**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE CIÊNCIAS E TECNOLOGIA**

**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

**MESTRADO ACADÊMICO EM ENGENHARIA ELÉTRICA**

**FRANCISCO HÉRCULES DOS SANTOS SILVA**

# UMA ABORDAGEM PARA RECONHECIMENTO DE OBJETOS EM 3D A PARTIR DE NUVENS DE PONTOS CAPTURADAS COM *LIDAR* UTILIZANDO APRENDIZAGEM PROFUNDA

**FORTALEZA**

**2022**

FRANCISCO HÉRCULES DOS SANTOS SILVA

# UMA ABORDAGEM PARA RECONHECIMENTO DE OBJETOS EM 3D A PARTIR DE NUVENS DE PONTOS CAPTURADAS COM *LIDAR* UTILIZANDO APRENDIZAGEM PROFUNDA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências e Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Sinais e Sistemas

Orientador: Prof. Dr. Pedro Pedrosa Rebouças Filho

FORTALEZA

2022

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

---

S58a Silva, Francisco Hércules dos Santos.
Uma abordagem para Reconhecimento de Objetos em 3D a partir de nuvens de pontos capturadas com LiDAR utilizando Aprendizagem Profunda / Francisco Hércules dos Santos Silva. – 2022.
61 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2022.
Orientação: Prof. Dr. Pedro Pedrosa Rebouças Filho.

1. LiDAR. 2. Aprendizagem profunda. 3. Nuvem de Pontos. 4. Pré-processamento. 5. Validação cruzada. I. Título.

CDD 621.3

---

FRANCISCO HÉRCULES DOS SANTOS SILVA

UMA ABORDAGEM PARA RECONHECIMENTO DE OBJETOS EM 3D A PARTIR DE NUVENS DE PONTOS CAPTURADAS COM *LIDAR* UTILIZANDO APRENDIZAGEM PROFUNDA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências e Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Sinais e Sistemas

Aprovada em: 19 de Abril de 2022

BANCA EXAMINADORA

---

Prof. Dr. Pedro Pedrosa Rebouças Filho (Orientador)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Fabrício Gonzalez Nogueira
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Leandro Bezerra Marinho
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

## AGRADECIMENTOS

Aos meus pais, por todo amor, carinho e apoio moral que sempre me prestaram em todas as etapas de minha vida.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo financiamento de parte da pesquisa do mestrado.

Ao meu orientador, Prof. Dr. Pedro Pedrosa Rebouças Filho, pelo orientação, companheirismo, conselhos e ensinamentos ao decorrer desta pesquisa.

Aos professores e colaboradores de pesquisa do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará (PPGEE-UFC).

À todos os amigos e parceiros do Laboratório de Processamento de Imagens, Sinais e Computação Aplicada (LAPISCO), que contribuíram diretamente para minha evolução pessoal e profissional.

Assim como a todos aqueles que contribuíram de alguma maneira com o desenvolvimento desta pesquisa.

## RESUMO

Este trabalho propõe o uso de Aprendizagem Profunda no reconhecimento de objetos 3D capturados com *Light Detection and Ranging* (LiDAR), incluindo uma etapa de pré-processamento. Além disso, utiliza um conjunto de dados do mundo real que contém três classes. A etapa de pré-processamento é proposta para normalizar a quantidade de pontos de cada nuvem capturada, contornando as discrepâncias de dimensionalidade entre o conjunto de dados proposto e os conjuntos existentes na literatura, como o ModelNet. Assim, uma validação cruzada com três partições é realizada na etapa de classificação, visando mensurar os desempenhos das redes em diferentes métricas de avaliação a partir de diferentes arranjos do conjunto. Em um primeiro experimento com dez redes selecionadas a rede de melhor desempenho foi a RS-CNN atingindo 98,64% de acurácia média, sendo, entretanto, umas das mais lentas, com tempo médio de 21,58 ms para cada nuvem, enquanto que a rede mais rápida foi a LDGCNN atingindo 25$\mu$s de tempo médio com uma acurácia de 92,50%. Já no segundo experimento a rede Lidar3DNetV1 foi otimizada e comparada com a rede original, onde se observou significativa melhora de desempenho e mantendo métricas de avaliação semelhantes, atingindo 84,80% de acurácia em um dos subconjuntos de teste com 180 $\mu$s de tempo médio de processamento.



**Palavras-chave:** *LiDAR*. Aprendizagem profunda. Nuvem de Pontos. Pré-processamento. Validação cruzada.

## ABSTRACT

This work proposes the use of Deep Learning in recognition of 3D objects captured with *Light Detection and Ranging* (LiDAR), including a pre-processing step. In addition, it uses a real-world dataset that contains three classes. The pre-processing step is proposed to normalize the number of points of each captured cloud, bypassing the dimensionality discrepancies between the proposed dataset and the existing sets in the literature, such as ModelNet. Thus, cross-validation with three partitions is performed in the classification stage, aiming to measure the performance of the networks in different evaluation metrics from different arrangements of the set. In the first experiment with ten selected networks, the best performing network was RS-CNN, reaching 98.64% of average accuracy, being, however, one of the slowest, with an average time of 21.58 ms for each cloud, while the fastest network was LDGCNN reaching 25$\mu$s of average time with an accuracy of 92.50%. In the second experiment, the Lidar3DNetV1 network was optimized and compared with the original network, where a significant performance improvement was observed while maintaining similar evaluation metrics, reaching 84.80% accuracy in one of the tests subsets with 180 $\mu$ s average processing time.



**Keywords:** LiDAR. Deep Learning. Point Cloud. Preprocessing. Cross-validation.

## LISTA DE FIGURAS


Figura 1 – Ilustração da operação convolucional em quatro iterações. . . . . 20
Figura 2 – Ilustração da operação de *pooling*. . . . . 22
Figura 3 – Dropout . . . . . 23
Figura 4 – Gráfico da funcão de ativação *sigmoid*. . . . . 24
Figura 5 – Gráfico da funcão de ativação *tanh*. . . . . 24
Figura 6 – Gráfico da funcão de ativação *relu*. . . . . 25
Figura 7 – Gráfico da funcão de ativação *leaky relu*. . . . . 26
Figura 8 – Representação gráfica da Metodologia implementada e utilizada para realização deste trabalho. . . . . 31
Figura 9 – Amostras de nuvens de pontos originais (à esquerda) e amostras de nuvens de pontos pré-processadas. . . . . 35
Figura 10 – Validação cruzada para três partições (K=3) com três classes. . . . . 36
Figura 11 – Número de amostras por classes em cada subconjunto de treino. . . . . 36
Figura 12 – Número de amostras por classes em cada subconjunto de teste. . . . . 36
Figura 13 – Arquitetura original do modelo Lidar3DNetV1. . . . . 37
Figura 14 – Arquitetura do modelo Lidar3DNetV1 após otimização dos hiper-parâmetros. 39
Figura 15 – Matriz de confusão para a classificação de objetos. . . . . 39
Figura 16 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento. 46
Figura 17 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em microssegundos, para a redes original e otimizada. 51

## LISTA DE TABELAS

Tabela 1 – Métricas para cada Rede em cada um dos conjuntos de teste de nuvens de pontos capturadas. . . . . 44
Tabela 2 – Métricas para cada Rede sobre o conjunto de dados ModelNet40. . . . . 45
Tabela 3 – Resultados do teste *one-way* ANOVA para experimentos sobre os dados originais. . . . . 48
Tabela 4 – Resultados do teste Tukey HSD para experimentos sobre os dados originais. 49
Tabela 5 – Métricas para a Lidar3DNetV1 e a Lidar3DNetV1 com hiper-parâmetros otimizados. . . . . 50
Tabela 6 – Resultados do teste *one-way* ANOVA para experimento com otimização de hiper-parâmetros. . . . . 51
Tabela 7 – Resultados do teste Tukey HSD para experimento com otimização de hiper-parâmetros. . . . . 52

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| RS-CNN | *Relation-Shape Convolutional Neural Network* |
| LDGCNN | *Linked Dynamic Graph Convolutional Neural Network* |
| PAConv | *Position Adaptive Convolution* |
| GBNet | *Geometric Back-projection Network* |
| 3DMedPT | *3D Medical Point Transformer* |
| PVT | *Point-Voxel Transformer* |
| ANOVA | *Analysis of Variance* |
| HSD | *Honestly Significant Difference* |
| MLP | *Multilayer Perceptron* |
| CNN | *Convolutional Neural Networks* |
| VP | Verdadeiro Positivo |
| VN | Verdadeiro Negativo |
| FP | Falso Positivo |
| FN | Falso Negativo |

**SUMÁRIO**


**1 INTRODUÇÃO** . . . . . 12
**1.1 Estado da arte** . . . . . 14
**1.2 Objetivos** . . . . . 17
***1.2.1 Objetivos específicos*** . . . . . 17
**1.3 Organização do trabalho** . . . . . 17
**2 FUNDAMENTAÇÃO TEÓRICA** . . . . . 18
**2.1 *Multilayer Perceptron*** . . . . . 18
**2.2 *Convolutional Neural Networks*** . . . . . 19
***2.2.1 Convolução*** . . . . . 19
***2.2.2 Batch Normalization*** . . . . . 21
***2.2.3 Pooling*** . . . . . 21
***2.2.4 Camada densa*** . . . . . 22
***2.2.5 Dropout*** . . . . . 22
***2.2.6 Funções de ativação*** . . . . . 23
*2.2.6.1 Sigmoide* . . . . . *23*
*2.2.6.2 Tanh* . . . . . *23*
*2.2.6.3 Relu* . . . . . *25*
*2.2.6.4 Leaky Relu* . . . . . *25*
*2.2.6.5 Softmax* . . . . . *26*
**3 TRABALHOS RELACIONADOS** . . . . . 27
**3.1 PointNet** . . . . . 27
**3.2 *Relation-Shape Convolutional Neural Network* (RS-CNN)** . . . . . 27
**3.3 *Linked Dynamic Graph Convolutional Neural Network* (LDGCNN)** . . . . 27
**3.4 CurveNet** . . . . . 28
**3.5 SimpleView** . . . . . 28
**3.6 *Position Adaptive Convolution* (PAConv)** . . . . . 28
**3.7 *Geometric Back-projection Network* (GBNet)** . . . . . 29
**3.8 *3D Medical Point Transformer* (3DMedPT)** . . . . . 29
**3.9 Lidar3DNetV1** . . . . . 29
**3.10 *Point-Voxel Transformer* (PVT)** . . . . . 30

**4 METODOLOGIA** . . . . . 31
**4.1 Aquisição de dados** . . . . . 32
**4.2 Pré-processamento** . . . . . 33
**4.3 Classificação** . . . . . 34
**4.4 Otimização de hiper-parâmetros da Lidar3DNetV1** . . . . . 37
**4.5 Métricas de avaliação** . . . . . 38
**4.6 Testes de hipóteses** . . . . . 41
***4.6.1 Analysis of Variance (ANOVA)*** . . . . . 41
***4.6.2 Tukey's Honestly Significant Difference (HSD)*** . . . . . 41
**5 RESULTADOS E DISCUSSÕES** . . . . . 43
**5.1 Modelos originais** . . . . . 43
***5.1.1 Testes Estatísticos*** . . . . . 47
**5.2 Otimização de hiper-parâmetros da Lidar3DNetV1** . . . . . 48
***5.2.1 Testes Estatísticos*** . . . . . 51
**6 CONCLUSÕES E TRABALHOS FUTUROS** . . . . . 53
**REFERÊNCIAS** . . . . . 54

# 1 INTRODUÇÃO

As nuvens de pontos são estruturas de dados que descrevem objetos, cenas ou ambientes externos ou internos em três dimensões através de pontos distribuídos nas posições $(x, y, z)$ (RUSU; COUSINS, 2011). O sensor *Light Detection and Ranging* (*LiDAR*) é o mais utilizado para a captura de nuvens de pontos. Outro sensor que também tem essa finalidade é o *Kinect*, entretanto este é limitado em diversos aspectos, tais como curto alcance e problemas com a iluminação não controlada, em ambiente externo, por exemplo (ASSAD-UZ-ZAMAN *et al.*, 2021; ROSELL-POLO *et al.*, 2017). Desta forma o *Kinect* é mais indicado para a reconstrução de ambientes em 3D (FERREIRA; SILVA, 2021), avaliação motora de idosos (ONO *et al.*, 2020), descrever estruturas morfológicas de plantas em três dimensões (SUN; WANG, 2019), reconhecimento de linga de sinais (YANG, 2014), captura de movimento de pessoas (MüLLER *et al.*, 2017) e análise biomecânica de movimentos dos membros do corpo humano(BIJALWAN *et al.*, 2021). No entanto a utilização de nuvem de pontos está dentro de várias áreas do conhecimento, na Indústria, na Saúde, na Segurança e na Urbanização, por exemplo. Diversas são as aplicações nestas áreas, tais como veículos autônomos (HECHT, 2018), robótica (MALAVAZI *et al.*, 2018), realidade aumentada (BONNAFFE *et al.*, 2007) e planejamento urbano (ZHANG, 2010). E, para tais aplicações se faz necessário a utilização de sensores de um porte maior, como o *LiDAR*, que é utilizado também em aplicações florestais (GIONGO *et al.*, 2010), sendo uma forma de capturar a topologia do terreno e estimar a altura das árvores, simultaneamente, por meio de capturas realizadas utilizando o *LiDAR* sobre florestas. Dados capturados com o *LiDAR* também são utilizados por modelos de elevação digital (LIU, 2008), os quais podem ser utilizados para identificar deslizamentos de terra, como no trabalho proposto por Jaboyedoff *et al.* (2012) na área de Geociência.

Em comparação com outros tipos de sensores e formas de capturar dados para fins de reconhecimento de objetos 3D, o *LiDAR* apresenta uma série de vantagens que o coloca como a opção mais indicada a se utilizar, dentre elas estão a não necessidade de iluminação no ambiente em que a coleta será realizada (GIONGO *et al.*, 2010), necessidade habitual de aplicações que utilizam câmeras radar, stereo ou realizem o processo de fotogrametria para mensurar as profundidades da cena capturada. Além disso, o *LiDAR* é um sensor de baixo consumo de energia e de alta frequência de varredura, superando a grande maioria das câmeras convencionais. Desta forma, o *LiDAR* passa a ser o sensor mais confiável e utilizado em aplicações que envolvam abordagem em três dimensões, principalmente quando se trata de nuvens de pontos.

Diversos conjuntos de dados foram propostos para aplicações em 3D, dentre os quais destacam-se o Paris-Lille-3D (ROYNARD *et al.*, 2018), o qual apresenta 50 classes de objetos capturados em duas cidades da França, Paris e Lille, e é utilizado para a tarefa de Classificação de objetos. Assim como o KITTI (GEIGER *et al.*, 2013), conjunto de dados e *benchmark* para a detecção de objetos 3D, tendo suas amostras capturadas a partir de uma plataforma que emula um carro autônomo. Outro conjunto de dados de destaque é o ModelNet (WU *et al.*, 2015), com mais de 600 categorias e subdividido em dois conjuntos, ModelNet10 e ModelNet40, este é um dos maiores e mais difundido conjuntos para a Classificação de objetos em 3D. O ShapeNet (CHANG *et al.*, 2015) conta com 55 categorias espalhadas por mais de 51000 objetos, todos anotados manualmente. Enquanto que o Semantic3D (HACKEL *et al.*, 2017) é um conjunto capturado com um *LiDAR* e conta com mais de 2 milhões de pontos rotulados de um todos de 4 bilhões de pontos, sendo um conjunto usual tanto para a Classificação, quanto para a Segmentação de objetos 3D.

Estes conjuntos de dados são os mais conhecidos e difundidos para a realização de diversas tarefas de Aprendizado de Máquina e Aprendizagem Profunda. Alguns trabalhos propuseram métodos para solucionar diferentes problemas a partir destas bases de dados citadas, tais como Detecção de objetos 3D (SUN *et al.*, 2021; LI *et al.*, 2021; FAN *et al.*, 2021; HAHNER *et al.*, 2021; HAHNER *et al.*, 2022; ZHANG *et al.*, 2022; HU *et al.*, 2022; WEI *et al.*, 2022) que consiste na localização e classificação de objetos presentes numa nuvem de pontos. Assim como na Segmentação de objetos 3D (THOMAS *et al.*, 2019; HU *et al.*, 2020a; ZHOU *et al.*, 2020; BOULCH, 2020; MILIOTO *et al.*, 2019; UNAL *et al.*, 2022), tarefa na qual o objeto encontrado na nuvem de pontos tem seus pontos classificados um a um, como sendo pertencentes ou não àquele objeto. Outros autores propuseram uma Fusão de dados (BAI *et al.*, 2022; LI *et al.*, 2022; KOH *et al.*, 2021; WANG *et al.*, 2022), métodos em que a rede recebe simultaneamente nuvem de pontos e imagem, gerando uma fusão interna das características extraídas de ambos os dados, garantindo um vetor de atributos com mais informações na saída da rede, antes da etapa de classificação. Além destes, também existem alguns estudos para a tarefa de Segmentação Panóptica (HONG *et al.*, 2021; ZHAO *et al.*, 2021; ZHOU *et al.*, 2021; XU *et al.*, 2022; FAZLALI *et al.*, 2022), abordagem na qual a rede recebe uma nuvem de pontos omnidirecional e segmenta os objetos presentes em toda abertura de captação.

Todas as tarefas citadas acima necessitam da execução da tarefa de Classificação de objetos na etapa final de suas abordagens, seja para classificar um objeto detectado, determinar a

classe do ponto de um objeto segmentado, ou a partir da fusão de uma imagem e uma nuvem de pontos, predizer a classe do objeto por ambas informações. Em vista disso, a tarefa de classificação é de extrema importância para todas as aplicações que envolvam Aprendizado de Máquina e processamento de nuvens de pontos.

Desta forma foi desenvolvida uma abordagem que utiliza que busca trazer as seguinte contribuições:

- utilização de dez métodos de Classificação de objetos 3D: PointNet, RS-CNN, LDGCNN, CurveNet, SimpleView, PAConv, GBNet, 3DMedPT, Lidar3DNetV1, PVT;
- um modelo otimizado obtido a partir da Lidar3DNetV1, chamado Lidar3DNetV1Opt;
- a construção de um conjunto de dados chamado *LAPISCO-LIDAR Real-world dataset*, proposto para a tarefa de Classificação de objetos 3D, contendo 3 classes e mais de 500 objetos reais.

Assim, para evidenciar a eficiência da abordagem, foi realizada uma avaliação de desempenho entre as dez redes citadas, assim como entre o modelo Lidar3DNetV1 e um modelo otimizado desta mesma arquitetura.

## 1.1 Estado da arte

O método proposto por Kumawat e Raman (2019) visa extrair a fase de uma vizinhança 3D com o bloco ReLPV, através da *Short Term Fourier Transform* (STFT) focada nos pontos de baixa frequência. Enquanto que a SO-Net, proposta por Li *et al.* (2018) utiliza *Self-Organizing Map* (SOM) para realizar uma extração hierárquica de características, gerando um vetor de atributos de saída através de campos receptivos ajustáveis pelas distância dos pontos para os nós SOM.

Landrieu e Simonovsky (2018) propôs o *Superpoint Graph* (SPG), técnica que consiste na representação de relações contextuais entre elementos geometricamente homogêneos, que é utilizada por uma rede convolucional por grafos na etapa de classificação. Já Ma *et al.* (2019) desenvolveu um método que combina um *Convolutional Neural Networks* (CNNs), com conexões residuais para extração de características de baixo nível, e uma *Long Short-Term Memory* (LSTM), a qual é utilizada com uma camada de votação de sequência para agregar as características em um vetor descritor utilizado na camada de classificação.

Maturana e Scherer (2015) propôs o modelo de CNN 3D chamado VoxNet, que se baseia na representação volumétrica de grade de ocupação dos pontos presentes na cena

capturada. Ao mesmo tempo que Balado *et al.* (2020) propôs um método baseado na conversão de nuvens de pontos em imagens, possibilitando a utilização de CNN 2D, sobre múltiplas visualizações geradas a partir de uma única nuvem de pontos.

Zhang *et al.* (2021) desenvolveu um método auto supervisionado para qualquer tipo de dados de representação tridimensional utilizando arquiteturas de modelo Xoxel pré-treinadas em nuvens de pontos padrões. Enquanto Hu *et al.* (2020b) propôs uma rede convolucional de dois fluxos, um para extrair informações de regiões e outro para extrair características de bordas.

A Point-BERT (YU *et al.*, 2021b) utiliza *Masked Point Modeling* (MPM) para pré-treinar os transformadores de nuvens de pontos, através do uso de *discrete Variational AutoEncoder* (dVAE) sobre partições de pontos em cada nuvem, o qual é responsável pela coleta de informação local de cada partição. Por sua vez, a *Multi-View Transformation Network* (MVTN) (HAMDI *et al.*, 2021) é uma rede proposta para aprender as características presentes na nuvem de pontos a partir de diferentes pontos de vista dentro da mesma.

Zhang *et al.* (2021a) propuseram um método de acréscimo de dados de nuvem de pontos chamado PointCutMix, o qual através da atribuição ideal de pontos entre pares de pontos de duas nuvens distintas, é capaz de gerar uma nova amostra de nuvem de pontos, aumentando o montante de amostras presentes no conjunto de dados original. De forma semelhante, Lee *et al.* (2021) propuseram o *Rigid Subset Mix* (RSMix), um método de acréscimo de dados que se baseia na combinação de regiões preservadas em diferentes nuvens de pontos para gerar uma amostra de nuvem de pontos mista. Esta combinação ocorre através da extração de subconjuntos de pontos, mantendo as informações contidas em cada um.

A *Geometry-Disentangled Attention Network* (GDANet) (XU *et al.*, 2021b) traz um módulo de desembaraçamento de pontos de uma nuvem para formar contornos e partes planas de objetos 3D a partir da denotação de componentes acentuadas e suaves, respectivamente. Este módulo é chamado de *Geometry-Disentangle Module*. Enquanto que Zhang *et al.* (2021b) desenvolveram a DSPoint, método que utiliza escala dupla com fusão de alta frequência para a extração de características locais e globais processando ao mesmo tempo voxels e pontos. As duas escalas se dividem em uma escala de granulação fina, onde ocorre a convolução pontual, e a outra de longo alcance, na qual ocorre a atenção global por voxel de modo a explorar estruturas maiores dentro da nuvem de pontos.

As *Point Convolutional Neural Networks* (PCNN) (ATZMON *et al.*, 2018) são redes neurais convolucionais para nuvens de pontos, que trabalham com dois tipos de operação, a

extensão e a restrição, ambas auxiliam na execução do recuo da convolução volumétrica Euclidiana, sendo, desta forma, invariante à ordenação dos pontos, densidades de pontos e translação da nuvem de pontos. Xu *et al.* (2018) propuseram a SpiderCNN para extrair características geométricas das nuvens de pontos, através das camadas convolucionais chamada SpiderConv, as quais realização convoluções de grades regulares mesmo em conjuntos não regulares de pontos, além disso, utilizam um filtro para captura de características geodésicas locais, implementado a partir do produto entre uma função degrau e um polinômio de Taylor.

Yavartanoo *et al.* (2021) propuseram a PolyNet, uma rede convolucional de duas operações, a primeira é a PolyConv, uma operação convolucional polinomial responsável por aprender distribuições contínuas como filtros convolucionais. A segunda operação é a PolyPool, que utiliza a representação de multi-resolução, chamada PolyShape, para reduzir as dimensões dos vetores de características extraídos. Enquanto que a PointConv (WU *et al.*, 2019) é uma CNN 3D que trata os filtros convolucionais como funções das coordenadas dos pontos compostas por componentes de densidade e peso. A primeira componente é obtida por meio de estimativa de densidade do filtro naquele ponto, já a segunda é calculada por meio de um perceptron de multicamadas. O filtro de convolução é utilizado para obter a convolução invariante de tradução e de permutação.

Desta forma, os métodos citados acima podem ser divididos em três abordagens principais: métodos que realizam o acréscimo de dados de nuvens de pontos, conhecido como *data augmentation* para ganhar maior conhecimento durante treinamento, métodos que utilizam CNN, que apesar de proporem diferentes operações, o funcionamento base continua sendo a extração de atributos por meio de filtros convolucionais aliada à uma etapa de classificação na camada de saída, e métodos que geram diferentes visualizações da nuvem de pontos para classificá-la, através da multi-visualização da nuvem de entrada.

Assim, o primeiro tipo adiciona uma etapa a mais antes ou durante o treinamento, além de extrair conhecimento de amostras sintéticas que não fazem parte do conjunto de dados original e consequentemente não têm rotulação. Enquanto que rede que geram diferentes visualizações da nuvem de pontos, passam a perder informação de profundidade ao trabalhar com duas dimensões, ao mesmo tempo que se torna mais lenta por ter mais informação para processar ao mesmo tempo.

## 1.2 Objetivos

O objetivo principal deste trabalho é o reconhecimento de objetos em 3D a partir de nuvens de pontos capturadas por *LiDAR* utilizando CNNs.

### *1.2.1 Objetivos específicos*

De maneira mais específica, este trabalho tem os seguintes objetivos:

- Analisar diversas arquiteturas de CNN presentes no estado da arte no reconhecimento de objetos em 3D;
- Levantar um conjunto de dados de nuvens de pontos obtidas com *LiDAR*;
- Propôr a aplicação de um método de pré-processamento para normalizar as nuvens de pontos;
- Otimizar os hiper-parâmetros de um modelo já existente;
- Examinar o desempenho de cada método utilizando métricas de avaliação, tais como Acurácia, Precisão, Sensibilidade, F1-*Score* e Especificidade, além do tempo de processamento;
- Comparar os resultados alcançados usando o modelo otimizado com os demais métodos do estado da arte.

## 1.3 Organização do trabalho

Esta dissertação está organizada em seis capítulos. Além do Capítulo 1, o qual introduz este trabalho, no Capítulo 2 está disposta a fundamentação teórica com a descrição das técnicas utilizadas. Em seguida, no Capítulo 3 todos os métodos utilizados são descritos. Além disso, no Capítulo 4 a metodologia é apresentada com todos os processos realizados sendo destacados. O Capítulo 5 apresenta uma análise dos resultados obtidos com os experimentos. Por fim, o Capítulo 6 trás as conclusões alcançadas e propostas para futuros trabalhos.

# 2 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção serão abordadas as principais técnicas utilizadas pelos métodos de Classificação de objetos 3D presentes no estado da arte. Passando pela estrutura e funcionamento de redes neurais mais simples, como *Multilayer Perceptron* (*MLP*), e chegando até redes mais complexas, como as CNN

## 2.1 *Multilayer Perceptron*

O *Multilayer Perceptron* (MLP) é um algoritmo de aprendizado de máquina supervisionado difundido entre aplicações de classificação e de regressão, este método baseia-se na utilização de múltiplos modelo do Perceptron simples, formando uma rede neural artificial para a solução de problemas não lineares (HAYKIN, 2001).

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 $\phi$ é a função de ativação, $x_i$ é o vetor de entrada e $w_i$ é o vetor de pesos quem 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 $\eta$ é taxa de aprendizagem, responsável por ponderar o erro de predição. Sabendo que o Perceptron não era capaz 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 $l0$ o número de neurônios é $D$. Ainda, $\phi$ 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 se 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, $\phi'$ 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 *Convolutional Neural Networks*

Nesta Seção serão apresentados os principais métodos, operações e funções de ativação que são utilizados na implementação, treinamento e validação de CNN, de modo a auxiliar no entendimento sobre o funcionamento de redes deste tipo.

### *2.2.1 Convolução*

A convolução representa a aplicação de um filtro para extrair informações relevantes da imagem de entrada, ou de camadas anteriores (ALBAWI *et al.*, 2017). Diferentes filtros são

aplicados, dependendo da estrutura da CNN, com a finalidade de extrair da imagem a maior quantidade de características, tais como bordas, contraste, textura e gradiente.

A Equação 2.7 representa a convolução do filtro K em uma imagem I. de modo que o filtro se desloca por toda imagem, e onde houver sobreposição entre ambos, haverá um somatório dos produtos de elemento por elemento (GOODFELLOW *et al.*, 2016).

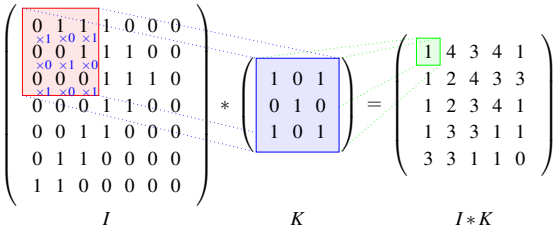
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.7}$$

em que $S$ é o mapa de características originado pela convolução do filtro $K$ na imagem $I$. Na Figura 1 estão dispostas quatro iterações da operação de convolução, neste caso de um filtro 3x3 em uma matriz 7x7, sem *padding* e com *stride*, passo com o qual o filtro se desloca, de um. A ordem da matriz de saída, G, é dada pela Equação 2.8,

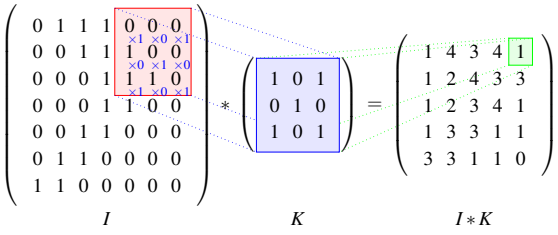
$$G = \frac{N-F}{S} + 1 \tag{2.8}$$

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 1a 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 1b apresenta a iteração que resulta no elemento $(1,5)$ da matriz de saída $I * K$. Enquanto as Figuras 1c e 1d ilustram a mesma operação para os elementos $(5,1)$ e $(5,5)$ de $I * K$, respectivamente.

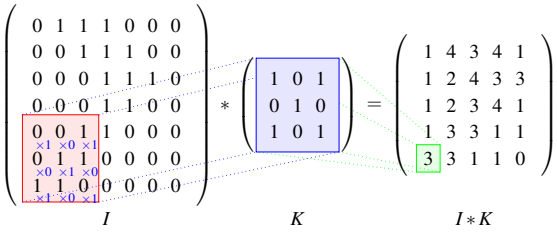
Figura 1 – Ilustração da operação convolucional em quatro iterações.



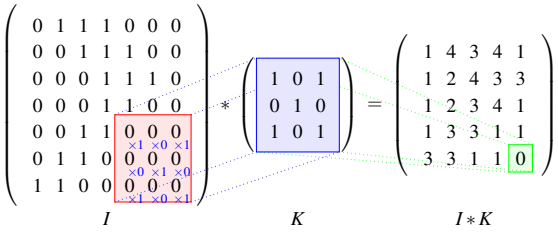

(a) $(I * K)_{11}$.




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




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




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

Fonte: Elaborada pelo autor.

A convolução, operação que foi descrita acima, é a operação base para o funcionamento de um Rede Neural Convolucional (CNN), uma vez que a partir dos dados de entrada, filtros aplicados seguidamente, extraem característica que descrevem em uma dimensionalidade reduzida, os dados recebidos na entrada da CNN. Nas seções seguintes serão descritos alguns métodos utilizados para realizar e otimizar este processo de extração e classificação de atributos.

### *2.2.2 Batch Normalization*

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.

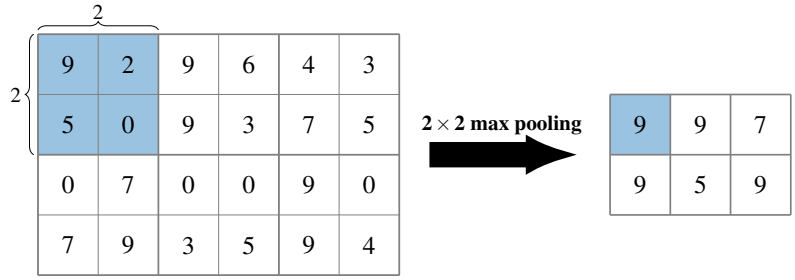
### *2.2.3 Pooling*

Uma estratégia para reduzir a dimensionalidade dos dados de entrada de uma CNN é utilizar na saída da camada convolucional a operação de *pooling*. 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). Assim como um filtro na convolução, esses campos receptivos também percorrem os dados de entrada, mas com *stride* igual ao tamanho do campo receptivo.

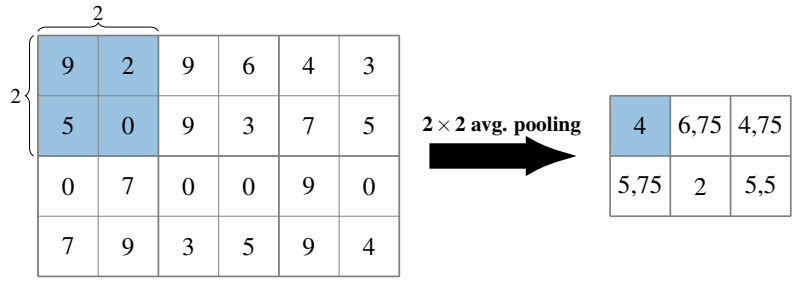
Existem dois tipos mais usados de *pooling*, ambos expostos na Figura 2, na qual os filtros receptivos têm tamanho 2x2 e percorrem uma matriz 4x6. Assim, ao final de ambas operações, a matriz de saída terá dimensão 2x3, corroborando com a redução de dimensionalidade que é objetivo da operação.

O primeiro é o *max pooling* visto na Figura 2a, é a operação em que o maior valor do campo receptivo é escolhido e atribuído à matriz de saída. Já o segundo *pooling* é

Figura 2 – Ilustração da operação de *pooling*.




(a) Ilustração do *max pooling*.




(b) Ilustração do *average pooling*.

Fonte: Elaborada pelo autor.

o *average pooling* que calcula a média dos valores campo receptivo e concede à matriz de saída (GOODFELLOW *et al.*, 2016), como observado na Figura 2b.

### *2.2.4 Camada densa*

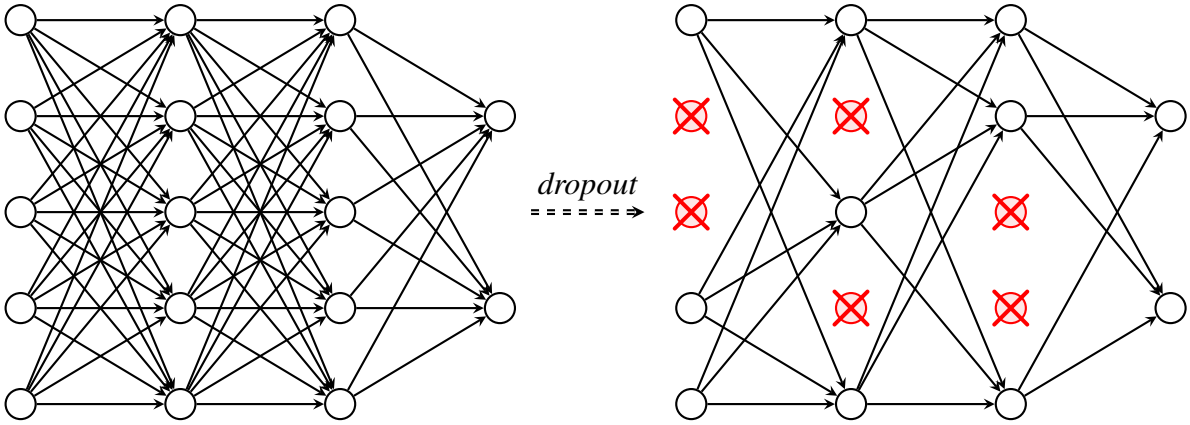
A Camada densa ou totalmente conectada é um conjunto de neurônios que operam propagando adiante as informações que pondera na sua entrada. Quanto posicionada na saída da rede, ela será acamada responsável pela classificação do vetor de características ao final das operações de convolução e *pooling* 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.

### *2.2.5 Dropout*

O *Dropout* foi proposto por Srivastava *et al.* (2014) e é uma técnica de regularização de uma rede neura. Na Figura 3 é apresentada uma ilustração da aplicação deste método, utilizado para evitar o *overfitting*, nome que é dado ao sobreajuste do aprendizado de uma rede neura no

subconjunto de treinamento.

Figura 3 – Dropout




Fonte: Elaborada pelo autor.

Este método desativa uma taxa fixa de neurônios em cada camada em é aplicado. No exemplo acima, 40% dos neurônios foram desativados por cada, deste modo os neurônios desativados (vermelho), não contribuem na alimentação direta, assim como não têm pesos atualizados pelo algoritmo de *backpropagation*.

### *2.2.6 Funções de ativação*

#### *2.2.6.1 Sigmoide*

A sigmoide (HAYKIN, 2001) é uma função de ativação difundida na implementação de redes neurais, sendo continuamente diferenciável e tendo um intervalo de saída entre 0 e 1. Esta função é descrita pela Equação 2.10.

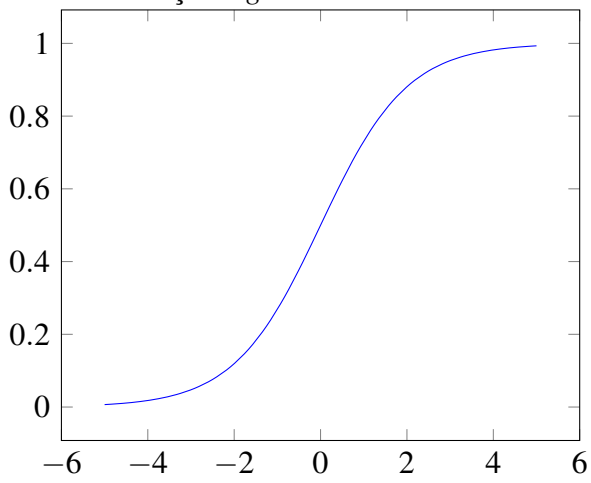
$$f(x) = \frac{1}{1+e^{-x}} \tag{2.10}$$

Na Figura 4 está exposto o gráfico correspondente a essa função de ativação, nota-se que é uma função suavizada, na qual pequenas variações na entrada não provocam grandes variações na saída. A sigmoide não é simétrica em relação ao eixo horizontal e por isso todos os valores que são passados a camada posterior são positivos.

#### *2.2.6.2 Tanh*

Outra função de ativação bastante conhecida e utilizada é a Tangente Hiperbólica ou *Tanh*, essa função apresenta um valor mínimo de -1 e um valor máximo de 1, sendo simétrica em

Figura 4 – Gráfico da funcão de ativação *sigmoid*.



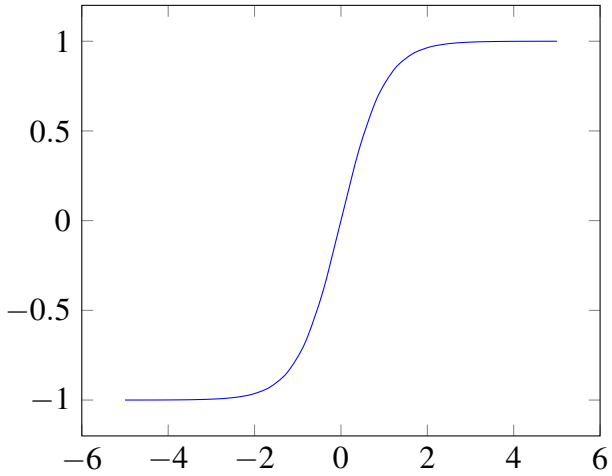

Fonte: Elaborada pelo autor.

relação ao eixo horizontal (HAYKIN, 2001), é dada pela Equação 2.11.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.11}$$

Na Figura 5 percebe-se que próximo à origem sua derivada é maior que a derivada da sigmoide na mesma região, o que reflete em uma variação maior na saída para mudanças no valor de entrada em intervalos próximos à origem. Além disso, ao invés da sigmoide, a tanh faz com que os valores, que sejam repassados adiante para os neurônios da próxima camada, não sejam apenas positivos.

Figura 5 – Gráfico da funcão de ativação *tanh*.




Fonte: Elaborada pelo autor.

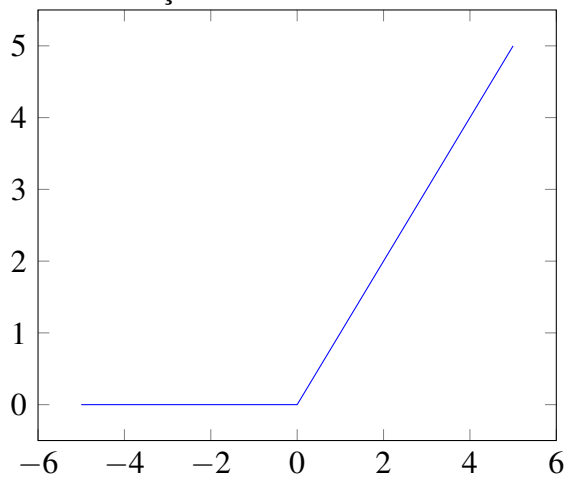
#### 2.2.6.3 *Relu*

A *relu*, ou unidade linear retificada é das uma funções de ativação mais utilizadas atualmente, esta é uma função não linear (AGARAP, 2018) dada pela Equação 2.12:

$$f(x) = max(0,x) \tag{2.12}$$

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. Isto faz com que o custo computacional durante o treino seja menos, já que menos operações serão realizadas ao mesmo tempo.

Figura 6 – Gráfico da funcão de ativação *relu*.




Fonte: Elaborada pelo autor.

#### 2.2.6.4 *Leaky Relu*

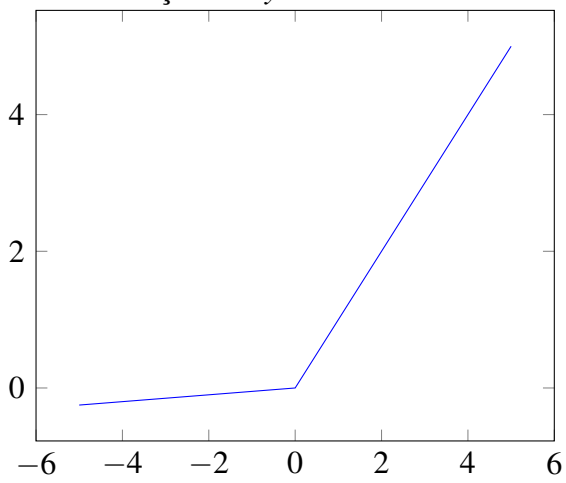
Quando o gradiente se aproximar de zero, devido à não linearidade da relu pode ocorrer o problema de explosão de gradiente, e para evitar este problema a *leaky relu* foi proposta (XU *et al.*, 2015). Esta função é uma melhoria da relu, uma vez que para valores menores que 0, a saída não será mais nula, com a *leaky relu* a saída passa a ser um valor menor e proporcional à entrada, como expresso na Equação 2.13:

$$f(x) = \begin{cases} x & se\ x \geq 0 \\ kx & se\ x < 0 \end{cases} \tag{2.13}$$

A partir da Figura 7 é visto a região em que a saída deixa de ser zero, como no caso da relu, e passa a ter um valor proporcional à entrada da função de ativação. Assim, os neurônios

estarão sempre ativados, mas quando sua saída for negativa, a informação será propagada com uma proporção menor que a informação positiva.

Figura 7 – Gráfico da funcão de ativação *leaky relu*.




Fonte: Elaborada pelo autor.

#### *2.2.6.5 Softmax*

Geralmente utilizada nas camadas de classificação das redes neurais, a *softmax* (BRIDLE, 1989) é responsável por transformar o espaço do vetor de saída da rede neural para um espaço probabilístico, no qual os valores estão dispostos entre 0 e 1, e a soma de todos os valores resulta em 1. A *softmax* é descrita pela Equação 2.14:

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{2.14}$$

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.

# 3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados trabalhos no estado da arte que também realizam a tarefa de Classificação de Objetos 3D em nuvens de pontos, e que serão avaliados nesta tarefa sobre o conjunto de dados proposto neste trabalho.

## 3.1 PointNet

A PointNet (QI *et al.*, 2017) é uma rede proposta para classificação de objetos, segmentação de regiões determinadas na nuvem de pontos e análise semântica da cena. A nuvem de entrada de sua rede é bruta, não é aplicado nenhum método de agrupamento, vocalização ou renderização. 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 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 não citou métricas de tempo de processamento.

## 3.2 RS-CNN

A rede RS-CNN (LIU *et al.*, 2019) utiliza a base da rede neural convolucional para a análise de dados regulares, como imagem, por exemplo, e adapta para processamento de nuvens de pontos com técnica de clusterização. Uma relação entre pontos locais é realizada considerando as topologias geométricas. O peso convencional de cada região aprende essa relação baseado em prioras 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.

## 3.3 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).

## 3.4 CurveNet

A CurveNet apresenta uma técnica com o objetivo de melhorar a metodologia de agrupamentos globais e locais, apresentada na rede LDGCNN(ZHANG *et al.*, 2021). Para tal foca na melhoria da geometria da nuvem de pontos por meio da geração de sequências contínuas de segmentos de ponto. A técnica proposta considera a relação entre os pontos da nuvem de pontos como um grafo não direcionado, onde os pontos discretos servem como nós do grafo e as conexões de pontos vizinhos como arestas do grafo, uma curva pode, portanto, ser descrita como um passeio no grafo. O autor destaca mais essa relação em agrupamentos de longo alcance. Para execução do método é primeiro detalhado regras para agrupar curvas de uma nuvem de pontos e em seguida são avaliadas as semelhanças das curvas agrupadas (XIANG *et al.*, 2021).

## 3.5 SimpleView

Ao avaliar que fatores como diferentes esquemas de avaliação, estratégias de aumento de dados e funções de perda interferem no desempenho das abordagens baseadas em pontos, independente da arquitetura do modelo. O método proposto por Goyal *et al.* (2021), a SimpleView é controlar essas limitações e melhorar a classificação de nuvens de pontos. A metodologia foi aplicada na rede PointNet(QI *et al.*, 2017), que possui um número variado de classes/objetos, e obteve resultados melhores ao comparar com o artigo original, porém o estudo não avaliou em outras redes da literatura.

## 3.6 PAConv

A rede PAConv propõe método de convolução adaptável onde o *kernel* de convolução gera matrizes de pesos básicos armazenadas no banco de peso. Os coeficientes dos pesos são treinados a partir de posições pontuais por meio de MLPs, também chamado de *Scorenet*. A técnica de *kernel* contorna a demanda de memória e carga computacional exigida para modelar

variações espaciais e estruturais geométricas complexas das nuvens de pontos. Sua metodologia obteve bons resultados quando incorporados especificamente em redes baseadas em MLP e sem modificações nas configurações de rede, sendo assim necessário uma etapa de ajuste dos dados de entrada da rede (XU *et al.*, 2021a).

## 3.7 GBNet

Aplicada no contexto de classificação de nuvens de pontos, especificamente na captura de características geométricas das nuvens de pontos, a metodologia da rede GBNet inicialmente adiciona mais informações geométricas sobre os pontos no espaço 3D global e utiliza estruturas baseadas em CNN para aprender o contexto geométrico local. Possíveis problemas com duplicidade de informações e dados redundantes são tratados por um módulo de afinidade quando nuvens de pontos pequenas são processadas. Ao adicionar mais informações a rede e módulos de afinidade recursos de processamento e tempo são elevados (QIU *et al.*, 2021)

## 3.8 3DMedPT

A rede 3D medical point Transformer (3DMedPT) utiliza a metodologia de análise de nuvens de pontos em imagens médicas para auxiliar na detecção e tratamento de doenças. Com inserção de informações contextuais sobre a nuvem de ponto, semelhante a GBNet, o método propõe examinar as estruturas biológicas complexas, pois seu módulo de atenção ou afinidade captura informações no contexto local quanto às informações globais da nuvem de pontos. Uma limitação da nuvem de pontos médicas é a disponibilidade de amostras de nuvens de pontos. O autor utiliza técnicas de embeddings e Multi-Graph Reasoning para contornar essa limitação (YU *et al.*, 2021a).

## 3.9 Lidar3DNetV1

A rede Lidar3DNetV1 propõe um método de deep learning com a estrutura da PointNet para reconhecimento de objetos 3D em nuvens de pontos geradas pelo *Light Detection and Ranging* (LIDAR) e no simulador Webots. Além disso, uma etapa de pré-processamento nas nuvens de pontos foi realizada com o objetivo de avaliar a variação da quantidade de pontos das nuvens de pontos e o desempenho no reconhecimento do objeto 3D. Três abordagens foram propostas para avaliar o método de classificação de objetos e no melhor desempenho alcançou a

acurácia de 98,33% com tempo de teste de 88s (SOUSA *et al.*, 2021).

## 3.10 PVT

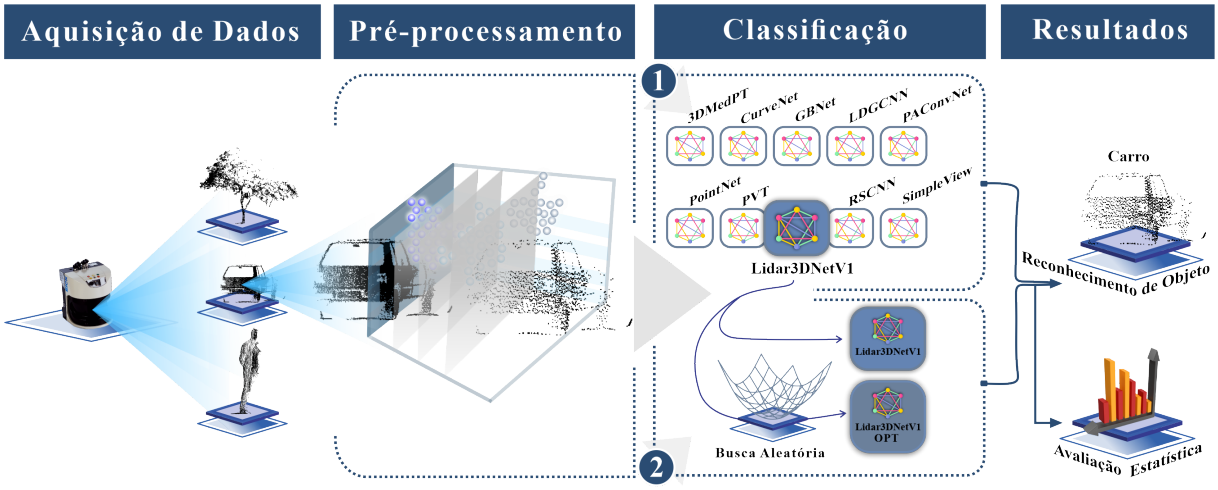
A rede Point-Voxel Transformer (PVT) é desenvolvida para contribuir na problemática de custo de processamento de processamento de nuvens de pontos. Para isso o autor propõe dois módulos o *Sparse Window Attention* (SWA) e módulo de atenção relativa (RA), sua combinação é a arquitetura neural PVT para 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).

# 4 METODOLOGIA

Nesta Seção será descrita toda Metodologia seguida para a elaboração deste trabalho. Primeiro será explicada a forma de aquisição e estruturação do conjunto de dados utilizado, em seguida o pré-processamento escolhido será apresentado, assim como será detalhada a etapa de classificação dos dados, e por fim, as métricas de avaliação utilizadas serão definidas.

Na Figura 8 está apresentada uma representação gráfica da Metodologia aplicada nesta dissertação, a qual consiste em quatro etapas: Aquisição de dados, Pré-processamento, Classificação e Resultados. Na etapa de Aquisição de dados três tipos de objetos reais foram capturados em diferentes cenas utilizando um *LiDAR*, ao final desta etapa, 514 nuvens de pontos foram geradas e armazenadas contendo estas três classes. Na segunda etapa ocorreu o Pré-processamento, operação pela qual todas as nuvens de pontos foram submetidas para que tivessem o número de pontos normalizados em 2048, dimensão padrão para todas as redes que serão experimentadas neste trabalho.

Figura 8 – Representação gráfica da Metodologia implementada e utilizada para realização deste trabalho.




Fonte: Elaborada pelo autor.

Já na etapa de Classificação, dois experimentos foram realizados, o primeiro utilizando dez métodos de Classificação de objetos 3D presentes no estado da arte e abordados no Capítulo 3, Trabalho Relacionados. Enquanto que o segundo experimento foi realizado, especificamente, sobre o modelo LiDAR3DNetV1, comparando uma versão de hiper-parâmetros otimizados com o modelo original. Em ambos experimentos as redes passaram por uma Validação Cruzada com a finalidade de gerar vetores de predições para cada uma das três partições em

que o conjunto de dados foi dividido.

Ao final, na etapa de Resultados, os vetores gerados na Classificação são usados para o cálculo de uma sequência de cinco métricas de avaliação para cada rede em ambos experimentos citados. Estas métricas passam por uma validação de significância estatística, a partir da qual uma rede pode ser escolhida como a melhor para a Classificação de objetos 3D.

## 4.1 Aquisição de dados

Para captar as nuvens de pontos, foi utilizado o sensor *LiDAR* LMS511, o qual está amplamente inserido em aplicações industriais. Este sensor é adequado para uso externo e empregado em medições de perímetro e volume, assim como em aplicações de veículos autônomos, tendo um ângulo de abertura de 190°, é capaz de captar objetos em um raio de até 80 metros dentro desta abertura, com diferentes frequências de varreduras, de 25 à 100 Hz. Além disso, a resolução angular pode ser ajustada de 0,1667 à 1,0°, e cada ponto da nuvem tem 11,9 mrad de diâmetro.

Originalmente o *LiDAR* LMS511 é um sensor 2D que retorna um vetor de distâncias dos pontos das superfícies em que seu o feixe reflete. Desta forma foi necessário utilizar um algoritmo integrado ao *software Robot Operating System* (ROS) para realizar a varredura em uma sequência de profundidades diferentes, de modo a montar um perfil tridimensional da cena na qual o sensor está inserido, gerando uma nuvem de pontos a partir da fusão de uma série de vetores de distância, originados pelas varreduras do *LiDAR*.

Para os objetos menores, carros e pessoas, o sensor foi posicionado em distâncias de 5 à 10 metros. Já para os objetos maiores, caso das árvores, o sensor foi deslocado de 10 a 20 metros, para que o sensor conseguisse captar toda altura de cada árvore. Esse processo de ajuste da posição do sensor de acordo com o objeto, garante que a densidade de pontos do objeto na nuvem seja alta, elevando a qualidade das informações contidas em cada varredura.

Uma vez tendo as varreduras realizadas, os objetos contidos em cada cena foram manualmente segmentados com o auxílio do aplicativo de código aberto *CloudCompare*, uma vez que visou-se evitar ao máximo a perca de informações dos objetos como ocorre em alguns métodos de segmentação de objetos 3D do estado da arte (THOMAS *et al.*, 2019; HU *et al.*, 2020a; UNAL *et al.*, 2022), mesmo que de forma reduzida, tais perdas acarretam na deformação dos objetos. Após a segmentação manual, as nuvens de pontos apresentam quantidades de pontos diferentes, logo existe a necessidade de normalizar os tamanhos de todas as nuvens de pontos

para um único tamanho. Como padrão escolheu-se 2048 pontos, quantidade padrão para todos os classificadores de objetos 3D utilizados, assim como para conjuntos de dados como o ModelNet. Além disso, para reduções de dimensionalidade maiores que esta, aumentam-se as chances de se perder informações importantes relativas a forma do objeto contido na nuvem de pontos.

Ao final, 514 objetos foram segmentados, sendo 258 árvores, 114 carros e 142 pessoas. Esses três tipos de objetos foram selecionados por serem os mais comuns em ambientes externos nos quais um veículo autônomo possa estar inserido, tais como ruas, estradas e rodovias. Desta forma, este conjunto de dados poderá auxiliar no desenvolvimento de modelos capazes de realizar a tarefa de mapeamento de objetos no entorno de um veículo autônomo. Este conjunto de dados leva o nome de *LAPISCO-LIDAR Real-world dataset* (SOUSA *et al.*, 2021). O conjunto de dados será disponibilizado através de solicitação mediante contato com os grupos de pesquisa GPAR-UFC e LAPISCO-IFCE.

## 4.2 Pré-processamento

Como uma das contribuições deste trabalho tem-se a adição de uma etapa de pré processamento da nuvem de pontos, com a finalidade de normalizar sua dimensão para 2048 pontos. Este pré processamento é realizado a partir da Interpolação Trilinear (BOURKE, 1999), que visa interpolar dados volumétrico através da combinação de oito pontos diferentes, o que implica na menor perca de informação no conjunto de pontos resultante. Supondo que nuvem de pontos tenha, inicialmente M pontos, e que é desejado que após esta etapa, a nuvem esteja normalizada em N pontos, neste caso N será 2048. Através da Equação 4.1, obtém-se as razões de proporção entre as dimensões de cada eixo cartesiano,

$$s_X = s_Y = s_Z = \frac{M}{N} \tag{4.1}$$

em 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. Logo, será possível calcular a nova posição de um ponto sobre cada uma destes eixos, através da Equação 4.2,

$$x_f = y_f = z_f = x's_X = y's_Y = z's_Z, \ \forall \ x = y = z = 1, ..., N \tag{4.2}$$

desta forma tem-se 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 $\Delta_z = z_f - z$.

A Equação 4.3 calcula, portanto, os pontos da nuvem pré-processada, por meio de uma combinação de oito pontos que forma um cudo, $(x,y,z)$, $(x+1,y,z)$, $(x,y+1,z)$, $(x,y,z+1)$,

$(x+1,y,z+1)$, $(x,y+1,z+1)$, $(x+1,y+1,z)$ e $(x+1,y+1,z+1)$ com suas, respectivas variações em relação ao ponto de origem.

$$\begin{aligned} K(x',y',z') = {} & L(x,y,z)(1-\Delta_x)(1-\Delta_y)(1-\Delta_z) + L(x+1,y,z)\Delta_x(1-\Delta_y)(1-\Delta_z) + \\ & L(x,y+1,z)(1-\Delta_x)\Delta_y(1-\Delta_z) + L(x,y,z+1)(1-\Delta_x)(1-\Delta_y)\Delta_z + \\ & L(x+1,y,z+1)\Delta_x(1-\Delta_y)\Delta_z + L(x,y+1,z+1)(1-\Delta_x)\Delta_y\Delta_z + \\ & L(x+1,y+1,z)\Delta_x\Delta_y(1-\Delta_z) + L(x+1,y+1,z+1)\Delta_x\Delta_y\Delta_z \end{aligned} \tag{4.3}$$

na qual $L$ é a nuvem de pontos original e $K$ é o resultado do pré-processamento. Na Figura 9 é possível observar o resultado do pré-processamento realizado sobre uma amostra de cada classe. Para a classe "Árvore" a amostra foi capturada originalmente com 9960 pontos, e pode ser vista na Figura 9a, o resultado visto Figura 9b tem 2048 pontos e percebe-se que a densidade de pontos diminui, como esperado. Já para a classe "Carro", a amostra original vista na Figura 9c continha 14911 pontos, e após esta etapa obtém-se a nuvem vista na Figura 9d, também com 2048 pontos. E, para a terceira classe, de "Pessoa", tem-se na Figura 9e uma nuvem com 8168 pontos após captura, e após pré-processada, esta nuvem ficou com 2048 pontos como se observa na Figura 9f.

Essa etapa é necessária para adequar o conjunto de dados às entradas das CNNs utilizadas, uma vez que para possibilitar a utilização de métodos como esses, é preciso ter dimensões padronizadas para cada amostra pertencente ao conjunto de dados. Logo, somente após esta etapa será possível passar para a etapa seguinte, que é a de Classificação dos dados.

## 4.3 Classificação

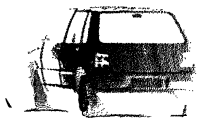
Após a aquisição e rotulação das nuvens de pontos, o conjunto de dados foi embaralhado e particionado em três partes de tamanhos iguais. Essa divisão foi feita para que fosse possível a realização de Validação Cruzada (REFAEILZADEH *et al.*, 2016) como meio de gerar resultados para cada modelo e assim mensurar o seu desempenho utilizando as métricas de avaliação.

A Validação cruzada é um método que engloba treinamentos e testes de um modelo de Aprendizagem de Máquina em diferentes arranjos de um mesmo conjunto de dados. Como pode ser visto na Figura 10, um 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 10, e o subconjunto de teste formado pela partição restante, em laranja na Figura 10. Em cada uma das $n$ iterações o subconjunto de treino é

Figura 9 – Amostras de nuvens de pontos originais (à esquerda) e amostras de nuvens de pontos pré-processadas.

(a) Amostra de árvore original.

(b) Amostra de árvore pré-processada.



(c) Amostra de carro original.

(d) Amostra de carro pré-processada.

(e) Amostra de pessoa original.

(f) Amostra de pessoa pré-processada.

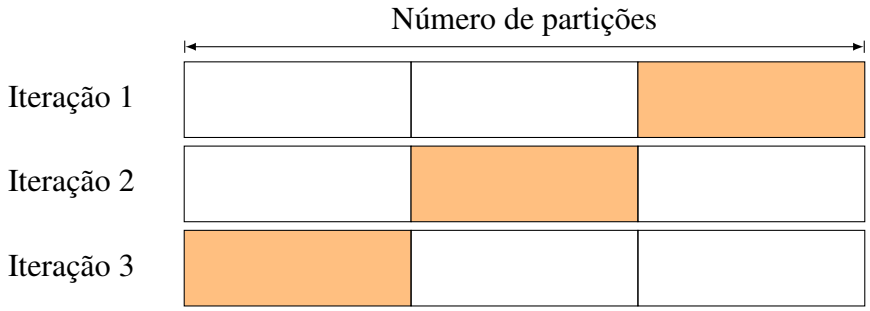
Fonte: Elaborada pelo autor.

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 mesmo conjunto de dados, tendo ao final deste 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 por 50 épocas, mantendo as configurações originais de quando foram propostas, modificando a quantidade de neurônios nas camadas de saída de 40, números de classe da ModelNet40, para 3, número de classes deste conjunto de dados.

Na Figura 11 estão dispostas as classes de todas as nuvens de pontos de cada um dos três subconjuntos de dados de treino. Já na Figura 12 tem-se as quantidades de amostras de cada classe para os subconjuntos de teste. Observa-se que existem amostras de todas as classes em

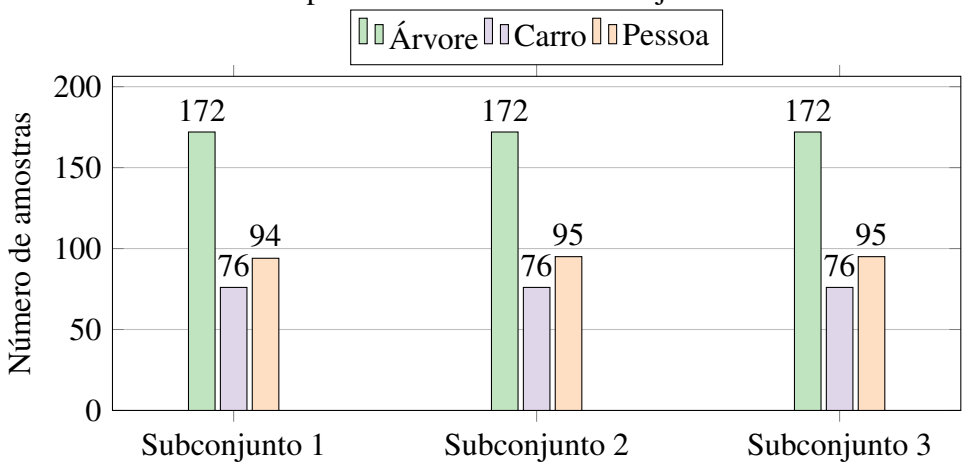
Figura 10 – Validação cruzada para três partições (K=3) com três classes.




Fonte: Elaborada pelo autor.

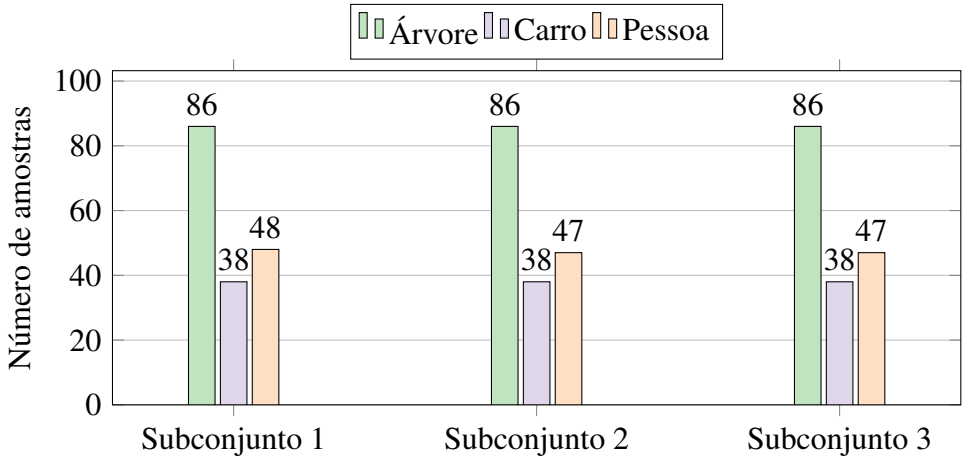
cada partição, de modo que não faltará conhecimento sobre determinada classe durante cada treinamento e nem amostras desta para predição durante cada teste.

Figura 11 – Número de amostras por classes em cada subconjunto de treino.




Fonte: Elaborada pelo autor.

Figura 12 – Número de amostras por classes em cada subconjunto de teste.




Fonte: Elaborada pelo autor.

Uma vez que a Validação Cruzada é realizada, obtém-se um vetor de predições

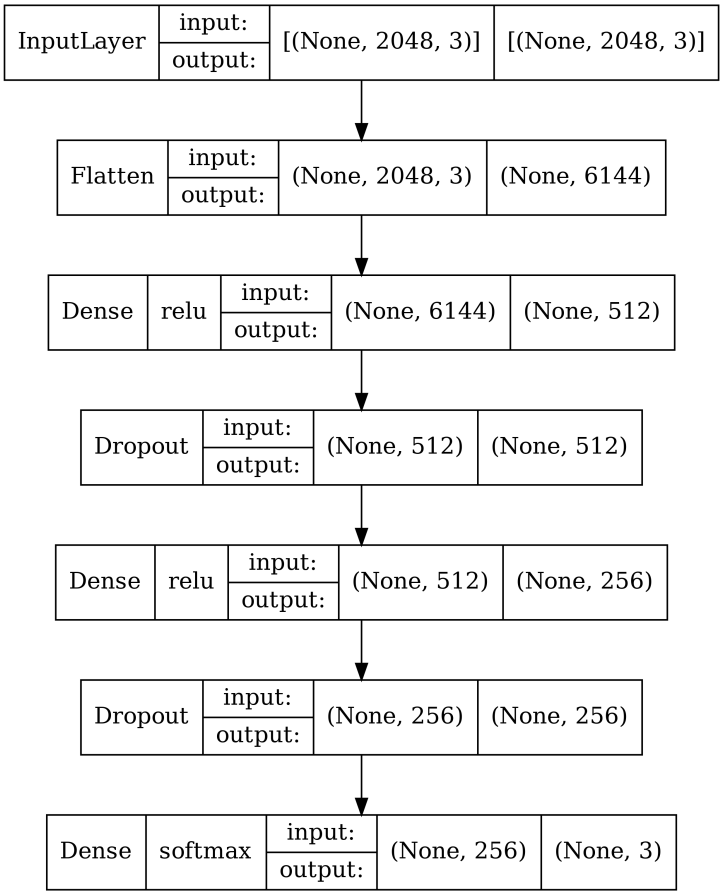
para cada iteração feita, totalizando três vetores, que serão utilizados para gerar as métricas de avaliação de cada partição em específico, que serão analisadas na Seção de Resultados.

## 4.4 Otimização de hiper-parâmetros da Lidar3DNetV1

Além do experimento com Validação Cruzada descrito na Seção anterior, nesta Seção um segundo experimento será descrito, a Otimização dos hiper-parâmetros da rede Lidar3DNetV1.

Na Figura 13 observa-se a arquitetura original da Lidar3DNetV1. Quando foi proposta, a rede continha uma camada *Flatten* para achatar a matriz de entrada para um vetor, seguida de uma camada *Dense* com 512 neurônios com a função de ativação *relu*. Um *Dropout* foi adicionado após a segunda camada, com taxa de 20% dos neurônios, seguido por outra camada *Dense* com 256 neurônios e a mesma função de ativação da camada anterior de mesmo tipo, e após esta um novo *Dropout* de mesma taxa. Ao final, a saída da rede é uma camada *Dense* com 3 neurônios e com a função de ativação *softmax*.

Figura 13 – Arquitetura original do modelo Lidar3DNetV1.




Fonte: Elaborada pelo autor.

Para a otimização dos hiper-parâmetros da Lidar3DNetV1, foi utilizado o *framework* *KerasTuner* (O'MALLEY *et al.*, 2019), com este é possível realizar diversos tipos de busca dentro de um espaço de hiper-parâmetros pré-definido. Dentre os tipo de otimização presente neste *framework*, tem-se a Otimização Bayesiana, *Hyperband*, e Busca Aleatória. Para este experimento, foi escolhida a Busca Aleatória (BERGSTRA; BENGIO, 2012), que consiste na escolha de pontos aleatórios dentro do espaço *n*-dimensional dos hiper-parâmetros.

O espaço escolhido foi composto por seis hiper-parâmetros, o primeiro deles foi a quantidade de neurônios da primeira camada *Dense* com valor mínimo de 32, valor máximo de 1024, e com passo 32. O segundo foi a função de ativação da primeira camada, variando entre *relu*, *tanh*, sigmoide. Já o terceiro hiper-parâmetros escolhido foi a inserção ou não do *dropout* com taxa de 20% após a primeira camada densa. Outro hiper-parâmetro inserido na busca foi a quantidade de neurônios da segunda camada *Dense*, desta vez com valor mínimo de 32, máximo de 512 e passo de 32. Para esta segunda camada, as mesmas funções de ativação foram escolhidas, assim como utilizar ou não o *dropout* na sequência.

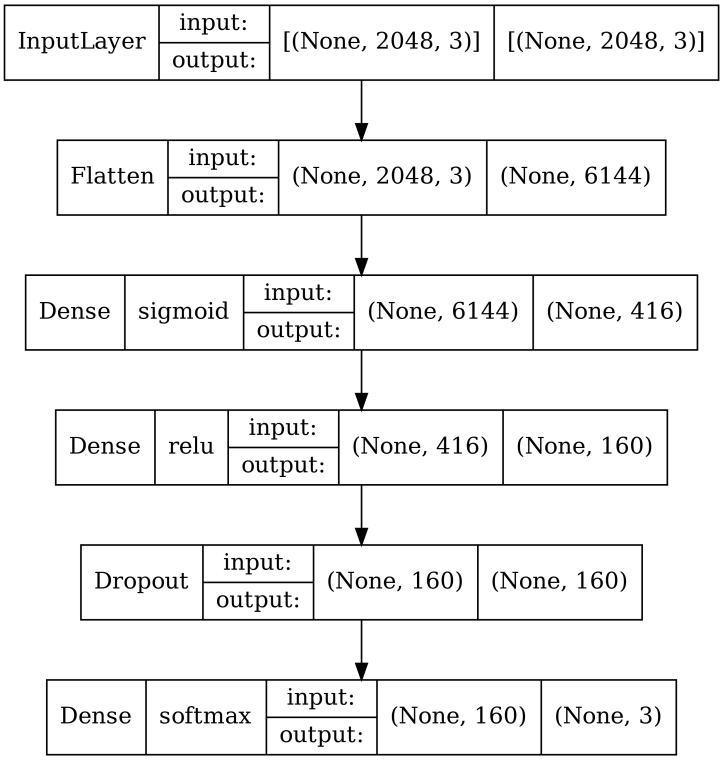
Na Figura 14 tem-se a estrutura resultante da Lidar3DNetV1 com os melhores hiper-parâmetros encontrados na Busca Aleatória. Para a primeira camada densa, o número ótimo de neurônios foi 416, 96 neurônios a menos que a mesma camada da versão original. A função de ativação atrelada a esta camada encontrada durante a busca foi a sigmoide, enquanto a original era *relu*. Durante a busca, constatou-se que não utilizar o primeiro *dropout* seria melhor, assim como na segunda camada densa a quantidade ideal de neurônios seria 160, cerca de 60% das unidades da versão original.

O intuito de realizar a otimização dos hiper-parâmetros, é que mesmo não melhorando os resultados obtidos sobre o conjunto de dados em relação ao modelo original, o modelo otimizado mantenha um patamar parecido em métricas de avaliação, reduzindo o número de parâmetros gerais, reduzindo consequentemente a complexidade da rede e sua alocação de memória e espaço em disco, que são fatores importantes relacionados ao desempenho em aplicações do mundo real.

## 4.5 Métricas de avaliação

Afim de avaliar o desempenho das redes utilizadas neste trabalho, cinco diferentes métricas foram escolhidas: Acurácia, Precisão, Sensibilidade, F1-*Score* e Especificidade. Estas métricas são calculadas a partir das frequências de classificação de cada classe para uma rede

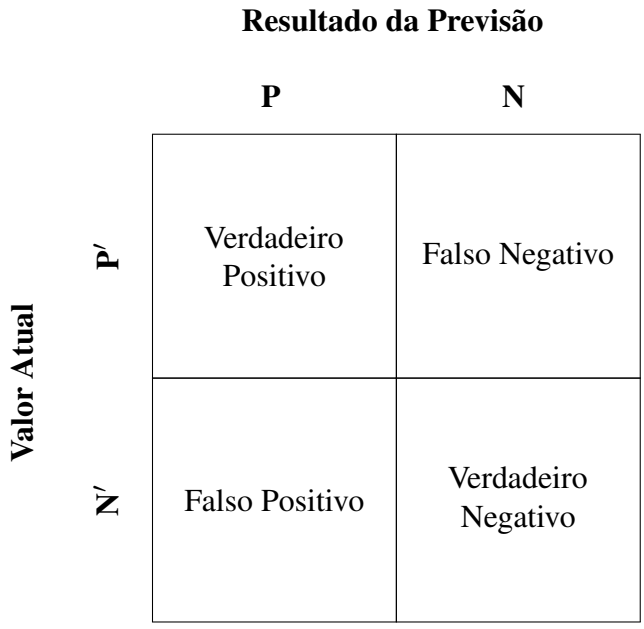
Figura 14 – Arquitetura do modelo Lidar3DNetV1 após otimização dos hiper-parâmetros.




Fonte: Elaborada pelo autor.

específica. Tais frequências estão dispostas na matriz de confusão, que pode ser vista na Figura 15.

Figura 15 – Matriz de confusão para a classificação de objetos.




Fonte: Elaborada pelo autor.

Existem quatro tipos de frequências, o Verdadeiro Positivo (VP) que corresponde à previsão correta da classe para uma amostra rotulada, classificando uma amostra como pessoa quando realmente é da classe pessoa. Já o Verdadeiro Negativo (VN) que representa a previsão correta para a classe que a amostra não é rotulada, corresponde a reconhecer a outra classe corretamente quando a amostra for de carro, por exemplo. Enquanto que o Falso Positivo (FP) é a frequência de previsões incorretas de uma classe para a qual a amostra está rotulada, como reconhecer como carro uma amostra da classe pessoa. Por fim, o Falso Negativo (FN) é a previsão incorreta para a classe que a amostra não pertence, como reconhecer a outra classe incorretamente quando a amostra for de árvore, por exemplo.

A Acurácia (FAWCETT, 2006) é a proporção entre as previsões corretas e todas as previsões realizadas por uma rede. É calculada por meio da Equação 4.4.

$$Acurácia = \frac{VP+VN}{VP+FP+FN+VN}. \tag{4.4}$$

A partir da Equação 4.5 é possível obter a Precisão (FAWCETT, 2006) do modelo, como um razão entre as previsões positivas corretas e todas as previsões positivas obtidas.

$$Precisão = \frac{VP}{VP+FP} \tag{4.5}$$

A Sensibilidade (SOKOLOVA; LAPALME, 2009) é calculada usando a Equação 4.6, que representa a relação entre as previsões positivas corretas e previsões negativas erradas.

$$Sensibilidade = \frac{VP}{VP+FN} \tag{4.6}$$

Já o F1-*Score* (SASAKI *et al.*, 2007), dado pela Equação 4.7, computa um balanceamento entre as métricas de Precisão e Sensibilidade, de modo que que ambas estejam próximas de 100% para se aproximar de tal valor.

$$F1-Score = 2 \cdot \frac{Precisão \cdot Sensibilidade}{Precisão + Sensibilidade} \tag{4.7}$$

Enquanto que a Especificidade (FAWCETT, 2006) é calculada usando a Equação 4.8, e reflete a relação entre as previsões negativas corretas e previsões positivas erradas.

$$Especificidade = \frac{VN}{VN+FP} \tag{4.8}$$

Assim, utilizando todas as métricas acima, em conjunto com os testes estatísticos que serão apresentados a seguir, se torna possível avaliar o desempenho de cada rede de forma completa na tarefa de reconhecimento de objetos em 3D, ampliando a análise das classificações em relação aos *benchmarks* presentes no estado da arte, como o *ModelNet40*, por exemplo.

## 4.6 Testes de hipóteses

Para definir a melhor rede para a tarefa de classificação de objetos 3D, não é correto olhar apenas para as métricas de avaliação utilizadas. Deste modo, se faz necessário realizar testes estatísticos nos quais, a partir de hipóteses criadas previamente, será possível determinar qual das dez redes é diferentes das demais CNNs escolhidas e, a partir das métricas, escolhe-la como a melhor para desempenhar a tarefa de classificação.

### *4.6.1 ANOVA*

Inicialmente, foi realizado o Teste ANOVA (Análise de Variância) (GIRDEN, 1992) com a finalidade de apontar se as redes utilizadas apresentam resultados com diferenças significativas entre si. Para este teste, o valor de $p < 0,05$ caracteriza a rejeição da hipótese nula $\mathbf{H_0}$. Logo, a hipótese alternativa $\mathbf{H_1}$ torna-se a hipótese verdadeira. A confiança utilizada para realizar este teste foi de 0,95, e as hipóteses estão listadas abaixo:

**Hipótese 0 ( $\mathbf{H_0}$ ):** *Todas as redes são estatisticamente equivalentes. O que significa que não há diferença entre usar uma ou outra rede para a tarefa de Classificação de objetos 3D.*

**Hipótese 1 ( $\mathbf{H_1}$ ):** *Pelo menos uma rede é diferente de outra. O que significa que existe uma rede com resultados melhores ou piores que as outras.*

Entretanto o teste ANOVA não retrata quais redes são diferentes das demais, e assim não é possível afirmar que uma rede é melhor ou pior que outra.

### *4.6.2 Tukey's HSD*

Para isso é necessário realizar o Teste Tukey's HSD (Diferença Honestamente Significativa de Tukey) (ABDI; WILLIAMS, 2010), o qual compara as redes uma a uma e foi realizado com a mesma regra para aceitação ou rejeição das hipóteses a seguir:

**Hipótese 0 ( $H_0$ ):** *A rede em questão é equivalente à outra rede em comparação. O que significa que não há diferença entre usar uma ou outra rede para a tarefa de Classificação de objetos 3D.*

**Hipótese 1 ( $H_1$ ):** *As redes são diferentes entre si. Logo, existe diferença de significância estática entre usar uma ou outra rede para a tarefa de Classificação de objetos 3D.*

Os testes acima foram realizados usando os valores de cada uma das cinco métricas de avaliação obtidos para cada arranjo da Validação Cruzada. Deste modo, as hipóteses citadas só serão aceitas ou recusadas quando os valores de $p$ de cada métrica obedecerem, simultaneamente, à condição estabelecida anteriormente.

# 5 RESULTADOS E DISCUSSÕES

Os resultados deste trabalho são apresentados em duas etapas. Na primeira etapa, a Seção 5.1, todas as dez redes de Classificação de objetos 3D foram comparadas, havendo uma análise das métricas alcançadas, realizando testes estatísticos para determinar os melhores métodos, além de uma análise do tempo de processamento. Já na etapa seguinte, Seção 5.2, dois modelos da Lidar3DNetV1 foram comparados, o original e o otimizado, que foi proposto neste trabalho, analisando as métricas de avaliação e determinando as diferenças estatísticas entre ambos.

Todos os experimentos foram realizados em um computador com processador Intel Core i7, 8 GB de RAM, placa de vídeo *GeForce* GTX 1070 e sistema operacional Linux LTS 16.04. Para implementação dos algoritmos e obtenção dos resultados, foram utilizados os *frameworks Scikit-learn*, *Pytorch*, *Tensorflow* e *Keras* através da linguagem *Python*.

## 5.1 Modelos originais

Nesta Seção, são apresentados os resultados de trinta experimentos, uma vez que os dez modelos de Classificação de objetos 3D foram testados em cada um dos três arranjos de dados da Validação Cruzada.

A Tabela 1 apresenta a Acurácia, o F1-*Score*, a Precisão, a Sensibilidade e a Especificidade que foram obtidos ao final da classificação dos subconjuntos de teste durante a Validação Cruzada. A Tabela 1 mostra que a rede RSCNN foi a que alcançou as maiores métricas, 100.00% nas cinco, todas ocorrendo no terceiro subconjunto de teste.

A Tabela 1 também mostra que apenas as redes LDGCNN e RSCNN obtiveram mais de 90,00% em todas as métricas para os subconjuntos 2 e 3, destacadas em verde. Além disso, observa-se que apenas as redes PVT e RSCNN alcançaram valores maiores que 90,00% em todos os subconjuntos de teste, para todas as métricas com exceção da Precisão.

Enquanto que a 3DMedPT atingiu uma Acurácia de 93,40% na classificação das nuvens presentes no ModelNet40, porém, para os subconjuntos 2 e 3 deste conjunto de dados, a rede não chegou até 65% para todas as métricas. Tal fato já era esperado, uma vez que a 3DMedPT foi proposta inicialmente para a classificação de Aneurismas Intracranianos 3D e foi adaptada para a classificação de objetos 3D.

Olhando para a rede Lidar3DNetV1, precebe-se que esta obteve pelo menos 70%

Tabela 1 – Métricas para cada Rede em cada um dos conjuntos de teste de nuvens de pontos capturadas.

| **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 |
| Lidar3DNetV1 | 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 | 92,50 | 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.

em todas as métricas para cada um dos subconjuntos de teste. Além disso, no subconjunto 2 a Lidar3DNetV1 alcançou a maior Especificidade entre todas, com 93,00%, ficando atrás apenas da RSCNN, a qual atingiu 98,74% na mesma métrica. Logo, é possível afirmar que essas redes foram as que mais acertaram quando uma nuvem de pontos realmente não era de determinada classe.

A CurveNet alcançou 94,20% na tarefa de classificação sobre o ModelNet40, já neste conjunto de dados essa rede atingiu pelo menos 85% em todas as métricas de avaliação sobre todos os subconjuntos de teste. O desempenho da CurveNet diminuiu, possivelmente, pelo fato de haver objetos que mesclam curvas longas e curtas dentro do conjunto proposto, uma vez que essa rede se baseia nesse tipo de característica para agrupar objetos de mesma classe.

Ao mesmo tempo que a rede GBNet, que alcançou uma Acurácia de 91,04% no conjunto de dados ModelNet40, sobre o conjunto de dados proposto neste trabalho não atingiu 60% em qualquer uma das métricas de avaliação para todos os subconjuntos utilizados. Deste modo, é possível afirmar que a rede em questão não foi proposta para a tarefa de classificação de objetos 3D em geral e sim para classificar as amostras presentes no conjunto ModelNet40.

Já a rede PAConv ficou com métricas entre 70 e 85%, dentre todos subconjuntos de teste. Essa rede havia alcançado 93,9% de Acurácia no ModelNet40 (WU *et al.*, 2015), como visto na Tabela 2. Tal redução está relacionada à maneira como a qual a PAConv realiza o processo de treinamento, utilizando porções locais da nuvem de pontos, que podem não estar presentes em casos de captura no mundo real, como ocorre no conjunto de dados utilizado.

Tabela 2 – Métricas para cada Rede sobre o conjunto de dados ModelNet40.

| **Rede** | **Acurácia** |
|---|---|
| 3DMedPT | 93,40 |
| Lidar3DNetV1 | 80,63 |
| CurveNet | 94,20 |
| GBNet | 91,04 |
| LDGCNN | 92,90 |
| PAConv | 93,90 |
| PointNet | 89,20 |
| PVT | 94,00 |
| RSCNN | 93,60 |
| SimpleView | 93,30 |

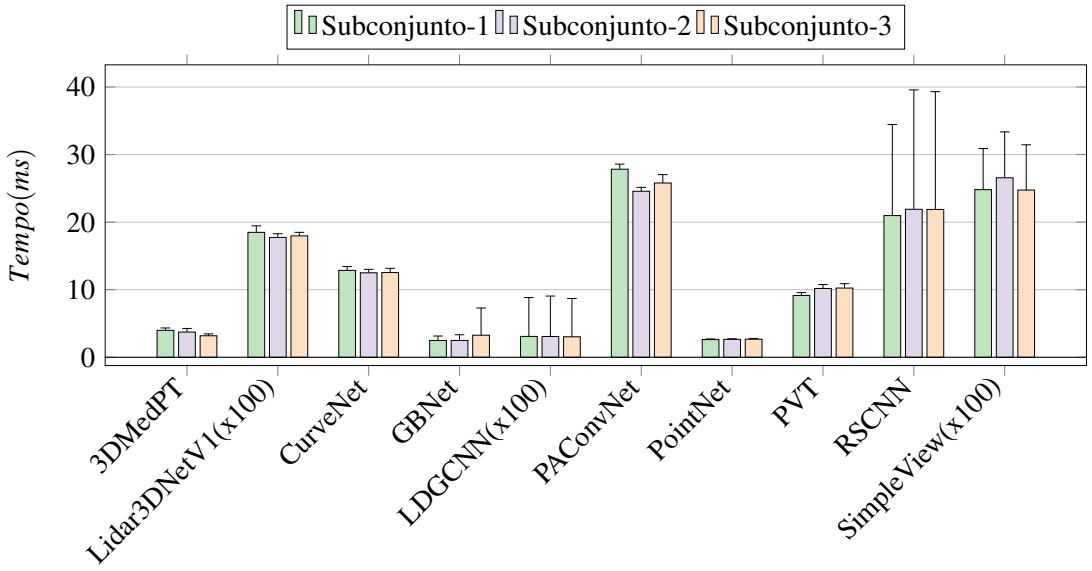
Fonte: Elaborada pelo autor.

A Tabela 2 mostra que a PointNet atingiu 89,20% na tarefa de classificação do conjunto ModelNet40, já ao se observar a Tabela 1, esta rede obteve as cinco métricas para todos os subconjuntos de teste condensadas entre 80 e 90%. Por ser uma rede que se propõe às tarefas de classificação, segmentação de objetos e segmentação semântica, apresenta resultados constantes, mas um pouco abaixo do que foi alcançado no conjunto para o qual foi proposta.

A última rede da Tabela 1, é a SimpleView, a qual obteve suas métricas entre 85% e 95%. Observando a Tabela 2 esta rede alcançou 93,30% ao classificar as nuvens de pontos do conjunto ModelNet40. Esta redução de desempenho deve-se ao fato de que, diferente das nuvens presentes no ModelNet40, neste conjunto as nuvens não apresentam pontos nas regiões que o laser do LiDAR não reflete, assim, como a SimpleView utiliza diversas vistas para realizar a classificar do objeto, acaba tendo seu funcionamento prejudicado.

Após feita a analise do desempenho a partir das métricas de avaliação, também é importante avaliar-se o tempo de processamento das redes utilizadas neste experimento, uma vez que não basta ter métricas melhores, se o método for lento e não for possível utilizá-lo em aplicações em tempo real. Na Figura 16 é apresentado o gráfico de tempo médio e desvio padrão em milissegundos por amostra de todas as redes em cada subconjunto de teste.

Figura 16 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em milissegundos, para todas as redes do experimento.




Fonte: Elaborada pelo autor.

Observano o gráfico, três redes se destacam por serem as mais rápidas neste experimento, são elas: Lidar3DNet, Simpleview e LDGCNN. A primeira, por se tratar de uma rede de arquitetura mais simplista, sendo uma MLP, a Lidar3DNet teve um tempo médio de processa-

mento próximo à 200 $\mu$s para cada uma das amostras nos subconjuntos de teste. Enquanto que a SimpleView, também teve destaque, alcançado 250 $\mu$s, porém por se tratar de um modelo que busca diversas visualizações da nuvem de pontos, se torna mais lenta que a Lidar3DNet. Já a LDGCNN, mesmo com uma arquitetura mais complexa, com camadas convolucionais, sendo uma CNN, atingiu, em média, 25 $\mu$s, sendo a mais rápida entre todas as redes deste experimento.

Apesar de ter alcançado as maiores métricas de avaliação em todos os subconjuntos, a rede RSCNN se mostrou uma das mais lentas durante o experimento, uma vez que teve um tempo médio de processamento de próximo a 20ms, sendo mais rápida apenas que a rede PAConvNet, que por sua vez atingiu um tempo médio de aproximadamente 25ms, sendo a mais lenta rede utilizada neste experimento.

Após analisar as métricas de avaliação alcançadas, assim como tempo médio de processamento de todas as redes utilizadas, na Seção seguinte, os testes estatísticos serão dispostos, de modo que será possível definir e escolher a rede de melhor desempenho, uma vez que já se sabe quais delas são as mais rápidas.

### *5.1.1 Testes Estatísticos*

Nesta seção serão realizados os testes de significância estatística ANOVA e Tukey's HSD, nos quais, a partir do valor de $p$ as hipóteses dispostas na Seção 4.6. Na Tabela 3 é visto que para todas as métricas de avaliação o valor de $p$ é próximo de zero, logo sendo menor que o limiar definido de 0,05. Dessa forma, a hipótese nula $\mathbf{H_0}$ é rejeitada, e consequentemente a hipótese alternativa $\mathbf{H_1}$ é aceita, logo é possível afirmar que as redes utilizadas neste experimento são diferentes entre si na tarefa de Classificação de objetos 3D.

Sabendo que as redes são diferentes entre si, agora é necessário conhecer quais redes se diferenciam das demais, apenas após isso que será possível constatar qual a melhor rede e qual é a pior rede para a tarefa de Classificação de objetos 3D. Para tal, o teste Tukey's HSD é realizado de modo que a partir do valor de $p$ para cada par de redes em comparação é possível dizer se são ou não diferentes entre si. Na Tabela 4 estão dispostos 45 pares de redes que foram comparadas estatisticamente, cada rede foi comparada com as outras nove redes, deste modo que a análise se torna completa.

A partir dos valores de $p$ obtidos para cada par de redes dispostos na Tabela 4 é possível afirmar que:

– a rede 3DMedPT é estatisticamente diferente das demais, logo, por ter atingindo mé-

Tabela 3 – Resultados do teste *one-way* ANOVA para experimentos sobre os dados originais.

| **Métrica** | | **Df** | **SS** | **MS** | **F-*value*** | **p-*value*** |
|---|---|---|---|---|---|---|
| Acurácia | C(Redes) | 9,000000 | 0,729410 | 0,081046 | 45,851185 | 0,000000 |
| | Residual | 20,000000 | 0,035352 | 0,001768 | - | - |
| F1-*Score* | C(Redes) | 9,000000 | 1,136268 | 0,126252 | 46,750031 | 0,000000 |
| | Residual | 20,000000 | 0,054012 | 0,002701 | - | - |
| Precisão | C(Redes) | 9,000000 | 1,144902 | 0,127211 | 56,849063 | 0,000000 |
| | Residual | 20,000000 | 0,044754 | 0,002238 | - | - |
| Sensibilidade | C(Redes) | 9,000000 | 0,960530 | 0,106726 | 59,743538 | 0,000000 |
| | Residual | 20,000000 | 0,035728 | 0,001786 | - | - |
| Especificidade | C(Redes) | 9,000000 | 0,235371 | 0,026152 | 82,919091 | 0,000000 |
| | Residual | 20,000000 | 0,006308 | 0,000315 | - | - |

Fonte: Elaborado pelo autor.

tricas maiores apenas que aquelas atingidas pela GBNet, esta rede teve o segundo pior desempenho;

- a Lidar3DNet é melhor que 3DMedPT e GBNet, não apresenta diferença significativa entre as redes CurveNet, LDGCNN, PAConv e PVT, sendo pior que a RSCNN, apenas;
- como GBNet também apresenta diferença significativa das demais, e por tem a menores métricas dentre todos, logo pode ser caracterizada como a pior rede para a Classificação de objetos 3D;
- PAConv e PointNet são melhores que 3DMedPT e GBNet, piores que a RSCNN e semelhantes ao restante;
- CurveNet, LDGCNN, PVT, RSCNN e SimpleView são classificadores de objetos semelhantes entre si, e são melhores que 3DMedPT e GBNet.

Deste modo é possível afirmar que as cinco melhores redes deste experimento são: CurveNet, LDGCNN, PVT, RSCNN e SimpleView. Logo fica a critério a escolha dentre estas redes. Considerando a semelhança estatística, a escolha mais plausível passa a ser a rede de tempo de processamento menor, neste caso é a rede LDGCNN. Uma vez que, apesar de apresentar métricas maiores, não é possível afirmar que a RSCNN é a melhor rede para esta tarefa de classificação, já que o resultado do teste acima não mostrou essa possibilidade.

## 5.2 Otimização de hiper-parâmetros da Lidar3DNetV1

A partir da otimização de hiper-parâmetros descrita na Seção 4.4, nesta Seção serão apresentados os resultados alcançados com o uso deste método, assim como a comparação da

Tabela 4 – Resultados do teste Tukey HSD para experimentos sobre os dados originais.

| **Grupo 1** | **Grupo 2** | **Acurácia** | **F1-*Score*** | **Precisão** | **Sensenb.** | **Especif.** |
|---|---|---|---|---|---|---|
| 3DMedPT | Lidar3DNetV1 | 0,005629 | 0,010188 | 0,087954 | 0,016942 | 0,008999 |
| 3DMedPT | CurveNet | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| 3DMedPT | GBNet | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| 3DMedPT | LDGCNN | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| 3DMedPT | PAConv | 0,002095 | 0,011487 | 0,050712 | 0,041793 | 0,002219 |
| 3DMedPT | PointNet | 0,001000 | 0,001000 | 0,004344 | 0,003796 | 0,001000 |
| 3DMedPT | PVT | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| 3DMedPT | RSCNN | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| 3DMedPT | SimpleView | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| Lidar3DNetV1 | CurveNet | 0,172596 | 0,247343 | 0,181655 | 0,087883 | 0,060422 |
| Lidar3DNetV1 | GBNet | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| Lidar3DNetV1 | LDGCNN | 0,057221 | 0,097651 | 0,074023 | 0,016958 | 0,012098 |
| Lidar3DNetV1 | PAConv | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| Lidar3DNetV1 | PointNet | 0,900000 | 0,893059 | 0,900000 | 0,581628 | 0,618743 |
| Lidar3DNetV1 | PVT | 0,066545 | 0,087792 | 0,073068 | 0,012362 | 0,012226 |
| Lidar3DNetV1 | RSCNN | 0,001000 | 0,002006 | 0,001000 | 0,001000 | 0,001000 |
| Lidar3DNetV1 | SimpleView | 0,418509 | 0,343990 | 0,315486 | 0,046234 | 0,102088 |
| CurveNet | GBNet | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| CurveNet | LDGCNN | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| CurveNet | PAConv | 0,356023 | 0,226121 | 0,287633 | 0,086852 | 0,199141 |
| CurveNet | PointNet | 0,880565 | 0,900000 | 0,900000 | 0,900000 | 0,872569 |
| CurveNet | PVT | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| CurveNet | RSCNN | 0,327612 | 0,397104 | 0,258764 | 0,198420 | 0,171936 |
| CurveNet | SimpleView | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| GBNet | LDGCNN | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| GBNet | PAConv | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| GBNet | PointNet | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| GBNet | PVT | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| GBNet | RSCNN | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| GBNet | SimpleView | 0,001000 | 0,001000 | 0,001000 | 0,001000 | 0,001000 |
| LDGCNN | PAConv | 0,135976 | 0,087679 | 0,125391 | 0,016662 | 0,046689 |
| LDGCNN | PointNet | 0,564758 | 0,731654 | 0,642487 | 0,579310 | 0,456383 |
| LDGCNN | PVT | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| LDGCNN | RSCNN | 0,644792 | 0,683975 | 0,508416 | 0,593129 | 0,528693 |
| LDGCNN | SimpleView | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| PAConv | PointNet | 0,900000 | 0,863063 | 0,900000 | 0,577173 | 0,900000 |
| PAConv | PVT | 0,156011 | 0,079197 | 0,124504 | 0,012146 | 0,047145 |
| PAConv | RSCNN | 0,002529 | 0,001775 | 0,001324 | 0,001000 | 0,001000 |
| PAConv | SimpleView | 0,665211 | 0,316643 | 0,465611 | 0,045466 | 0,307413 |
| PointNet | PVT | 0,605406 | 0,702010 | 0,638917 | 0,499756 | 0,459061 |
| PointNet | RSCNN | 0,020539 | 0,045496 | 0,016298 | 0,017748 | 0,008434 |
| PointNet | SimpleView | 0,900000 | 0,900000 | 0,900000 | 0,838279 | 0,900000 |
| PVT | RSCNN | 0,604144 | 0,713616 | 0,511987 | 0,672684 | 0,526107 |
| PVT | SimpleView | 0,900000 | 0,900000 | 0,900000 | 0,900000 | 0,900000 |
| RSCNN | SimpleView | 0,126817 | 0,289391 | 0,144828 | 0,332393 | 0,105149 |

Fonte: Elaborado pelo autor.

rede original Lidar3DNetV1 com a rede com hiper-parâmetros otimizados, que será chamada de Lidar3DNetV1Opt.

Na Tabela 5 estão apresentadas as cinco métricas de avaliação para as redes original e otimizada sobre os três subconjuntos de teste. Observando os resultados para o subconjunto 1, percebe-se que o modelo otimizado, Lidar3DNetV1Opt, atingiu valores maiores que o modelo original, em todas as métricas. O mesmo ocorreu para o terceiro subconjunto, no qual o modelo otimizado teve todas as métricas maiores que 80% pelo menos, algo que não aconteceu para a rede com hiper-parâmetros de origem.

Tabela 5 – Métricas para a Lidar3DNetV1 e a Lidar3DNetV1 com hiper-parâmetros otimizados.

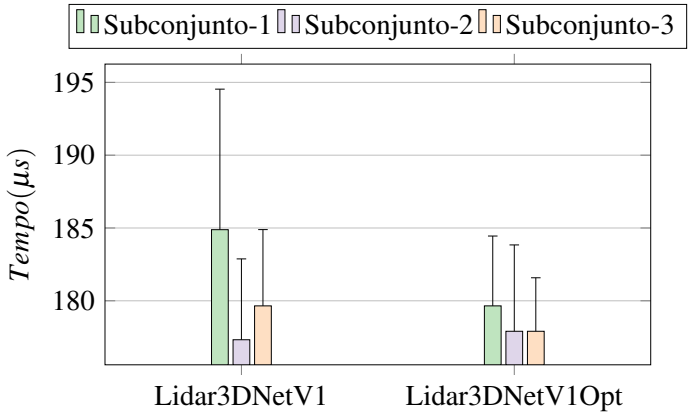
| **Rede** | **Subconjunto** | **Acurácia** | **F1-*Score*** | **Precisão** | **Sensib.** | **Especif.** |
|---|---|---|---|---|---|---|
| Lidar3DNetV1 | 1 | 80,23 | 77,13 | 77,75 | 76,65 | 89,54 |
| | 2 | 84,80 | 83,06 | 82,22 | 84,87 | 93,00 |
| | 3 | 75,44 | 71,24 | 71,47 | 71,69 | 87,84 |
| Lidar3DNetV1Opt | 1 | 81,40 | 78,89 | 80,79 | 78,77 | 90,17 |
| | 2 | 84,21 | 82,28 | 81,81 | 83,03 | 92,18 |
| | 3 | 83,04 | 81,36 | 81,05 | 81,93 | 91,38 |

Fonte: Elaborado pelo autor.

Ainda analisando a Tabela 5, nota-se que apenas no segundo subconjunto de teste é que a Lidar3DNetV1 desempenhou melhor que a Lidar3DNetV1Opt. Assim como no experimento anterior, neste também será necessário realizar os testes de significância estatística para determinar se os modelos apresentam diferenças entre si e assim ser possível afirmar que um é melhor do que outro. Outro ponto que deve ser avaliado é o tempo de processamento deste modelo, e para tal é necessário observar o gráfico presente na Figura 17, no qual percebe-se que em nenhum dos subconjuntos de teste a Lidar3DNetV1Opt passou de 180$\mu$s, permanecendo entre este valor e 175$\mu$s. Já a Lidar3DNetV1 demorou em média 185$\mu$s para processar uma amostra do primeiro subconjunto de teste.

Além de apresentar um tempo médio menor que o modelo original, a Lidar3DNetV1-Opt tem uma quantidade total de parâmetros menor, são 3.278.339 no modelo original contra 2.623.523 do modelo otimizado, uma redução de aproximadamente 20%. Visando a utilização destes modelos em aplicações reais e a preocupação com espaço alocado por estes, observou-se que o modelo original tinha um tamanho em disco de 112,7 kB, enquanto que o modelo otimizado passou a ocupar 102,0 kB de espaço em memória, reduzindo quase 10% de espaço alocado.

Figura 17 – Tempo médio e desvio padrão para executar as predições sobre cada amostra das nuvens de pontos, em microssegundos, para a redes original e otimizada.




Fonte: Elaborada pelo autor.

### *5.2.1 Testes Estatísticos*

Para determinar se existe um modelo melhor dentre o original e o otimizado, primeiro é preciso identificar se existe diferença entre ambos modelos da Lidar3DNetV1. Para isso o teste ANOVA e Tukey's HSD foram realizados sobre os vetores de métricas obtidos na Validação Cruzada para ambos modelos.

Neste caso, por se tratar de uma comparação de dois elementos apenas, os resultados obtidos pelo teste ANOVA serão os mesmos para o teste Tukey's HSD. Assim é possível analisar simultaneamente os valores de $p$ com as Tabelas 6 e 7, observando que em nenhuma das métricas o valor de $p$ é menor que 0.05, assim a hipótese nula é aceita e é possível afirmar que os modelos não apresentam diferenças significativas entre si.

Tabela 6 – Resultados do teste *one-way* ANOVA para experimento com otimização de hiperparâmetros.

| **Métrica** | | **Df** | **SS** | **MS** | **F-*value*** | **p-*value*** |
|---|---|---|---|---|---|---|
| Acurácia | C(Redes) | 1.000000 | 0.001115 | 0.001115 | 0.933627 | 0.388639 |
| | Residual | 4.000000 | 0.004778 | 0.001195 | - | - |
| F1-*Score* | C(Redes) | 1.000000 | 0.002056 | 0.002056 | 1.081417 | 0.357106 |
| | Residual | 4.000000 | 0.007603 | 0.001901 | - | - |
| Precisão | C(Redes) | 1.000000 | 0.002483 | 0.002483 | 1.685693 | 0.263967 |
| | Residual | 4.000000 | 0.005892 | 0.001473 | - | - |
| Sensibilidade | C(Redes) | 1.000000 | 0.001844 | 0.001844 | 0.749551 | 0.435458 |
| | Residual | 4.000000 | 0.009843 | 0.002461 | - | - |
| Especificidade | C(Redes) | 1.000000 | 0.000189 | 0.000189 | 0.475180 | 0.528518 |
| | Residual | 4.000000 | 0.001590 | 0.000397 | - | - |

Fonte: Elaborado pelo autor.

Tabela 7 – Resultados do teste Tukey HSD para experimento com otimização de hiper-parâmetros.

| **Grupo 1** | **Grupo 2** | **Acurácia** | **F1-*Score*** | **Precisão** | **Sensenb.** | **Especif.** |
|---|---|---|---|---|---|---|
| Lidar3DNetV1Opt | Lidar3DNetV1 | 0.388638 | 0.357104 | 0.263966 | 0.435459 | 0.533854 |

Fonte: Elaborado pelo autor.

Desta forma, com conhecimento das vantagens que o modelo otimizado, Lidar3DNetV1Opt, tem em relação ao modelo original, Lidar3DNetV1, e sabendo que não existem diferenças de significância estatísticas entre elas, a escolha mais plausível dentre estas redes seria aquela com os hiper-parâmetros otimizados, uma vez que contém menos parâmetros, é mais leve, ocupa menos espaço em disco e tem tempo de processamento ligeiramente menor.

# 6 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresenta uma abordagem para Classificação de Objetos 3D sobre um conjunto de dados com três classes. Dois experimentos foram realizados, no primeiro dez redes foram testadas na tarefa de Classificação de objetos 3D e no segundo foi realizada a otimização dos hiper-parâmetros do modelo Lidar3DNetV1 para mensurar seu desempenho na mesma tarefa.

Foi realizada uma análise extensa de redes neurais na tarefa de Classificação de objetos 3D, os experimentos foram efetuados sobre um conjunto de dados de nuvens de pontos capturado como contribuição deste trabalho, onde cada amostra presente neste conjunto passou por uma etapa de pré-processamento para a normalização do número de pontos. A rede RSCNN produziu o melhor desempenho no primeiro experimento realizado, atigindo 100,00% em todas as métricas para um dos subconjuntos de teste, porém como este método não apresentou diferença estatística significativa para outros métodos como LDGCNN, que se mostrou 800 vezes mais rápido e tão eficiente quanto a RSCNN, alcançando 92,66% de Acurácia em dois dos subconjuntos. Logo a LDGCNN seria a escolha mais rápida e eficiente para a tarefa citada, realizando predições com alta precisão e com baixo custo computacional.

Já no segundo experimento uma otimização de hiper-parâmetros da rede Lidar3DNet foi realizada. Ao comparar o modelo otimizado, Lidar3DNetOpt, com a rede original, não houve diferença significativa relatada pelos testes estatísticos, de modo que o modelo otimizado apresentou desempenho tão bom quanto ao original, com menor tempo de processamento, menor número de parâmetros, menor espaço alocado em memória, sendo uma opção mais indicada para qualquer aplicação envolvendo Classificação de Objetos 3D.

Em trabalhos futuros, planeja-se implementar uma segunda versão da Lidar3DNet para as tarefas de detecção e segmentação de objetos 3D, assim como utilizar o modelo otimizado em aplicações reais, como em veículos autônomos que utilizem *LiDAR* como sensor para aquisição de nuvens de pontos. Além disso, pretende-se avaliar a viabilidade de se utilizar estes modelos em aplicações embarcadas com Internet das Coisas e Computação de Borda.

# REFERÊNCIAS

ABDI, H.; WILLIAMS, L. J. Tukey's honestly significant difference (hsd) test. **Encyclopedia of research design**, Sage Thousand Oaks, CA, v. 3, n. 1, p. 1–5, 2010.

AGARAP, A. F. Deep learning using rectified linear units (relu). **CoRR**, abs/1803.08375, 2018. Disponível em: http://arxiv.org/abs/1803.08375.

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

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.

ATZMON, M.; MARON, H.; LIPMAN, Y. Point convolutional neural networks by extension operators. **arXiv preprint arXiv:1803.10091**, 2018.

BAI, X.; HU, Z.; ZHU, X.; HUANG, Q.; CHEN, Y.; FU, H.; TAI, C.-L. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. **arXiv preprint arXiv:2203.11496**, 2022.

BALADO, J.; SOUSA, R.; DÍAZ-VILARIÑO, L.; ARIAS, P. Transfer learning in urban object classification: Online images to recognize point clouds. **Automation in Construction**, Elsevier BV, v. 111, p. 103058, mar. 2020. Disponível em: https://doi.org/10.1016/j.autcon.2019.103058.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **The Journal of Machine Learning Research**, JMLR. org, v. 13, n. 1, p. 281–305, 2012.

BIJALWAN, V.; SEMWAL, V. B.; MANDAL, T. K. Fusion of multi-sensor-based biomechanical gait analysis using vision and wearable sensor. **IEEE Sensors Journal**, Institute of Electrical and Electronics Engineers (IEEE), v. 21, n. 13, p. 14213–14220, jul. 2021. Disponível em: https://doi.org/10.1109/jsen.2021.3066473.

BONNAFFE, F.; JENNETTE, D.; ANDREWS, J. A method for acquiring and processing ground-based lidar data in difficult-to-access outcrops for use in three-dimensional, virtual-reality models. **Geosphere**, Geological Society of America, v. 3, n. 6, p. 501–510, 2007.

BOULCH, A. Convpoint: Continuous convolutions for point cloud processing. **Computers & Graphics**, Elsevier, v. 88, p. 24–34, 2020.

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

BRIDLE, J. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. **Advances in neural information processing systems**, v. 2, 1989.

CHANG, A. X.; FUNKHOUSER, T.; GUIBAS, L.; HANRAHAN, P.; HUANG, Q.; LI, Z.; SAVARESE, S.; SAVVA, M.; SONG, S.; SU, H.; XIAO, J.; YI, L.; YU, F. **ShapeNet: An Information-Rich 3D Model Repository**. [*S. l.*], 2015.

FAN, L.; XIONG, X.; WANG, F.; WANG, N.; ZHANG, Z. Rangedet: In defense of range view for lidar-based 3d object detection. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [*S. l.*: *s. n.*], 2021. p. 2918–2927.

FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FAZLALI, H.; XU, Y.; REN, Y.; LIU, B. A versatile multi-view framework for lidar-based 3d object detection with guidance from panoptic segmentation. **arXiv preprint arXiv:2203.02133**, 2022.

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)**. [*S. l.*], 2021. p. 1–5.

GEIGER, A.; LENZ, P.; STILLER, C.; URTASUN, R. Vision meets robotics: The kitti dataset. **International Journal of Robotics Research (IJRR)**, 2013.

GIONGO, M.; KOEHLER, H. S.; MACHADO, S. do A.; KIRCHNER, F. F.; MARCHETTI, M. Lidar: princípios e aplicações florestais. **Pesquisa Florestal Brasileira**, v. 30, n. 63, p. 231–231, 2010.

GIRDEN, E. R. **ANOVA: Repeated measures**. [*S. l.*]: Sage, 1992.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S. l.*]: 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.

HACKEL, T.; SAVINOV, N.; LADICKY, L.; WEGNER, J. D.; SCHINDLER, K.; POLLEFEYS, M. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In: **ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences**. [*S. l.*: *s. n.*], 2017. IV-1-W1, p. 91–98.

HAHNER, M.; SAKARIDIS, C.; BIJELIC, M.; HEIDE, F.; YU, F.; DAI, D.; GOOL, L. V. Lidar snowfall simulation for robust 3d object detection. **arXiv preprint arXiv:2203.15118**, 2022.

HAHNER, M.; SAKARIDIS, C.; DAI, D.; GOOL, L. V. Fog simulation on real lidar point clouds for 3d object detection in adverse weather. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [*S. l.*: *s. n.*], 2021. p. 15283–15292.

HAMDI, A.; GIANCOLA, S.; GHANEM, B. Mvtn: Multi-view transformation network for 3d shape recognition. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [*S. l.*: *s. n.*], 2021. p. 1–11.

HAYKIN, S. **Redes neurais: princípios e prática**. [*S. l.*]: Bookman Editora, 2001.

HECHT, J. Lidar for self-driving cars. **Optics and Photonics News**, Optical Society of America, v. 29, n. 1, p. 26–33, 2018.

HONG, F.; ZHOU, H.; ZHU, X.; LI, H.; LIU, Z. Lidar-based panoptic segmentation via dynamic shifting network. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2021. p. 13090–13099.

HU, J. S.; KUAI, T.; WASLANDER, S. L. Point density-aware voxels for lidar 3d object detection. **arXiv preprint arXiv:2203.05662**, 2022.

HU, Q.; YANG, B.; XIE, L.; ROSA, S.; GUO, Y.; WANG, Z.; TRIGONI, N.; MARKHAM, A. Randla-net: Efficient semantic segmentation of large-scale point clouds. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2020. p. 11108–11117.

HU, Z.; ZHEN, M.; BAI, X.; FU, H.; TAI, C. lan. JSENet: Joint semantic segmentation and edge detection network for 3d point clouds. In: **Computer Vision – ECCV 2020**. Springer International Publishing, 2020. p. 222–239. Disponível em: https://doi.org/10.1007/978-3-030-58565-5_14.

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

JABOYEDOFF, M.; OPPIKOFER, T.; ABELLÁN, A.; DERRON, M.-H.; LOYE, A.; METZGER, R.; PEDRAZZINI, A. Use of lidar in landslide investigations: a review. **Natural hazards**, Springer, v. 61, n. 1, p. 5–28, 2012.

KOH, J.; KIM, J.; YOO, J.; KIM, Y.; CHOI, J. W. Joint 3d object detection and tracking using spatio-temporal representation of camera image and lidar point clouds. **arXiv preprint arXiv:2112.07116**, 2021.

KUMAWAT, S.; RAMAN, S. **LP-3DCNN: Unveiling Local Phase in 3D Convolutional Neural Networks**. arXiv, 2019. Disponível em: https://arxiv.org/abs/1904.03498.

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**. [*S. l.*: *s. n.*], 2018. p. 4558–4567.

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, D.; LEE, J.; LEE, J.; LEE, H.; LEE, M.; WOO, S.; LEE, S. Regularization strategy for point cloud via rigidly mixed sample. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2021. p. 15900–15909.

LI, J.; CHEN, B. M.; LEE, G. H. **SO-Net: Self-Organizing Network for Point Cloud Analysis**. arXiv, 2018. Disponível em: https://arxiv.org/abs/1803.04249.

LI, Y.; YU, A. W.; MENG, T.; CAINE, B.; NGIAM, J.; PENG, D.; SHEN, J.; WU, B.; LU, Y.; ZHOU, D. *et al.* Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. **arXiv preprint arXiv:2203.08195**, 2022.

LI, Z.; WANG, F.; WANG, N. Lidar r-cnn: An efficient and universal 3d object detector. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2021. p. 7546–7555.

LIU, X. Airborne LiDAR for DEM generation: some critical issues. **Progress in Physical Geography: Earth and Environment**, SAGE Publications, v. 32, n. 1, p. 31–49, fev. 2008. Disponível em: https://doi.org/10.1177/0309133308089496.

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.

MA, C.; GUO, Y.; YANG, J.; AN, W. Learning multi-view representation with LSTM for 3-d shape recognition and retrieval. **IEEE Transactions on Multimedia**, Institute of Electrical and Electronics Engineers (IEEE), v. 21, n. 5, p. 1169–1182, maio 2019. Disponível em: https://doi.org/10.1109/tmm.2018.2875512.

MALAVAZI, F. B.; GUYONNEAU, R.; FASQUEL, J.-B.; LAGRANGE, S.; MERCIER, F. Lidar-only based navigation algorithm for an autonomous agricultural robot. **Computers and electronics in agriculture**, Elsevier, v. 154, p. 71–79, 2018.

MATURANA, D.; SCHERER, S. VoxNet: A 3d convolutional neural network for real-time object recognition. In: **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 2015. Disponível em: https://doi.org/10.1109/iros.2015.7353481.

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)**. [*S. l.*], 2019. p. 4213–4220.

MüLLER, B.; ILG, W.; GIESE, M. A.; LUDOLPH, N. Validation of enhanced kinect sensor based motion capturing for gait assessment. **PLOS ONE**, Public Library of Science (PLoS), v. 12, n. 4, p. e0175813, abr. 2017. Disponível em: https://doi.org/10.1371/journal.pone.0175813.

O'MALLEY, T.; BURSZTEIN, E.; LONG, J.; CHOLLET, F.; JIN, H.; INVERNIZZI, L. *et al.* **Keras Tuner**. 2019. https://github.com/keras-team/keras-tuner.

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)**. [*S. l.*], 2020. p. 61–67.

QI, C. R.; SU, H.; MO, K.; GUIBAS, L. J. **PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation**. 2017.

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

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.

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.

ROYNARD, X.; DESCHAUD, J.-E.; GOULETTE, F. Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. **The International Journal of Robotics Research**, SAGE Publications, v. 37, n. 6, p. 545–557, abr. 2018. Disponível em: https://doi.org/10.1177/0278364918767506.

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**. [*S. l.*], 2011. p. 1–4.

SASAKI, Y. *et al.* The truth of the f-measure. **Teach Tutor mater**, v. 1, n. 5, p. 1–5, 2007.

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

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information processing & management**, Elsevier, v. 45, n. 4, p. 427–437, 2009.

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.

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.; WANG, W.; CHAI, Y.; ELSAYED, G.; BEWLEY, A.; ZHANG, X.; SMINCHISESCU, C.; ANGUELOV, D. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2021. p. 5725–5734.

THOMAS, H.; QI, C. R.; DESCHAUD, J.-E.; MARCOTEGUI, B.; GOULETTE, F.; GUIBAS, L. J. Kpconv: Flexible and deformable convolution for point clouds. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [*S. l.*: *s. n.*], 2019. p. 6411–6420.

UNAL, O.; DAI, D.; GOOL, L. V. Scribble-supervised lidar semantic segmentation. **arXiv preprint arXiv:2203.08537**, 2022.

WANG, X.; FU, C.; LI, Z.; LAI, Y.; HE, J. Deepfusionmot: A 3d multi-object tracking framework based on camera-lidar fusion with deep association. **arXiv preprint arXiv:2202.12100**, 2022.

WEI, Y.; WEI, Z.; RAO, Y.; LI, J.; ZHOU, J.; LU, J. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. **arXiv preprint arXiv:2203.14956**, 2022.

WU, W.; QI, Z.; FUXIN, L. Pointconv: Deep convolutional networks on 3d point clouds. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2019. p. 9621–9630.

WU, Z.; SONG, S.; KHOSLA, A.; YU, F.; ZHANG, L.; TANG, X.; XIAO, J. 3d ShapeNets: A deep representation for volumetric shapes. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2015. Disponível em: https://doi.org/10.1109/cvpr.2015.7298801.

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

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. **arXiv preprint arXiv:1505.00853**, 2015.

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

XU, M.; ZHANG, J.; ZHOU, Z.; XU, M.; QI, X.; QIAO, Y. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. **arXiv preprint arXiv:2012.10921**, v. 2, 2021.

XU, S.; WAN, R.; YE, M.; ZOU, X.; CAO, T. Sparse cross-scale attention network for efficient lidar panoptic segmentation. **arXiv preprint arXiv:2201.05972**, 2022.

XU, Y.; FAN, T.; XU, M.; ZENG, L.; QIAO, Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [*S. l.*: *s. n.*], 2018. p. 87–102.

YANG, H.-D. Sign language recognition with the kinect sensor based on conditional random fields. **Sensors**, MDPI, v. 15, n. 1, p. 135–147, 2014.

YAVARTANOO, M.; HUNG, S.-H.; NESHATAVAR, R.; ZHANG, Y.; LEE, K. M. Polynet: Polynomial neural network for 3d shape recognition with polyshape representation. In: IEEE. **2021 International Conference on 3D Vision (3DV)**. [*S. l.*], 2021. p. 1014–1023.

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.

YU, X.; TANG, L.; RAO, Y.; HUANG, T.; ZHOU, J.; LU, J. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. **arXiv preprint arXiv:2111.14819**, 2021.

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

ZHANG, J.; CHEN, L.; OUYANG, B.; LIU, B.; ZHU, J.; CHEN, Y.; MENG, Y.; WU, D. Pointcutmix: Regularization strategy for point cloud classification. **arXiv preprint arXiv:2101.01461**, 2021.

ZHANG, R.; ZENG, Z.; GUO, Z.; GAO, X.; FU, K.; SHI, J. Dspoint: Dual-scale point cloud recognition with high-frequency fusion. **arXiv preprint arXiv:2111.10332**, 2021.

ZHANG, W. Lidar-based road and road-edge detection. In: IEEE. **2010 IEEE Intelligent Vehicles Symposium**. [*S. l.*], 2010. p. 845–848.

ZHANG, Y.; HU, Q.; XU, G.; MA, Y.; WAN, J.; GUO, Y. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. **arXiv preprint arXiv:2203.11139**, 2022.

ZHANG, Z.; GIRDHAR, R.; JOULIN, A.; MISRA, I. **Self-Supervised Pretraining of 3D Features on any Point-Cloud**. arXiv, 2021. Disponível em: https://arxiv.org/abs/2101.02691.

ZHAO, Y.; ZHANG, X.; HUANG, X. A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [*S. l.*: *s. n.*], 2021. p. 2464–2473.

ZHOU, H.; ZHU, X.; SONG, X.; MA, Y.; WANG, Z.; LI, H.; LIN, D. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. **arXiv preprint arXiv:2008.01550**, 2020.

ZHOU, Z.; ZHANG, Y.; FOROOSH, H. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2021. p. 13194–13203.