView	0,001	0,001	0,001	0,001	0,001
Lidar3DNet *(Acc = 84,80%)*	CurveNet	**0.090**	**0.109**	**0.112**	**0.158**	**0.170**
	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.391**	**0.342**	**0.412**	**0.474**	**0.505**
	PAConv	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PointNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PVT	**0.091**	**0.072**	**0.062**	**0.058**	**0.070**
	RSCNN	0.0041	0.036	0.046	0.046	0.045
	SimpleView	**0.792**	**0.734**	**0.785**	**0.786**	**0.853**
CurveNet *(Acc = 90,64%)*	GBNet	0,001	0,001	0,001	0,001	0,001
	LDGCNN	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	PAConv	**0,356**	**0,226**	**0,287**	**0,086**	**0,199**
	PointNet	**0,880**	**0,900**	**0,900**	**0,900**	**0,872**
	PVT	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
GBNet *(Acc = 50,29%)*	LDGCNN	0,001	0,001	0,001	0,001	0,001
	PAConv	0,001	0,001	0,001	0,001	0,001
	PointNet	0,001	0,001	0,001	0,001	0,001
	PVT	0,001	0,001	0,001	0,001	0,001
	RSCNN	0,001	0,001	0,001	0,001	0,001
	SimpleView	0,001	0,001	0,001	0,001	0,001
LDGCNN *(Acc = 92,50%)*	PAConv	**0,135**	**0,087**	**0,125**	0,016	0,046
	PointNet	**0,564**	**0,731**	**0,642**	**0,579**	**0,456**
	PVT	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	RSCNN	**0,644**	**0,683**	**0,508**	**0,593**	**0,528**
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
PAConv *(Acc = 84,30%)*	PointNet	**0,600**	**0,704**	**0,688**	**0,590**	**0,688**
	PVT	**0,156**	**0,079**	**0,124**	0,012	0,047
	RSCNN	**0,054**	**0,054**	**0,056**	**0,056**	**0,056**
	SimpleView	**0,665**	**0,316**	**0,465**	0,045	0,307
PointNet *(Acc = 87,79%)*	PVT	**0,605**	**0,702**	**0,638**	**0,499**	0,459
	RSCNN	0,020	0,045	0,016	0,017	0,008
	SimpleView	**0,900**	**0,900**	**0,900**	0,838	**0,900**
PVT *(Acc = 93,60%)*	RSCNN	**0,604**	**0,713**	**0,511**	**0,672**	**0,526**
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
RSCNN *(Acc = 100,00%)*	SimpleView	**0,126**	**0,289**	**0,144**	**0,332**	**0,105**

Tabela 7 – Métricas para cada Rede em cada um dos conjuntos de teste de nuvens de pontos sobre os dados os **pré-processados ($\mathbf{S}_c$)** e **reais** (S_c).

Rede	**SubSubconjunto**	**Acurácia**	**F1-*Score***	**Precisão**	**Sensib.**	**Especif.**
3DMedPT	1_c	69,18	51,77	49,33	60,85	60,85
	2_c	70,58	57,71	75,39	63,63	63,73
	3_c	**73,52**	66,97	68,18	67,70	67,70
Lidar3DNet	1_c	97,13	95,74	94,95	95,59	98,71
	2_c	**97,70**	96,83	95,61	97,87	96,98
	3_c	96,85	92,19	93,42	92,53	92,84
CurveNet	1_c	93,60	92,90	92,47	93,47	93,47
	2_c	93,56	92,53	92,29	92,81	92,81
	3_c	**95,90**	95,36	95,82	95,32	95,32
GBNet	1_c	34,88	30,17	31,32	31,35	31,35
	2_c	36,25	31,65	31,63	32,26	32,26
	3_c	**38,59**	36,51	36,69	36,75	36,75
LDGCNN	1_c	95,62	95,18	94,35	96,14	96,14
	2_c	95,00	94,67	93,71	95,87	95,87
	3_c	**96,82**	96,51	96,35	96,82	96,82
PAConv	1_c	87,79	86,23	87,61	0.86,96	86,96
	2_c	**88,88**	87,60	86,87	89,20	89,20
	3_c	87,13	84,47	86,83	85,10	85,10
PointNet	1_c	81,39	77,14	79,20	78,47	78,47
	2_c	86,66	76,05	77,05	81,64	81,64
	3_c	**92,85**	92,14	91,97	92,71	92,71
PVT	1_c	95,93	95,70	94,76	96,98	96,98
	2_c	94,73	94,16	93,53	94,88	94,88
	3_c	**97,66**	97,50	97,12	97,96	97,96
RSCNN	1_c	98,25	97,99	97,56	98,53	98,53
	2_c	98,24	98,26	98,52	98,02	98,02
	3_c	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**
SimpleView	1_c	94,76	94,03	93,42	94,73	94,73
	2_c	95,90	95,64	95,87	95,49	95,49
	3_c	**98,83**	98,40	98,63	98,24	98,24

Fonte: Elaborada pelo autor.

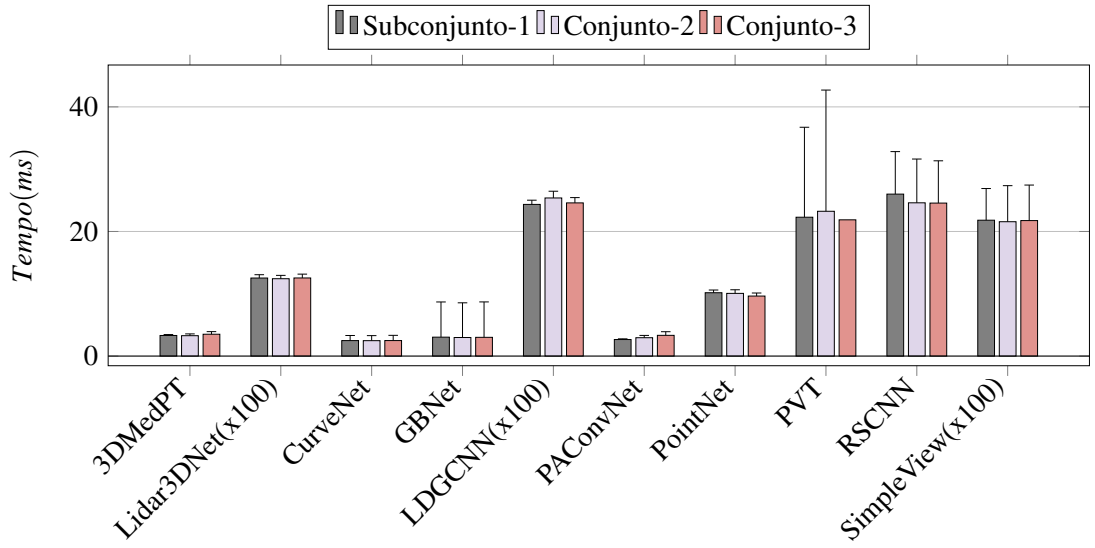


Figura 21 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento sobre dados os **pré-processados** e **reais**

Fonte: Elaborada pelo autor.

de classificação real, o tempo de classificação por amostra da RSCNN deve ser levado em consideração na sua escolha. Considerando as métricas de precisão, sensibilidade, especificidade e F1-*Score* a rede RSCNN demonstrou ser capaz de indicar o objeto que pertence a classe, o que não pertence a classe e classificar 100% das amostras positivas corretas.

No *dataset benchmark* ModelNet40 a rede PVT está entre as duas redes com acurácia acima de 94% . Quando os dados propostos foram inseridos na entrada da rede PVT a acurácia reduziu aproximadamente 1% no melhor conjunto e as métricas adicionadas figuraram acima de 92% demonstrando o método ser capaz de indicar o objeto que pertence a classe, indicar qual objeto não pertence a classe e classificar 100% das amostras positivas corretas. Sua configuração baseada nas tratativas de identificar *voxels* vazios e refinar estruturas com transformações rígidas são fatores que interferem no desempenho da rede. Dessa forma, quando a rede PVT recebe as nuvens de pontos com o método de pré-processamento a acurácia aumenta 97,66%, assim como as demais métricas, alcançando resultados melhores do que a ModelNet40. Essa melhora na perfomace indica que o método de pré-processamento proposta proporciona melhorias não só na rede LidarDNetV1 mas também na arquitetura da rede PVT, que possui métodos de classificação que atuam diretamente na organização pontos, através da criação e organização de *voxels*.

Com arquitetura projetada para classificar nuvens de pontos de imagens médicas, especificamente de nuvem de pontos de objetos fechados, como vasos sanguíneos. A rede neural 3DMedPT reduziu valores na métrica de acurácia, aproximadamente 18%, quando os dados

de entrada da ModelNet40 foram substituídos pelo *dataset* com nuvens de pontos reais. Seu bom desempenho na ModelNet40 atingiu acurácia acima de 93% porque as nuvens de pontos são geradas em CAD e também são nuvens fechadas, como descrito na seção 3.4. E ao aplicar as nuvens de pontos pré-processados, visando melhorar as métricas de avaliação, os valores não alteraram significativamente devido ao método ter como base a transformação das nuvens de pontos. Dessa forma, é possível afirmar que o método de pré-processamento proposto, não compromete de forma incisiva na classificação do objeto, pois ao normalizar e movimentar a nuvem de pontos no plano cartesiano o arranjo dos pontos não modificaram.

Sabendo que a quantidade de pontos dos objetos 3D na entrada da rede neural PointNet são 2048 pontos e comparando os resultado de acurácia da tabela 3 com a tabela 4 os valores foram próximos e esperados. Porém, com a inserção do *dataset* com nuvens de pontos reais a acurácia reduziu 2%, possivelmente causado pela redução da quantidade de classes. Como a acurácia, as demais métricas também tiveram valores próximos a 87%, mostrando assim que a PointNet é estável e indicada para classificação de nuvens de pontos. Ao aplicar as nuvens de pontos pré-processadas, a acurácia atingiu 92,85%, assim como as demais métricas, em seu melhor subconjunto. Considerando que a rede neural PointNet utiliza os dados diretos na sua entrada, como a rede Lidar3DNet, a normalização e renderização das nuvens de pontos melhoram consideravelmente seu desempenho.

Os resultados da rede CuverNet, com as nuvens de pontos do dataset proposto, a acurácia reduziu 4% aproximadamente em relação ao resultado do ModelNet40. As demais demais métricas tiveram valores estáveis e próximos de 90%, caracterizando a CurveNet recomentada para classificação de objetos 3D. Sabendo que seu método de orientar caminho ao percorrer os pontos foi treinada com nuvens de pontos desenhados em softwares (CAD), quando aplicado nuvens de pontos reais, que não possuem uma distribuição de pontos ordenada igual à geradas no CAD, a redução na eficiência de classificação é esperada. Após as nuvens de pontos serem pré-processadas, o resultado da rede CurveNet recuperou quase 5%, no agrupamento 3. Com as nuvens de pontos ajustadas para a mesma origem, a orientação do caminho realizada pelo método pode ter a mesma origem como referência, contribuindo para melhor performance. Com isso as métricas de avaliação ficaram acima de 95%, demonstrando o bom desempenho do método de pré-processamento proposto na pesquisa e a estabilidade do método, pois as cinco métricas demonstraram valores balanceados em seus resultados.

A redução das métrica de acurácia de 93,90% com o dataset da ModelNet40 para

90 % no melhor subconjunto do *dataset* real proposto e 86 % no pior subconjunto é motivado pelo aumento do número de pontos dos objetos para 2048 pontos. O aumento da quantidade de pontos tem o objetivo de igualar as condições para todas as redes, porém a rede SimpleView tem seu modelo treinado com 1024 pontos e isso deve ter influenciado na perfomace e assim reduziu suas métricas. No entanto, após a etapa de pré-processamento proposta, a rede SimpleView obteve um dos melhores aumentos na porcentagem de suas métricas e figurou entre as melhores redes, como ilustra a tabela **??**. Com as métricas próximas de 98%, seu desempenho nos dados reais pré-processados foi melhor do que a acurácia de 93,90% utilizando as nuvens de pontos da ModelNet40. A rede PAConv teve destaque negativo pois sua acurácia reduziu 15% em seu menor resultado, se comparado com os 93,90% da ModelNet40. Porém, quando aplicados os dados reais pré-processados as métricas chegaram a cerca de 89% mas não foram superiores às do benchmark ModelNet40. Considerando que a rede PAConv é baseada na construção de kernel, a utilização de dados reais e não ordenados prejudica sua perfomace. Além disso, a rede projetou seu modelo para nuvem de pontos com 1024 pontos, do mesmo modo que a rede SimpleView. Com os dados pré-processados, o resultado no subconjunto 3 aumentou de 78% para 87%, mesmo com nuvens de pontos com 2048 pontos. Um aumento significativo e embasado que a padronização das nuvem de pontos para o centro da origem interfere significativamente na rede PAConv, especificamente na convolução do kernel e montagem das matrizes de peso que a rede propõe.

Aplicada a rede ModalNet40, a GBNet atingiu 93% de acurácia porém ao inserir os dados com nuvens de pontos reais sua acurácia reduziu para 50% em seu melhor desempenho. Alguns fatores contribuem para esse resultado como a aplicação de dados reais, a quantidade de pontos na entrada da rede pois o autor utilizou 1024 para seus testes e o baixo recurso de informações geométricas em determinadas regiões da nuvem de pontos reais, pois as nuvens de pontos reais não possuem uma distribuição dos pontos regular. Com a etapa de pré-processamento da nuvem de pontos a GBNet reduziu ainda mais seu desempenho e se destacou com o pior desempenho, com acurácia e demais métricas abaixo de 38%. A redundância na geração das características locais da nuvem de pontos é um fator relevante para baixa performance, mesmo com os dados pré-processados que nas demais redes proporcionam um ganho no desempenho.

A tabela com os a validação de métricas com o teste estatístico Anova está representada na tabela 5 e 8. Como reportado na seção 3.7, o método estatístico Anova avalia se os métodos de classificação diferem ou são equivalentes de forma estatística, ou seja, a escolha

da rede importa ou não para fins de classificação. De acordo com as tabelas 5 e 8, o *p* é menor do que 0,05, dessa forma podemos considerar que ao menos uma rede difere de outra quando comparadas, implicando que existe um melhor do que a outra. E o teste estatístico se aplica em todas as métricas de avaliação pois em todos os casos o *p* é menor que 0,05. No entanto, o método de ANOVA não destaca qual o melhor ou pior método. Desse modo, o teste de Tukey é aplicado para quantificar de forma estatística o quanto os métodos de classificação das redes neurais são diferentes. O mesmo valor de referência *p* menor do que 0,05 é considerado para diferir. Porém se *p* é maior do que 0,05, é possível quantificar de forma estatística o quantos os métodos de classificação são equivalentes, que pode atingir no máximo 1. Nesse caso de estudo, o máximo é 0,9 pois o teste ANOVA indica que todos os métodos de classificação possuem uma diferença.

Os resultados do teste estatístico Tukey HSD são apresentados na tabela e 6 e 9. As tabelas são formadas por sete colunas, na primeira é o subconjunto referência, no caso a rede que será confrontado individualmente com as demais. Foram realizados 36 confrontos. Embaixo da rede referência está o valor de sua acurácia para facilitar a visualização ao comparar as redes. Na segunda coluna está o subconjunto2, em que estão presentes todas as redes que não foram confrontadas com a rede referência,. E as demais colunas são as métricas de avaliação adotadas na pesquisa. O primeiro destaque são as redes GBNet e 3DMedPT que diferem entre si e das demais. Com acurácia de 75,29% a 3DMedPT não obteve valores próximos dos melhores resultados e também esteve distante da rede GBNet que obteve 50,29%. A rede Lidar3DNet, com 84.80%, teve valor de acurácia próximo às redes CurveNet, LDGCNN, PAConv, PointNet, PVT e SimpleView e assim o valor de *p* foi destacado em preto na tabelas 6. A rede RSCCNN teve um bom desempenho e atingiu 100% de acurácia na classificação dos objetos 3D e ao ser confrontada com redes com acurácia acima de 88% demonstraram ter uma mínima equivalência estatística. As redes LGDCNN, CurveNet, PVT, PointNet e SimpleView alcançaram acurácia próxima de 90% e atingiram o valor máximo de 0,9 quando confrontadas. Na tabela 9 modificou em relação a tabela /6 pois com a etapa de pré-processamento as redes, sua maioria, melhorando seu desempenho e as redes Lidar3DNet, PointNet e PAConv alcançaram percentuais acima de 92%, aumentando assim a similaridade estatística com as demais redes que apresentavam essa acurácia. Dessa forma, o número do p-*value* em destaque aumentou e muitos com 0,900. Com esse ganho na performance, comprovadas pelas métricas de avaliação e testes estatísticos, é possível destacar que a etapa de pré-processamento proposta é relevante para tarefa de classificação de objetos 3D

com base em nuvens de pontos, utilizando dados reais.

Porém, para avaliar com afinco qual o método de classificação se destaca, o tempo médio para executar a predição de cada amostra foi calculado e está ilustrado nas figuras 20 e 21. A rede RSCNN, com 100% em todos as métricas, figurou entre as três redes que mais precisam de tempo para predizer a classe do objeto, junto com PVT e PointNet. Os tempos das redes Lidar3DNet, LDGCNN e SimpleView foram multiplicadas por 100 (x100) devido seu tempo de processamento ficarem bem abaixo das demais e ao plotar no gráfico a visualização seria prejudicada. Dessa forma as três são as redes com menor tempo médio para executar a predição da classe do objeto sendo assim as mais indicadas para aplicações que envolvem navegação de veículos autônomos e em tempo real. Considerando a análise da acurácia em conjunto com análise do tempo médio de predição, a rede Lidar3DNet obteve o melhor desempenho e assim, para dados reais, é o método mais indicado para classificar objetos 3D com base em nuvens de pontos.

Tabela 8 – Resultados do teste *one-way* ANOVA sobre os dados os **pré-processados (S_c)** e **reais**.

Métrica		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Acurácia	C(Redes)	8.00	0.96	0.12	56.53	3.49e-11
	Residual	18.00	0.03	0.00	-	-
F1-*Score*	C(Redes)	8.00	1.20	0.15	32.72	3.47e-09
	Residual	18.00	0.08	0.00	-	-
Precisão	C(nets)	8.00	1.10	0.13	23.99	4.32e-08
	Residual	18.00	0.10	0.00	-	-
Sensib.	C(nets)	8.00	1.12	0.14	46.13	1.96e-10
	Residual	18.00	0.05	0.00	-	-
Especif.	C(nets)	8.00	0.26	0.03	77.02	2.43e-12
	Residual	18.00	0.00	0.00	-	-

Fonte: Elaborado pelo autor.

4.3 Avaliação com o *dataset* sintético como entrada nas redes do estado da arte, com dados originais e pré-processados.

Na seção 4.1, a validação das arquitetura das redes neurais foram discutidas, na seção 4.2 foi analisado o desempenho das redes neurais desta pesquisa com o *dataset* real, com dados originais e pré-processados, como entrada das redes. Nesta seção é dissertado o desempenho das redes neurais com o uso das nuvens de pontos do *dataset* sintético, original e pré-processado,

Tabela 9 – Resultados do teste Tukey sobre os dados originais e **pré-processados (S_c)** e **reais**.

Subconjunto1	Subconjunto2	Acurácia	F1-*Score*	Precisão	Sensenb.	Especif.
3DMedPT *(Acc = 73,52%)*	Lidar3DNet	0.001	0.001	0.001	0.004	0.001
	CurveNet	0.001	0.001	0.001	0.004	0.001
	GBNet	0.001	0.001	0.004	0.002	0.001
	LDGCNN	0.001	0.001	0.001	0.002	0.001
	PAConv	0.001	0.003	0.001	0.009	0.001
	PointNet	0.025	0.030	0.053	0.051	0.010
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
Lidar3DNet *(Acc = 97,70%)*	CurveNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	**0.363**	**0.390**	**0.301**	**0.353**	**0.321**
	PointNet	**0.603**	**0.604**	**0.703**	**0.703**	**0.690**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
CurveNet *(Acc = 95,90%)*	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	**0.699**	**0.812**	**0.899**	**0.900**	**0.854**
	PointNet	**0.722**	**0.712**	**0.759**	**0.780**	**0.754**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	0.900
GBNet *(Acc = 38,59%)*	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
LDGCNN *(Acc = 96,82%)*	PAConv	**0.458**	**0.547**	**0.690**	**0.900**	**0.538**
	PointNet	**0.304**	**0.210**	**0.212**	**0.252**	**0.246**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
PAConv *(Acc = 88,88%)*	PointNet	**0.502**	**0.605**	**0.505**	**0.596**	**0.500**
	PVT	**0.411**	**0.486**	**0.652**	**0.900**	**0.495**
	RSCNN	**0.010**	**0.015**	**0.031**	**0.037**	**0.029**
	SimpleView	**0.346**	**0.526**	**0.626**	**0.820**	**0.554**
PointNet *(Acc = 92,85%)*	PVT	**0.583**	**0.540**	**0.510**	**0.545**	**0.530**
	RSCNN	**0.292**	**0.342**	**0.492**	**0.306**	**0.443**
	SimpleView	**0.472**	**0.407**	**0.490**	**0.432**	**0.460**
PVT *(Acc = 97,66%)*	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
RSCNN *(Acc = 100,00%)*	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**

como entrada das redes. O desempenho das redes neurais estão expostos nas tabelas 10 e 13.

Tabela 10 – Métricas, em porcentagem (%), de cada Rede Neural e em cada subconjuntos de teste sobre os dados **originais** e **sintéticos**.

Rede	Subconjunto	Acc	F1-*Score*	Prec.	Sensib.	Especif.
3DMedPT	1	33,92	33,85	33,85	33,84	66,96
	2	33,13	33,04	33,04	33,04	66,57
	3	**34,50**	34,52	34,52	34,52	67,23
Lidar3DNet	1	78,03	77,04	80,63	78,51	89,08
	2	75,49	74,56	78,59	75,94	87,83
	3	**78,43**	77,02	79,74	78,59	89,09
CurveNet	1	99,60	99,60	99,60	99,60	99,80
	2	99,60	99,60	99,60	99,60	99,80
	3	**99,80**	99,80	99,80	99,81	99,90
GBNet	1	**34,70**	28,13	36,36	34,29	67,15
	2	31,56	21,17	27,22	32,04	65,96
	3	31,96	20,33	22,63	32,53	66,28
LDGCNN	1	97,61	97,59	97,60	97,59	98,81
	2	97,81	97,77	97,79	97,76	98,91
	3	**99,40**	99,41	99,42	99,40	99,69
PAConv	1	**91,17**	91,14	91,23	91,10	95,62
	2	89,21	89,08	89,12	89,06	94,65
	3	79,60	78,87	80,25	79,19	89,89
PointNet	1	75,00	75,46	77,87	74,94	87,57
	2	80,49	81,86	83,96	81,64	90,84
	3	**81,49**	80,97	81,29	81,19	90,77
PVT	1	99,41	99,41	99,40	99,43	99,70
	2	**99,80**	99,80	99,81	99,80	99,90
	3	**99,80**	99,80	99,80	99,81	99,90
RSCNN	1	**77,45**	75,11	85,91	78,21	88,82
	2	74,70	70,83	83,82	75,56	87,46
	3	73,33	68,97	83,44	74,38	86,78
SimpleView	1	**99,60**	99,61	99,60	99,62	99,80
	2	98,23	98,23	98,24	98,24	99,12
	3	98,43	98,41	98,43	98,43	99,22

Fonte: Elaborado pelo autor.

Com os dados sintéticos como entrada, a rede RSCNN obteve o melhor desempenho, com 99,80%, assim como o teste anterior com os dados reais. Com os dados sintéticos originais a rede RSCNN atingiu métricas em torno de 70% e com a aplicação da etapa de pré-processamento proposto nas nuvens de pontos dos objetos 3D, o seu desempenho foi de 99,80% e se destacou

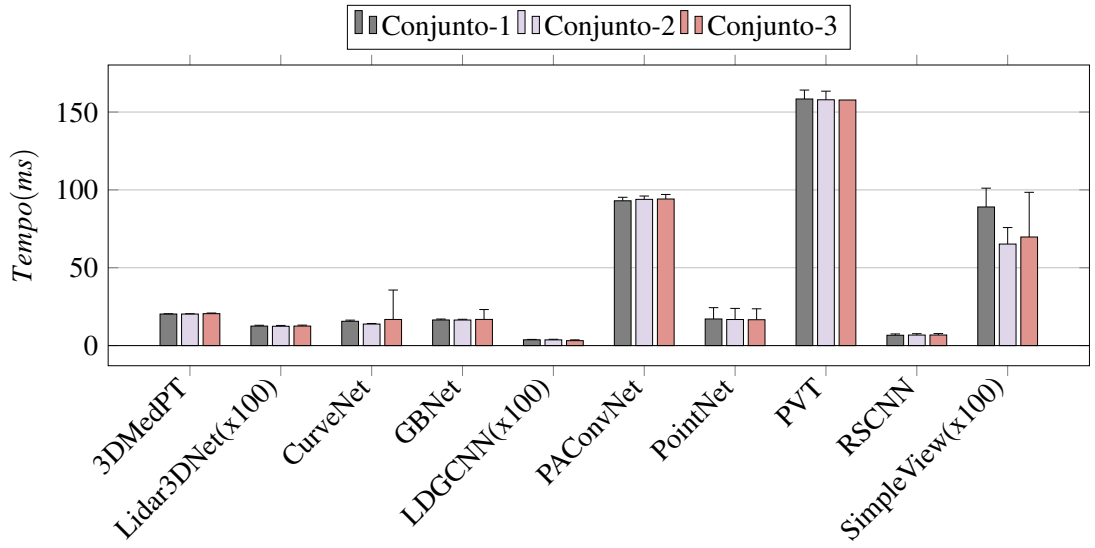


Figura 22 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento sobre os dados **originais** e **sintéticos**.

Fonte: Elaborada pelo autor.

Tabela 11 – Resultados do teste *one-way* ANOVA para experimentos sobre os dados **originais** e **sintéticos**.

Métrica		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Acurácia	C(Redes)	9.00	1.91	0.21	365.40	0.00
	Residual	20.00	0.01	0.00	-	-
F1-*Score*	C(Redes)	9.00	2.22	0.24	322.24	0.00
	Residual	20.00	0.01	0.00	-	-
Precisão	C(nets)	9.00	2.04	0.212	237.58	0.00
	Residual	20.00	0.01	0.00	-	-
Sensib.	C(nets)	9.00	1.91	0.21	359.98	0.00
	Residual	20.00	0.01	0.00	-	-
Especif.	C(nets)	9.00	0.47	0.05	379.79	0.00
	Residual	20.00	0.00	0.00	-	-

Fonte: Elaborado pelo autor.

com a melhor acurácia entre todas das redes do estudo, seguida das redes CurveNet e PVT, considerando todas as métricas das tabelas 10 e 13.

A diferença entre as três melhores redes pode ser encontrada no tempo médio de predição por amostra, ilustrado na figura 22 e 23, onde a RSCNN possui o melhor tempo com 6,5 ms, CurveNet 17 ms e PVT com 150 ms para predizer a classe de uma amostra. Esse elevado tempo pode ser justificado por seus métodos aplicarem transformações e clusterização nas nuvens de pontos.

As redes Lidar3DNet e LDGCNN se destacam por executar a predição em menor

Tabela 12 – Resultados do teste Tukey sobre os dados **originais** e **sintéticos**.

Subconjunto	Subconjunto2	Acurácia	F1-*Score*	Precisão	Sensenb.	Especif.
3DMedPT *(Acc = 34,50%)*	Lidar3DNet	0.001	0.001	0.001	0.001	0.001
	CurveNet	0.001	0.001	0.001	0.001	0.001
	GBNet	**0.900**	**0.900**	0.900	**0.900**	**0.900**
	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
Lidar3DNet *(Acc = 78,43%)*	CurveNet	0.001	0.001	0.001	0.001	0.001
	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	0.001	0.001	0.001	0.001	0.001
CurveNet *(Acc = 99,80%)*	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
GBNet *(Acc = 34,70%)*	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
LDGCNN *(Acc = 99,40%)*	PAConv	0.001	0.001	0.001	**0.006**	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
PAConv *(Acc = 91,17%)*	PointNet	0.034	0.029	0.012	0.042	0.040
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.004	0.001
PointNet *(Acc = 81,49%)*	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
PVT *(Acc = 99,80%)*	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
RSCNN *(Acc = 77,45,80%)*	SimpleView	0.001	0.001	0.001	0.001	0.001

tempo pois seus métodos de classificação são aplicados diretamente na nuvem de pontos dos objetos e sem nenhum tipo de agrupamento. Nas tabelas 22 e 23, o tempo das redes Lidar3DNet, LDGCNN e SimpleView foram multiplicadas por 100 (x100) pois seu tempo ficou bem abaixo das demais e a visualização seria prejudicada. Desse modo, as redes Lidar3DNet e LDGCNN obtiveram os melhores tempos e atingiram, em sua melhor configuração, acurácia de 94,11% e 99,60%, respectivamente. Desse modo, as redes Lidar3DNet e LDGCNN são as redes mais indicadas para aplicações que necessitam de uma predição rápida e com acurácia robusta.

As redes neurais LGDCNN, CurveNet, PVT e SimpleView obtiveram resultados acima de 99% de acurácia utilizando os dados originais e com pré-processamento, pois são redes que utilizam métodos de agrupamentos e transformações das nuvens de pontos. Além disso os testes de ANOVA, expostas nas tabelas 11 e 14, validam que as redes possuem resultado que diferem estatisticamente e o teste estatístico de Tukey quantifica de forma estatística o quanto são diferentes, representados na tabela 12 e 15. O valor de *p-value* menor que 0,05, exposta nas tabelas 11 e 14, confirmam que os métodos que diferem estatisticamente, segundo o teste de ANOVA. E nas tabela 12 e 15 os valores de *p-value* maior que 0,05 implicam que as redes LGDCNN, CurveNet, PVT e SimpleView são equivalentes. Quanto mais próximo do número um, maior é equivalência estatística entre as redes. O mesmo comportamento é possível perceber entre as redes GBNet e 3DMedPT que obtiveram os piores resultados. Na tabela 12, por exemplo, quando a rede 3DMedPT é a referência, com acurácia de 34,50%, apenas a rede GBNet é maior que 0,05, em todas as métricas, e estão destacadas em preto.

As redes Lidar3DNet, PointNet e RSCNN aplicam seus métodos de classificação diretamente nas nuvens de pontos e com os dados originais obtiveram resultados próximos a 80% de acurácia. As nuvens de pontos dos objetos 3D sintéticos são fechadas e dessa forma a metodologia de aplicar métodos diretamente na nuvem de pontos não apresenta resultados relevantes. Porém, com a aplicação da etapa de pré-processamento, proposta na pesquisa, nas nuvens de pontos sintéticos, a rede Lidar3DNet melhorou 16%, a aumentou 17% e a RSCNN subiu 25%. Dessa forma, todas ficaram acima de 98% de acurácia, inclusive as demais métricas, caracterizando assim redes capazes de indicar o objeto que pertence a classe, o que não pertence a classe e classificar com precisão as amostras positivas de forma correta. Esse ganho causado pela etapa de pré-processamento pode ser validado com testes estatísticos ANOVA e Tukey. O teste de ANOVA destacou que as redes são estatisticamente diferentes, como esplanado no parágrafo anterior. O teste de Tukey enfatiza de forma mais clara a equivalência estatística

entre as redes, com valor acima de 0.05 e a diferença das demais redes, com valores menores que 0,05. Na tabela 12, quando a rede Lidar é referência, com 78,43% de acurácia, os valores em negrito acima de 0,05 estão presente apenas nas redes PointNet e RSCNN. Após a etapa de pré-processamento, as redes Lidar3DNet, PointNet e RSCNN obtem um ganho relevante e suas acurácias ficam acima de 98%. Ao analisar mais uma vez o teste de Tukey, exposto na tabela 15, e tomar como referência a rede Lidar3DNet é notório que todas as redes, exceto as piores redes GBNet e 3DMetPT, estão em negito. Dessa forma, o teste estatístico valida o ganho proporcionado pela etapa de pré-processamento e quantifica estatisticamente o quanto as redes são semelhantes, pois seus resultados ficaram entre 98% e 99% de acurácia.

Assim como na aplicação com dados reais, as redes GBNet e 3DMedPT foram as redes neurais mais limitadas para o caso de estudo. Com acurácia, precisão, sensibilidade e F1-*Score* próximas a 35%, figuraram como os piores métodos de classificação de objetos 3D com base em nuvens de pontos. A métrica especificidade destaca a capacidade que o modelo tem de dizer quando uma amostra não pertence a classe, considerando que a probabilidade de acertar é 2/3 (66%) o valor de especificidade é coerente. Considerando o tempo médio de executar as predições as redes GBNet e 3DMedPT não tiverem resultados relevante para a aplicação em tempo real, como ilustra as figuras 22 e 23. A rede 3DMedPT é projetada para imagens médicas, especificamente vasos sanguíneos e a GBNet captura características geométricas das nuvens de pontos para classificação. A utilização de três classes de objetos, quantidade de pontos das nuvens de pontos e a abordagem para classificação dos objetos 3D dessas redes contribuíram para o baixo rendimento das redes na classificação de objetos 3D.

Os teste estatístico de Tukey destaca a diferença de rendimento da GBNet e 3DMedPT, como ilustra as figuras 12 e 15. Nas duas situações, com dados reais e pré-processados, o teste estatístico de Tukey não quantificou nenhum grau de semelhança estatística entre elas ou com qualquer outra rede.

4.4 Avaliação da variação da amostragem dos dados de entrada.

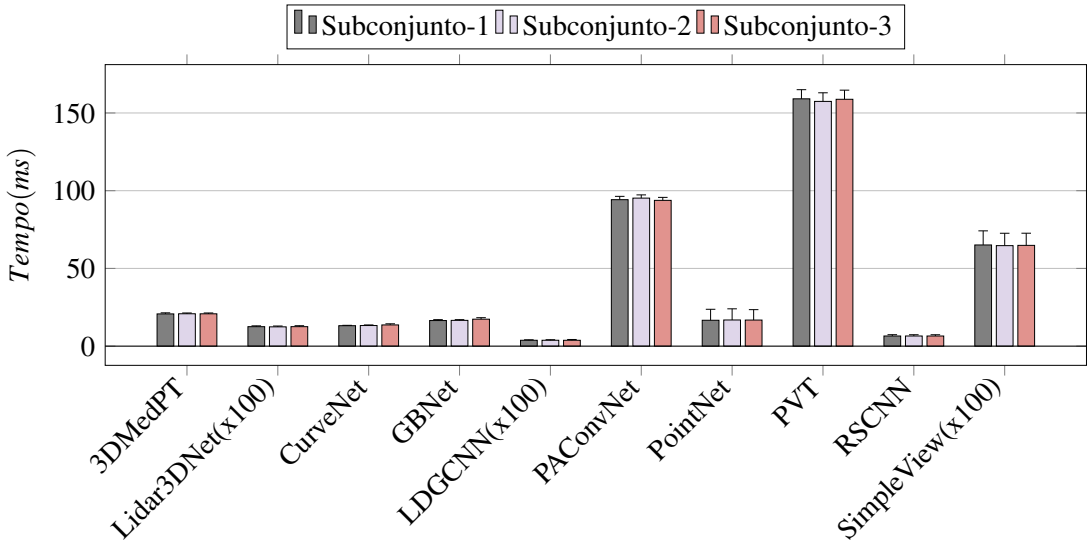
Nas seções anteriores foram discutidos os resultados das redes que desempenharam a melhor classificação de objetos 3D a partir da nuvem de pontos. Um ponto comum na configuração dessas redes é a quantidade de pontos dos objetos de entrada, especificamente todos os objetos possuem 2048 pontos. Nesta seção é gerada uma hipótese sobre a variação da quantidade de pontos dos objetos na entrada da rede. Para isso foi escolhida a rede proposta,

Tabela 13 – Métricas, em porcentagem (%), de cada Rede Neural e em cada subconjuntos de teste sobre os dados **pré-processamento (S_c)** e **sintéticos**.

Rede	**Subconjunto**	**Acc**	**F1-*Score***	**Prec.**	**Sensib.**	**Especif.**
3DMedPT	1_c	33,92	33,85	33,85	33,84	66,96
	2_c	33,72	33,66	33,72	33,65	66,86
	3_c	**36,07**	35,99	36,10	36,14	68,04
Lidar3DNet	1_c	93,92	93,83	93,91	93,85	96,96
	2_c	92,35	92,22	92,22	92,22	96,19
	3_c	**94,11**	94,07	94,17	94,06	97,07
CurveNet	1_c	**99,60**	99,60	99,60	99,61	99,80
	2_c	99,21	99,21	99,22	99,20	99,60
	3_c	99,21	99,22	99,21	99,23	99,60
GBNet	1_c	31,17	30,85	31,18	31,33	65,66
	2_c	29,60	28,56	28,75	29,83	64,88
	3_c	**31,56**	31,22	31,41	31,80	65,84
LDGCNN	1_c	**99,60**	99,60	99,59	99,60	99,80
	2_c	99,20	99,19	99,22	99,17	99,60
	3_c	99,40	99,41	99,42	99,40	99,69
PAConv	1_c	92,54	92,52	92,97	92,69	96,30
	2_c	97,45	97,44	97,45	97,45	98,73
	3_c	**98,23**	98,24	98,23	98,27	99,12
PointNet	1_c	97,63	97,62	97,64	97,65	98,82
	2_c	95,86	95,81	95,89	95,80	97,93
	3_c	**98,03**	98,04	98,03	98,04	99,01
PVT	1_c	99,41	99,41	99,40	99,43	99,70
	2_c	**99,60**	99,61	99,62	99,60	99,80
	3_c	99,21	99,21	99,20	99,23	99,61
RSCNN	1_c	**99,80**	99,80	99,80	99,81	99,90
	2_c	**99,80**	99,80	99,81	99,80	99,90
	3_c	**99,80**	99,80	99,81	99,80	99,89
SimpleView	1_c	**99,60**	99,61	99,60	99,62	99,80
	2_c	**99,60**	99,61	99,61	99,61	99,80
	3_c	**99,60**	99,61	99,60	99,62	99,80

Fonte: Elaborada pelo autor.

Figura 23 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento sobre os dados **pré-processamento (S_c)** e **sintéticos**.



Fonte: Elaborada pelo autor.

Tabela 14 – Resultados do teste *one-way* ANOVA para experimentos sobre os dados **pré-processamento (S_c)** e **sintéticos**.

Métrica		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Acurácia	C(Redes)	9.000	1.97	0.29	1079.72	0.00
	Residual	20.00	0.00	0.00	-	-
F1-*Score*	C(Redes)	9.00	2.00	0.23	829.73	0.00
	Residual	20.00	0.00	0.00	-	-
Precisão	C(nets)	9.000	1.97	0.21	1349.39	0.00
	Residual	20.00	0.00	0.00	-	-
Sensib.	C(nets)	9.00	1.96	0.21	1105.38	0.00
	Residual	20.00	0.00	0.00	-	-
Especif.	C(nets)	9.00	0.49	0.05	1094.62	0.00
	Residual	20.00	0.00	0.00	-	-

Fonte: Elaborado pelo autor.

Lidar3DNet, para esses testes, pois é a única que propôs dois *datasets* com nuvens de pontos reais e sintéticas. Na tabela 16 é avaliada a variação do número de pontos na entrada da rede Lidar3DNet, considerando as métricas de tempo de processamento por amostra e acurácia. Além disso. os dados foram executados utilizando recurso de GPU e CPU aplicado nos *dataset* com dados sintéticos e reais.

A variação do número de pontos na classificação de objetos 3D é importante pois aplicado a um sistema embarcado e em tempo real a quantidade do número de pontos pode interferir na velocidade e acurácia de classificação do objeto e como consequência proporcionar

Tabela 15 – Resultados do teste Tukey sobre os dados **pré-processamento (S_c)** e **sintéticos**.

Subconjunto	Subconjunto2	Acurácia	F1-*Score*	Precisão	Sensenb.	Especif.
3DMedPT *(Acc = 36,07%)*	Lidar3DNet	0.001	0.001	0.001	0.001	0.001
	CurveNet	0.001	0.001	0.001	0.001	0.001
	GBNet	**0.087**	**0.103**	**0.097**	0.021	**0.117**
	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
Lidar3DNet *(Acc = 94,11%)*	CurveNet	**0,171**	**0,247**	**0,181**	**0,087**	**0,060**
	GBNet	0,001	0,001	0,001	0,001	0,001
	LDGCNN	**0,457**	**0,497**	**0,454**	**0,416**	**0,412**
	PAConv	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	PointNet	**0,900**	**0,893**	**0,900**	**0,581**	**0,618**
	PVT	**0,066**	**0,087**	**0,073**	**0,012**	**0,012**
	RSCNN	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	SimpleView	**0,418**	**0,342**	**0,315**	**0,466**	**0,102**
CurveNet *(Acc = 99,60%)*	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	**0.198**	**0.199**	**0.350**	**0.139**	**0.202**
	PointNet	**0.670**	**0.662**	**0.795**	**0.556**	**0.650**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
GBNet *(Acc = 31,56%)*	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
LDGCNN *(Acc = 99,60%)*	PAConv	**0.181**	**0.183**	**0.329**	**0.124**	**0.188**
	PointNet	**0.642**	**0.637**	**0.772**	**0.522**	**0.628**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0,327**	**0,397**	**0,258**	**0,198**	**0,171**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
PAConv *(Acc = 98,23%)*	PointNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PVT	**0.180**	**0.180**	**324**	**0.124**	**0.180**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0,133**	**0,133**	**0,259**	**0,087**	**133**
PointNet *(Acc = 98,03%)*	PVT	**0.639**	**0.631**	**0.767**	**0.522**	**0.615**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.548**	**0.539**	**0.688**	**0.419**	**0.523**
PVT *(Acc = 99,60%)*	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
RSCNN *(Acc = 99,80%)*	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**

maior segurança ao sistema que pode auxiliar na navegação de veículos autônomos. Avaliando os resultados da tabela 16 todas as situações apresentaram acurácia acima de 90%. Ao analisar especificamente a variação da quantidade de pontos das nuvens é factível que interfere diretamente na acurácia e velocidade de processamento, tanto nos dados sintéticos quanto reais. Porém essa influência é mais incisiva nos dados sintéticos devido a rede Lidar3DNet ter melhor desempenho para dados reais. À medida que o número de pontos das nuvens dos objetos 3D aumentam a acurácia reduz, especificamente de 98% para 87%. Considerando os dados reais, os valores de tempo de processamento e acurácia não sofrem tanta interferência com a variação no número de pontos, reduzindo cerca de 2% na maior redução.

O uso de máquinas com unidade de processamento gráfico para classificar nuvens de pontos não é tão perceptível considerando nuvens de pontos com poucos pontos. Porém, a partir de 1024 pontos, o recurso de GPU contribui significativamente na velocidade de classificação aplicado em dados reais. Considerando os dados sintéticos, o tempo de processamento de uma forma geral é maior. E sem o recurso de GPU a nuvem do objeto com 8192 pontos o tempo chega a mais de 8 segundos e acurácia de 87%. Com o recurso de GPU, a acurácia mantém porém o tempo reduz em 6 segundos, aproximadamente. Em geral o tempo de processamento dos dados sintéticos são maiores por serem nuvens de pontos fechadas e a rede proposta tem o melhor desempenho em dados reais. Por fim, a configuração mais indicada para classificar objetos 3D é utilizando a rede LidarDNetV1 e com dados reais, com recurso de GPU aplicado a nuvens de pontos normalizadas com 64 pontos.

Tabela 16 – Resultados obtidos pela rede LidarDNetV1 no dataset proposto.

Base de dados	**Proces.**	**Métricas**	**Normalização**							
			64	**128**	**256**	**512**	**1024**	**2048**	**4096**	**8192**
Dados Sintéticos	**sem** GPU	Acc (%)	98.13	**98.26**	97.73	92.59	93.73	92.93	91.13	87.66
		Tmtr/a	1s 339μs	1s 350μs	1s 386μs	1s 483μs	1s 636μs	2s 1ms	3s 2ms	7s 5ms
		Tmtr / a	0s 81μs	0s 67μs	0s 87μs	0s 107μs	0s 131μs	0s 186μs	1s 382μs	1s 469μs
	com GPU	Acc (%)	98.00	**98.33**	97.73	95.99	95.46	94.11	90.53	87.46
		Tmtr/a	1s 383μs	1s 371μs	1s 388μs	1s 426μs	1s 465μs	1s 547μs	1s 703μs	2s 1ms
		Tmtr / a	0s 95μs	0s 88μs	0s 99μs	0s 91μs	0s 95μs	0s 115μs	0s 121μs	0s 167μs

Base de dados	**Proces.**	**Métricas**	**Normalização**							
			64	**128**	**256**	**512**	**1024**	**2048**	**4096**	**8192**
Dados Reais	**sem** GPU	Acc (%)	**98.48**	98.47	97.16	96.73	96.73	97.60	97.38	97.16
		Tmt/a	0s 343μs	0s 349μs	0s 414μs	0s 486μs	0s 696μs	1s 1ms	1s 2ms	2s 4ms
		Tmtr / a	0s 145μs	0s 144μs	0s 109μs	0s 174μs	0s 172μs	0s 254μs	0s 368μs	0s 535μs
	com GPU	Acc (%)	**98.47**	97.38	97.94	97.73	98.03	97.70	97.03	96.51
		Tmtr / a	0s 387μs	0s 401μs	0s 406μs	0s 423μs	0s 473μs	0s 542μs	0s 721μs	0s 1ms
		Tmts / a	0s 125μs	0s 158μs	0s 173μs	0s 155μs	0s 140μs	0s 196μs	0s 207μs	0s 255μs

4.5 Análise Geral dos Resultados.

Nesta seção é realizada uma análise geral das discussões realizadas nas seções 4.1, 4.2 e 4.3. Com a validação da configuração das arquiteturas das redes do estado da arte, discutidas na seção 4.1, a rede CurveNet obteve a melhor acurácia com o uso do *dataset* ModelNet40. Ao modificar os dados de entrada das redes neurais para o *dataset* com nuvens de pontos reais. Todas as redes reduziram sua acurácia, exceto a rede RSCNN que atingiu 100% de acurácia. Porém com a inserção da etapa de pré-processamento nas nuvens de pontos, todas as redes neurais elevaram seu rendimento, exceto a 3DMedPT e GBNet que possuem métodos de agrupamento e transformação das nuvens de pontos. Isso implica que a etapa de pré-processamento proposta não modifica o arranjo dos pontos dos objetos e consequentemente a sua forma do objeto 3D.

Com o uso das nuvens de pontos sintéticos na entrada das redes neurais, o comportamento das redes Lidar3DNet, CurveNet, LGDCNN, PAConv, Pointnet, PVT, RSCNN e SimpleView foram semelhantes com o uso das nuvens de pontos da ModelNet40 pois ambos utilizam dados sintéticos desenvolvidos em softwares. Dessa forma as nuvens de pontos possuem características semelhantes e assim validam o *dataset* proposto nesta pesquisa. A quantidade de classes do dataset proposto e a aplicação que as redes neurais 3DMedPT e GBNet foram projetadas podem ter interferido no desempenho e por isso não atingiram resultados semelhantes. A aplicação da etapa de pré-processamento nos dados sintéticos potencializou os resultados das redes Lidar3DNet, PointNet e RSCNN, do mesmo modo que ocorreu com a aplicação dos dados reais. Dessa forma é possível afirmar que o método de pré-processamento das nuvens de pontos proposta nesta pesquisa contempla nuvens de pontos reais, geradas por sensores como o Lidar, e nuvens pontos sintéticas, geradas por softwares como o Webots ou CAD.

Os testes estatísticos enfatizaram os resultados das seções 4.2 e 4.3, pois o método Anova destacou que todas as redes possuem métodos de classificação que diferem estatisticamente uma da outra e dessa forma existe uma superior ou inferior do que outra. As redes neurais que obtiveram métricas próximas, em percentual, o teste estatístico ANOVA indicou que seus métodos são equivalentes estatisticamente, por exemplo as redes Lidar3DNet e LGDCNN. Para complementar uma limitação do teste ANOVA, que é não identificar qual ou quais redes diferem das outras, o teste de diferença honestamente significativa, Tukey, foi aplicado para apresentar a diferença de significância estatísticas no uso de uma ou outra rede. E caso sejam equivalentes, quantificar a significância estatística entre as redes neurais, por exemplo as redes LidarNet3D, RSCNN e LDGCNN.

Considerando o experimento da variação da quantidade de pontos das nuvens de pontos, destacada na seção 4.4 é possível afirmar que ela interfere diretamente na perfomace e tempo de processamento dos dados. O uso de recursos da GPU é válido a partir de nuvens de pontos com mais de 1024, seja em nuvens de pontos com dados reais ou sintéticos. A maior acurácia com nuvens de pontos sintéticos foi de 98,33% utilizando normalização de 128 pontos e com nuvens de pontos reis foi de 98,47% com normalização de 64 pontos, ambas situações utilizando o recurso de GPU.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho propõe arquitetura baseada em perceptrom multicamadas para classificação de objetos 3D com base em nuvens de pontos. O estudo seleciona as principais redes neurais do estado da arte e as compara, utilizando cinco métricas de avaliação e dois métodos de avaliação estatística, com a rede proposta, Lidar3DNet, utilizando três *dataset* com dados reais e sintéticos.

Discussão gerada com a variação de dados na arquitetura proposta contribui para a sociedade científica na tarefa de classificação de nuvens de pontos, especificamente no desempenho de diferentes redes do estado da arte aplicando dados reais e simulados, etapas de pré-processamento e testes com cinco métricas diferentes e por fim dois testes estatísticos. No primeiro momento, a criação dos dez ambientes diferentes, um para cada rede neural artificial, foram configuradas com êxito e utilizado o *dataset* ModelNet40 como entrada. Sendo destaques as redes Curvenet e PVT com acurácia acima de 94%. Em seguida, as redes configuradas recebem como entrada os dados do *dataset* proposto com nuvens de pontos de objetos reais originais e com pré-processamento. Com os testes, é notório a existência de uma diferença significativa no desempenho das redes quando aplica-se dados reais e dados sintéticos. Todas as redes, exceto a 3DNetPT, tiveram ganhos, acima de 5%, nas métricas de avaliação e o destaque foi a rede Lidar3DNet que de 84,80% aumentou para 97,70%. A rede Lidar3DNet figurou entre os métodos com menor tempo de predição por amostra junto com a rede LDGCNN, porém a rede proposto teve a melhor acurácia e nos testes com dados reais é a mais indica para aplicações em sistemas embarcados, considerando as cinco métricas de avaliação e o tempo de processamento de classificação.

Nos testes com dados sintéticos, a etapa de pré-processamento também proporcionou ganhos as redes neurais da pesquisa, especificamente as redes Lidar3DNet, RSCNN e PointNet com ganhos acima de 15%. Com exceção da rede RSCNN, que atingiu acurácia de 100%, todas obtiveram resultados inferiores se comparados com a situação anterior.

Os métodos de análises estatísticas propostas, ANOVA e Turkey, contribuíram para o embasamento e validação dos diversos resultados gerados e comparações entres os métodos de pesquisa. O destaque foi a distinção das piores redes,GNet e 3DMedPT, para a tarefa de classificação de objetos 3D. E também destacaram a evolução do ganho das redes Lidar3DNet, RSCNN e PointNet, onde com os dados sintéticos nenhuma equivalência estatísticas existia entre elas e as demais redes e com o ganho o teste de Turkey mostrou o grau de equivalência

estatísticas entre elas e as demais redes com métricas acimas de 90%. Os testes com aplicação de nuvens de pontos reais e sintéticas destacou as limitações das redes GBNet, PACon e 3DMedPT, com acurácias abaixo de 90%, com nuvens de pontos reais e assim não são indicadas para aplicação em sistemas embarcados e em tempo real. Seus métodos de classificação de nuvens de pontos atingiram baixas taxas de acerto e com um tempo de processamento alto, considerando aplicações em sistemas embarcados.

Para analisar e validar estatisticamente os resultados, os testes de Anova e Tukeys foram aplicados. Com essa análise a rede GBNet é a mais indicada, estatisticamente, para classificar objetos 3D. Seguida das redes PointNet e 3DMedPT que apresentaram significância estatística para classificar objetos 3D porém as duas estatisticamente são semelhantes. Por fim, a rede proposta, LidarNetV1, foi utilizada para classificar nuvens de pontos com diferentes quantidade de pontos. Além do uso de dados sintéticos e reais, também foi avaliado o tempo de processamento com recurso de GPU e CPU. Com essa configuração destacamos que o os dados sintéticos tiveram 98% de acurácia utilizando nuvens de pontos com 128 pontos e com os dados reais a mesma acurácia foi alcançada utilizando nuvens de pontos com 64 pontos.

Nesse contexto, todos os objetivos listados no início da pesquisa foram contemplados com êxito, contribuindo para a área de navegação de veículos autônomos, especificamente na tarefa de classificação de objetos 3D com base em nuvens de pontos.

Para trabalhos futuros, planeja-se aquisitar novas amostrar e adicionar mais classes no *dataset* proposto, avaliar a interferência da variação de pontos em mais redes relevantes do estado da arte,implementar uma segunda versão da Lidar3DNet e adicionar segmentação de objetos 3D e avaliar a viabilidade dos modelos em aplicações com sistemas embarcadas, com ênfase em internet das Coisas e Computação de Borda.

REFERÊNCIAS

AGARAP, A. F. Deep learning using rectified linear units (relu). **CoRR**, abs/1803.08375, 2018.

AHMED, I.; SEADAWY, A. R.; LU, D. M-shaped rational solitons and their interaction with kink waves in the fokas–lenells equation. **Physica Scripta**, IOP Publishing, v. 94, n. 5, p. 055205, 2019.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. **2017 international conference on engineering and technology (ICET)**. Antalya,TR, 2017. p. 1–6.

ALNAGGAR, Y. A.; AFIFI, M.; AMER, K.; ELHELW, M. Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds. In: **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision**. Waikoloa,HI: IEEE, 2021. p. 1800–1809.

AMARADI, P.; SRIRAMOJU, N.; DANG, L.; TEWOLDE, G. S.; KWON, J. Lane following and obstacle detection techniques in autonomous driving vehicles. In: IEEE. **2016 IEEE International Conference on Electro Information Technology (EIT)**. Grand Forks,ND, 2016. p. 0674–0679.

ANADU, D.; MUSHAGALUSA, C.; ALSBOU, N.; ABUABED, A. S. Internet of things: Vehicle collision detection and avoidance in a vanet environment. In: IEEE. **2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)**. Houston,TX: IEEE, 2018. p. 1–6.

AOKI, Y.; GOFORTH, H.; SRIVATSAN, R. A.; LUCEY, S. Pointnetlk: Robust & efficient point cloud registration using pointnet. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. Long Beach,CA: IEEE Xplore, 2019. p. 7163–7172.

ARBELAEZ, P.; MAIRE, M.; FOWLKES, C.; MALIK, J. Contour detection and hierarchical image segmentation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 33, n. 5, p. 898–916, 2010.

ARMENI, I.; SENER, O.; ZAMIR, A. R.; JIANG, H.; BRILAKIS, I.; FISCHER, M.; SAVARESE, S. 3d semantic parsing of large-scale indoor spaces. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Las Vegas,NV: IEEE.

ASSAD-UZ-ZAMAN, M.; ISLAM, M. R.; RAHMAN, M. H.; WANG, Y.-C.; MCGONIGLE, E. Kinect controlled nao robot for telerehabilitation. **Journal of Intelligent Systems**, De Gruyter, v. 30, n. 1, p. 224–239, 2021.

BOULCH, A.; SAUX, B. L.; AUDEBERT, N. Unstructured point cloud semantic labeling using deep segmentation networks. **3DOR**, v. 2, p. 7, 2017.

BOURKE, P. Interpolation methods. **Miscellaneous: projection, modelling, rendering**, v. 1, n. 10, 1999.

BRONSTEIN, M. M.; BRUNA, J.; LECUN, Y.; SZLAM, A.; VANDERGHEYNST, P. Geometric deep learning: going beyond euclidean data. **IEEE Signal Processing Magazine**, IEEE, v. 34, n. 4, p. 18–42, 2017.

CAESAR, H.; BANKITI, V.; LANG, A. H.; VORA, S.; LIONG, V. E.; XU, Q.; KRISHNAN, A.; PAN, Y.; BALDAN, G.; BEIJBOM, O. nuscenes: A multimodal dataset for autonomous driving. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. Long Beach,CA: IEEE Xplore, 2020. p. 11621–11631.

CAO, L.; GORESHNIK, I.; COVENTRY, B.; CASE, J. B.; MILLER, L.; KOZODOY, L.; CHEN, R. E.; CARTER, L.; WALLS, A. C.; PARK, Y.-J. *et al.* De novo design of picomolar sars-cov-2 miniprotein inhibitors. **Science**, American Association for the Advancement of Science, v. 370, n. 6515, p. 426–431, 2020.

CHANG, A. X.; FUNKHOUSER, T.; GUIBAS, L.; HANRAHAN, P.; HUANG, Q.; LI, Z.; SAVARESE, S.; SAVVA, M.; SONG, S.; SU, H. *et al.* Shapenet: An information-rich 3d model repository. **arXiv preprint arXiv:1512.03012**, 2015.

CHEN, T.-H.; CHANG, T. S. Rangeseg: range-aware real time segmentation of 3d lidar point clouds. **IEEE Transactions on Intelligent Vehicles**, IEEE, v. 7, n. 1, p. 93–101, 2021.

CHEN, X.; KUNDU, K.; ZHANG, Z.; MA, H.; FIDLER, S.; URTASUN, R. Monocular 3d object detection for autonomous driving. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Las Vegas,NV: CVPR, 2016. p. 2147–2156.

CHEN, X.; MA, H.; WAN, J.; LI, B.; XIA, T. Multi-view 3d object detection network for autonomous driving. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Honolulu,HI: CVPR, 2017. p. 1907–1915.

CHEN, Z.; ZHANG, J.; TAO, D. Progressive lidar adaptation for road detection. **IEEE/CAA Journal of Automatica Sinica**, IEEE, v. 6, n. 3, p. 693–702, 2019.

DAI, A.; CHANG, A. X.; SAVVA, M.; HALBER, M.; FUNKHOUSER, T.; NIESSNER, M. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Honolulu,HI: CVPR, 2017. p. 5828–5839.

DENK, T. I.; REISSWIG, C. Bertgrid: Contextualized embedding for 2d document representation and understanding. **arXiv preprint arXiv:1909.04948**, 2019.

FAN, J.; BOCUS, M. J.; HOSKING, B.; WU, R.; LIU, Y.; VITYAZEV, S.; FAN, R. Multi-scale feature fusion: Learning better semantic segmentation for road pothole detection. In: IEEE. **2021 IEEE International Conference on Autonomous Systems (ICAS)**. Montréal,CA: IEE, 2021. p. 1–5.

FAN, R.; WANG, H.; CAI, P.; LIU, M. Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection. In: **European Conference on Computer Vision**. Honolulu,HI: CVPR. p. 340–356.

FERREIRA, A. R.; SILVA, A. C. Virtual reconstruction of objects by point cloud capture to measurement of density parameters using low cost device. In: IEEE. **2021 16th Iberian Conference on Information Systems and Technologies (CISTI)**. Chaves,PT: CVPR, 2021. p. 1–5.

FILHO, P. P. R.; SILVA, S. P. P. da; OHATA, E. F.; ALMEIDA, J. S.; SOUSA, P. H. F. de; NASCIMENTO, N. M. M.; SILVA, F. H. dos S. A new strategy for mobile robots localization

based on omnidirectional sonar images and machine learning. In: SBC. **Anais Estendidos da XXXII Conference on Graphics, Patterns and Images**. Rio de Janeiro,RJ, 2019. p. 168–171.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. online: MIT Press, 2016. http://www.deeplearningbook.org.

GOYAL, A.; LAW, H.; LIU, B.; NEWELL, A.; DENG, J. **Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline**. 2021.

GUO, L.; WANG, P. Art product design and vr user experience based on iot technology and visualization system. **Journal of Sensors**, Hindawi, v. 2021, 2021.

HACKEL, T.; SAVINOV, N.; LADICKY, L.; WEGNER, J. D.; SCHINDLER, K.; POLLEFEYS, M. Semantic3d. net: A new large-scale point cloud classification benchmark. **arXiv preprint arXiv:1704.03847**, 2017.

HAFEEZ, S.; KATHIRISETTY, N. Effects and comparison of different data pre-processing techniques and ml and deep learning models for sentiment analysis: Svm, knn, pca with svm and cnn. In: IEEE. **2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)**. Hyderabad,IN, 2022. p. 1–6.

HAO, C.-Y.; CHEN, M.-H.; CHOU, T.-Y.; LIN, C.-W. Design of a resource-oriented framework for point cloud semantic annotation with deep learning. In: IEEE. **2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)**. Laguna Hills,CA, 2018. p. 228–233.

HAYKIN, S. **Redes neurais: princípios e prática**. São Paulo,BR: Bookman Editora, 2001.

HU, Z.; ZHEN, M.; BAI, X.; FU, H.; TAI, C.-l. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In: SPRINGER. **European Conference on Computer Vision**. online, 2020. p. 222–239.

HUA, B.-S.; PHAM, Q.-H.; NGUYEN, D. T.; TRAN, M.-K.; YU, L.-F.; YEUNG, S.-K. Scenenn: A scene meshes dataset with annotations. In: IEEE. **2016 fourth international conference on 3D vision (3DV)**. Stanford,CA, 2016. p. 92–101.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. **International conference on machine learning**. Mountain View,CA, 2015. p. 448–456.

IZATT, G.; MIRANO, G.; ADELSON, E.; TEDRAKE, R. Tracking objects with point clouds from vision and touch. In: IEEE. **2017 IEEE International Conference on Robotics and Automation (ICRA)**. Marina Bay Sands,SG, 2017. p. 4000–4007.

KAZHDAN, M.; FUNKHOUSER, T.; RUSINKIEWICZ, S. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In: **Symposium on geometry processing**. Goslar,DE: Eurographics Association, 2003. v. 6, p. 156–164.

KOTB, A.; HASSAN, S.; HASSAN, H. A comparative study among various algorithms for lossless airborne lidar data compression. In: IEEE. **2018 14th International Computer Engineering Conference (ICENCO)**. Cairo,EG, 2018. p. 17–21.

KOVÁCS, Z. L. **Redes neurais artificiais**. Sao Paula,SP: Editora Livraria da Fisica, 2002.

LANDRIEU, L.; SIMONOVSKY, M. Large-scale point cloud semantic segmentation with superpoint graphs. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. San Juan,PR: IEEE, 2018. p. 4558–4567.

LANG, A. H.; VORA, S.; CAESAR, H.; ZHOU, L.; YANG, J.; BEIJBOM, O. Pointpillars: Fast encoders for object detection from point clouds. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. Long Beach,CA: IEEE, 2019. p. 12697–12705.

LANGER, F.; MILIOTO, A.; HAAG, A.; BEHLEY, J.; STACHNISS, C. Domain transfer for semantic segmentation of lidar data using deep neural networks. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Las Vegas, NV, 2020. p. 8263–8270.

LECUN, Y.; BOSER, B.; DENKER, J.; HENDERSON, D.; HOWARD, R.; HUBBARD, W.; JACKEL, L. Handwritten digit recognition with a back-propagation network. **Advances in neural information processing systems**, v. 2, 1989.

LEE, S.; KO, H.; HAHN, H. **Multisensor fusion and integration for intelligent systems**. Seoul,UG: Springer, 2009.

LI, B. 3d fully convolutional network for vehicle detection in point cloud. In: IEEE. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Vancouver,BC, 2017. p. 1513–1518.

LI, J.; CHEN, B. M.; LEE, G. H. So-net: Self-organizing network for point cloud analysis. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Seattle,WA: CVPR, 2018. p. 9397–9406.

LI, R.; LI, X.; FU, C.-W.; COHEN-OR, D.; HENG, P.-A. Pu-gan: a point cloud upsampling adversarial network. In: **Proceedings of the IEEE/CVF international conference on computer vision**. Softcover: IEEE, 2019. p. 7203–7212.

LIU, Y.; FAN, B.; XIANG, S.; PAN, C. **Relation-Shape Convolutional Neural Network for Point Cloud Analysis**. 2019.

LUO, P.; WANG, X.; SHAO, W.; PENG, Z. Towards understanding regularization in batch normalization. **arXiv preprint arXiv:1809.00846**, 2018.

LUO, R. C.; KAY, M. G. Multisensor integration and fusion in intelligent systems. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, v. 19, n. 5, p. 901–931, 1989.

MATURANA, D.; SCHERER, S. Voxnet: A 3d convolutional neural network for real-time object recognition. In: IEEE. **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Hamburg,DE, 2015. p. 922–928.

MEI, J.; ZHU, A. Z.; YAN, X.; YAN, H.; QIAO, S.; CHEN, L.-C.; KRETZSCHMAR, H. Waymo open dataset: Panoramic video panoptic segmentation. In: SPRINGER. **European Conference on Computer Vision**. Dan Pnorama Hotel, Tel Aviv, 2022. p. 53–72.

MICHEL, O. Cyberbotics ltd. webots™: professional mobile robot simulation. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 1, n. 1, p. 5, 2004.

MILIOTO, A.; BEHLEY, J.; MCCOOL, C.; STACHNISS, C. Lidar panoptic segmentation for autonomous driving. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Las Vegas,NV, 2020. p. 8505–8512.

MILIOTO, A.; VIZZO, I.; BEHLEY, J.; STACHNISS, C. Rangenet++: Fast and accurate lidar semantic segmentation. In: IEEE. **2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Macau,CH: IEEE, 2019. p. 4213–4220.

MOHAMMADI, S. S.; WANG, Y.; BUE, A. D. Pointview-gcn: 3d shape classification with multi-view point clouds. In: IEEE. **2021 IEEE International Conference on Image Processing (ICIP)**. Anchorage,AL, 2021. p. 3103–3107.

MOHANTY, A.; MAHAPATRA, S.; BHANJA, U. Traffic congestion detection in a city using clustering techniques in vanets. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 13, n. 2, p. 884–891, 2019.

MOHAPATRA, S.; YOGAMANI, S.; GOTZIG, H.; MILZ, S.; MADER, P. Bevdetnet: bird's eye view lidar point cloud based real-time 3d object detection for autonomous driving. In: IEEE. **2021 IEEE International Intelligent Transportation Systems Conference (ITSC)**. Indianapolis,IN, 2021. p. 2809–2815.

NAGY, I.; ONIGA, F. Free space detection from lidar data based on semantic segmentation. In: IEEE. **2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)**. Cluj-Napoca,RO, 2021. p. 95–100.

ONO, T.; EGUCHI, R.; TAKAHASHI, M. Dynamic motion tracking based on point cloud matching with personalized body segmentation. In: IEEE. **2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)**. New York City,NY, 2020. p. 61–67.

PASZKE, A.; CHAURASIA, A.; KIM, S.; CULURCIELLO, E. Enet: A deep neural network architecture for real-time semantic segmentation. **arXiv preprint arXiv:1606.02147**, 2016.

QI, C. R.; LIU, W.; WU, C.; SU, H.; GUIBAS, L. J. Frustum pointnets for 3d object detection from rgb-d data. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Salt Lake City, UT: IEEE/CVF, 2018. p. 918–927.

QI, C. R.; SU, H.; MO, K.; GUIBAS, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Honolulu,HI: IEEE/CVF, 2017. p. 652–660.

QI, C. R.; SU, H.; NIESSNER, M.; DAI, A.; YAN, M.; GUIBAS, L. J. Volumetric and multi-view cnns for object classification on 3d data. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Las Vegas, NV: IEEE, 2016. p. 5648–5656.

QIU, S.; ANWAR, S.; BARNES, N. **Geometric Back-projection Network for Point Cloud Classification**. 2021.

QUEIROZ, R. L. D.; CHOU, P. A. Compression of 3d point clouds using a region-adaptive hierarchical transform. **IEEE Transactions on Image Processing**, IEEE, v. 25, n. 8, p. 3947–3956, 2016.

RAN, H.; LIU, J.; WANG, C. Surface representation for point clouds. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. New Orleans,LA: IEEE, 2022. p. 18942–18952.

RAVISHANKAR, P. G.; LOPEZ, A. M.; SANCHEZ, G. M. Unstructured road segmentation using hypercolumn based random forests of local experts. **arXiv preprint arXiv:2207.11523**, 2022.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: ____. **Encyclopedia of Database Systems**. New York, NY: Springer New York, 2016. p. 1–7. ISBN 978-1-4899-7993-3. Disponível em: https://doi.org/10.1007/978-1-4899-7993-3_565-2.

REIZENSTEIN, J.; SHAPOVALOV, R.; HENZLER, P.; SBORDONE, L.; LABATUT, P.; NOVOTNY, D. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. virtual: IEEE Xplore, 2021. p. 10901–10911.

RORIZ, R.; CABRAL, J.; GOMES, T. Automotive lidar technology: A survey. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, 2021.

ROSELL-POLO, J. R.; GREGORIO, E.; GENÉ, J.; LLORENS, J.; TORRENT, X.; ARNÓ, J.; ESCOLA, A. Kinect v2 sensor-based mobile terrestrial laser scanner for agricultural outdoor applications. **IEEE/ASME Transactions on Mechatronics**, IEEE, v. 22, n. 6, p. 2420–2427, 2017.

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.

RUKHOVICH, D.; VORONTSOVA, A.; KONUSHIN, A. Fcaf3d: Fully convolutional anchor-free 3d object detection. **arXiv preprint arXiv:2112.00322**, 2021.

RUMELHART, D. E.; DURBIN, R.; GOLDEN, R.; CHAUVIN, Y. Backpropagation: The basic theory. **Backpropagation: Theory, architectures and applications**, Lawrence Erlbaum Hillsdale, NJ, USA, p. 1–34, 1995.

RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). In: IEEE. **2011 IEEE international conference on robotics and automation**. Shanghai,CN: IEEE, 2011. p. 1–4.

SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: SPRINGER. **International conference on artificial neural networks**. Thessaloniki,GR, 2010. p. 92–101.

SHEN, D.; WU, G.; SUK, H.-I. Deep learning in medical image analysis. **Annual review of biomedical engineering**, NIH Public Access, v. 19, p. 221, 2017.

SHI, B.; BAI, S.; ZHOU, Z.; BAI, X. Deeppano: Deep panoramic representation for 3-d shape recognition. **IEEE Signal Processing Letters**, IEEE, v. 22, n. 12, p. 2339–2343, 2015.

SICK, A. Lms5xx laser measurement sensors operating instructions. **Sick AG: Waldkirch, Germany**, 2015.

SONG, S.; LICHTENBERG, S. P.; XIAO, J. Sun rgb-d: A rgb-d scene understanding benchmark suite. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Boston, MA: IEEE, 2015. p. 567–576.

SONG, S.; XIAO, J. Deep sliding shapes for amodal 3d object detection in rgb-d images. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Las Vegas, NV: IEEE, 2016. p. 808–816.

SOUSA, P. H. F. de; ALMEIDA, J. S.; OHATA, E. F.; NOGUEIRA, F. G.; TORRICO, B. C.; ALBUQUERQUE, V. H. C. de; HASSAN, M. M.; KUMAR, N.; HASSAN, M. R.; FILHO, P. P. R. Intelligent 3d objects classification for vehicular ad hoc network based on lidar and deep learning approaches. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, 2021.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html.

SU, H.; MAJI, S.; KALOGERAKIS, E.; LEARNED-MILLER, E. Multi-view convolutional neural networks for 3d shape recognition. In: **Proceedings of the IEEE international conference on computer vision**. Santiago,CL: IEEE, 2015. p. 945–953.

SUN, G.; WANG, X. Three-dimensional point cloud reconstruction and morphology measurement method for greenhouse plants based on the kinect sensor self-calibration. **Agronomy**, Multidisciplinary Digital Publishing Institute, v. 9, n. 10, p. 596, 2019.

SUN, P.; KRETZSCHMAR, H.; DOTIWALLA, X.; CHOUARD, A.; PATNAIK, V.; TSUI, P.; GUO, J.; ZHOU, Y.; CHAI, Y.; CAINE, B. *et al.* Scalability in perception for autonomous driving: Waymo open dataset. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. Seattle,DC: IEEE, 2020. p. 2446–2454.

TCHAPMI, L.; CHOY, C.; ARMENI, I.; GWAK, J.; SAVARESE, S. Segcloud: Semantic segmentation of 3d point clouds. In: IEEE. **2017 international conference on 3D vision (3DV)**. Qingdao,CN, 2017. p. 537–547.

TORAL, R. l.; CHAKRABARTI, A. Generation of gaussian distributed random numbers by using a numerical inversion method. **Computer physics communications**, Elsevier Science, v. 74, n. 3, p. 327–334, 1993.

UY, M. A.; PHAM, Q.-H.; HUA, B.-S.; NGUYEN, T.; YEUNG, S.-K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: **Proceedings of the IEEE/CVF international conference on computer vision**. Seoul,KOR: IEEE, 2019. p. 1588–1597.

WAN, E. A.; MERWE, R. V. D. The unscented kalman filter for nonlinear estimation. In: IEEE. **Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)**. Lake Louise,AB, 2000. p. 153–158.

WANG, H.; FAN, R.; SUN, Y.; LIU, M. Applying surface normal information in drivable area and road anomaly detection for ground mobile robots. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. online, 2020. p. 2706–2711.

WANG, W.; YU, R.; HUANG, Q.; NEUMANN, U. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. Salt Lake City, UT: IEEE, 2018. p. 2569–2578.

WANG, Y.; MENKOVSKI, V.; HO, I. W.-H.; PECHENIZKIY, M. Vanet meets deep learning: The effect of packet loss on the object detection performance. In: IEEE. **2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)**. Kuala Lumpur, MY, 2019. p. 1–5.

WU, B.; ZHOU, X.; ZHAO, S.; YUE, X.; KEUTZER, K. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In: IEEE. **2019 International Conference on Robotics and Automation (ICRA)**. Sri Lanka,CO, 2019. p. 4376–4382.

WU, Z.; SONG, S.; KHOSLA, A.; YU, F.; ZHANG, L.; TANG, X.; XIAO, J. 3d shapenets: A deep representation for volumetric shapes. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Boston, MA: IEEE, 2015. p. 1912–1920.

XIANG, T.; ZHANG, C.; SONG, Y.; YU, J.; CAI, W. **Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis**. 2021.

XU, M.; DING, R.; ZHAO, H.; QI, X. **PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds**. 2021.

XU, Y.; NASCIMENTO, N. M. M.; SOUSA, P. H. F. de; NOGUEIRA, F. G.; TORRICO, B. C.; HAN, T.; JIA, C.; FILHO, P. P. R. Multi-sensor edge computing architecture for identification of failures short-circuits in wind turbine generators. **Applied Soft Computing**, Elsevier, v. 101, p. 107053, 2021.

YU, J.; ZHANG, C.; WANG, H.; ZHANG, D.; SONG, Y.; XIANG, T.; LIU, D.; CAI, W. **3D Medical Point Transformer: Introducing Convolution to Attention Networks for Medical Point Cloud Analysis**. 2021.

ZHANG, C.; WAN, H.; SHEN, X.; WU, Z. Pvt: Point-voxel transformer for point cloud learning. **arXiv preprint arXiv:2108.06076**, 2021.

ZHANG, C.; WAN, H.; SHEN, X.; WU, Z. **PVT: Point-Voxel Transformer for Point Cloud Learning**. 2022.

ZHANG, K.; HAO, M.; WANG, J.; CHEN, X.; LENG, Y.; SILVA, C. W. de; FU, C. Linked dynamic graph CNN: Learning through point cloud by linking hierarchical features. In: **2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)**. IEEE, 2021. Disponível em: https://doi.org/10.1109/m2vip49856.2021.9665104.

ZHOU, Y.; TUZEL, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. Salt Lake City, UT: IEEE, 2018. p. 4490–4499.