UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

CARLOS EDUARDO CASSIMIRO DA SILVA

PREDIÇÃO DE DESCRITORES GEOMÉTRICOS EM NUVENS DE PONTOS UTILIZANDO A ARQUITETURA POINTNET

FORTALEZA

2023

CARLOS EDUARDO CASSIMIRO DA SILVA

PREDIÇÃO DE DESCRITORES GEOMÉTRICOS EM NUVENS DE PONTOS UTILIZANDO A ARQUITETURA POINTNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. José Marques Soares.
Coorientador: Me. Artur Rodrigues Rocha Neto.

FORTALEZA

2023

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

C338p Silva, Carlos Eduardo Cassimiro da.
Predição de descritores geométricos em nuvens de pontos utilizando a arquitetura PointNet / Carlos Eduardo Cassimiro da Silva. – 2023.
64 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Prof. Dr. José Marques Soares.
Coorientação: Prof. Me. Artur Rodrigues Rocha Neto.

1. Descritores geométricos. 2. PointNet. 3. Nuvem de pontos. 4. Redes neurais. I. Título.
CDD 621.39

CARLOS EDUARDO CASSIMIRO DA SILVA

PREDIÇÃO DE DESCRITORES GEOMÉTRICOS EM NUVENS DE PONTOS UTILIZANDO A ARQUITETURA POINTNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. José Marques Soares (Orientador)
Universidade Federal do Ceará (UFC)

Me. Artur Rodrigues Rocha Neto (Coorientador)
Laboratório de Desenvolvimento e Inovação (LDI) -
Universidade Estadual do Ceará (UECE)

Prof. Dr. George André Pereira Thé
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada. E à Deus, por corroborar com todos esses em minha vida.

AGRADECIMENTOS

Aos meus pais por todo incentivo e apoio diário em todos os meus passos até aqui.

Ao Prof. Dr. José Marques Soares, pela excelente orientação e empatia no meu processo de conclusão de curso.

Ao Ms. Artur Rodrigues pelo tempo cedido para coorientação e pelas sempre oportunas contribuições.

Aos colegas de graduação pela convivência e companheirismo, em especial Erik Ray, Francisco David e Rafael Costa, dividindo as alegrias e reveses.

Aos professores que se dedicaram em contribuir verdadeiramente para formação.

À Universidade Federal do Ceará, pelos diversos programas de apoio à permanência e serviços aos estudantes.

"Até mais, e obrigado pelos peixes!"

(O Guia do Mochileiro das Galáxias)

RESUMO

Descritores geométricos são características ou representações matemáticas que capturam informações sobre a geometria de objetos ou cenários, sendo bastante utilizados na identificação de correspondências entre pontos de nuvens distintas no contexto de registro. Descritores geométricos geralmente são custosos para se calcular por conta da falta de ordenação na estrutura das nuvens, sendo necessário procurar iterativamente pelos vizinhos de cada ponto. Dessa maneira, convém buscar um meio de otimizar a determinação de descritores geométricos locais. Neste trabalho foram propostos cinco arquiteturas com três estratégias diferentes para regressão ponto-a-ponto baseadas na PointNet. Apesar dos modelos desenvolvidos não terem atingido o objetivo de criar uma função aproximadora para o cálculo de descritores geométricos, o processo de modelagem e testes renderam observações importantes acerca da utilização de redes neurais em nuvens de pontos.

Palavras-chave: Descritores Geométricos. PointNet. Nuvem de Pontos. Redes neurais.

ABSTRACT

Geometric descriptors are characteristics or mathematical representations that capture information about the geometry of objects or scenes and are widely used in matching points between different point clouds in the context of registration. Geometric descriptors are often computationally expensive to calculate due to the lack of ordering in the structure of point clouds, requiring iterative neighbor search for each point. Therefore, it is desirable to optimize the determination of local geometric descriptors. In this work, five architectures with three different strategies for point-to-point regression based on PointNet were proposed. Although the developed models did not achieve the goal of creating an approximating function for calculating geometric descriptors, the modeling and testing process yielded important insights regarding the use of neural networks in point clouds.

Keywords: Geometric Descriptors. PointNet. Point Cloud. Neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A demanda por informações tridimensionais representadas computacionalmente tem crescido nos últimos anos, impulsionando o desenvolvimento de novas ferramentas e métodos para a coleta de dados e informações espaciais (MEKURIA *et al.*, 2017). Existem várias maneiras de se representar um objeto tridimensional computacionalmente, cada uma com suas particularidades. Dentre elas, podemos destacar as nuvens de pontos, que oferecem: praticidade para aquisição, por conta da praticidade dos seus sensores (ZHANG, 2018); armazenamento, por conda sua representação simples; e transformação para outras representações geométricas, como malhas triangulares e cubóides (RUSU; COUSINS, 2011).

Nuvens de pontos são conjuntos de pontos em um espaço tridimensional que representam a geometria de um objeto ou cena através de sua superfície. Cada ponto na nuvem possui coordenadas tridimensionais, normalmente expressas como coordenadas cartesianas, podendo conter informações extras referente a sua relação com outros pontos.

Embora os sensores de nuvens de pontos ofereçam praticidade no manuseio, eles não conseguem capturar toda a superfície de um objeto de uma vez, gerando vários pedaços de um objeto ou ambiente (ZHANG, 2018). Dessa maneira, faz-se necessário alinharmos todas as amostras para reconstruir o objeto ou ambiente em questão, sendo esse processo denominado como registro.

Os métodos de registro geométrico podem realizar o registro entre dois conjuntos de pontos ou duas superfícies. Isso é feito encontrando a transformação geométrica ideal entre os dois conjuntos, minimizando o erro médio entre eles (MEN *et al.*, 2011). Essa transformação é obtida pela correspondência de pontos entre os dois conjuntos que, por sua vez, pode ser realizada baseando-se em certas características ou propriedades dos pontos (HAN *et al.*, 2020).

Para otimizar o procedimento de registro, podemos utilizar informações da própria nuvem de pontos através de filtragem ou segmentação (RUSU *et al.*, 2008). Informações locais dos pontos geralmente são custosos para se calcular por conta da falta de ordenação na estrutura das nuvens, sendo necessário procurar iterativamente pelos vizinhos de cada ponto.

Dessa maneira, convém buscar um meio de otimizar a determinação de descritores geométricos locais. O presente trabalho se insere nesse contexto objetivando utilizar redes neurais, que são aproximadores universais de funções matemáticas (SCHAUL *et al.*, 2015), para simular o cálculo de descritores geométricos e predizer seus valores ponto-a-ponto de maneira eficiente com uma resolução satisfatória.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados alguns dos principais elementos e conceitos que serão trabalhados e discutidos no decorrer da dissertação. Onde serão caracterizados: as nuvens de pontos, que são os objetos de estudo deste trabalho; o registro, que é o método para o qual se quer buscar otimização por meio da seleção de pontos mais representativos; os descritores geométricos, sendo as representações dos objetos que pretendemos predizer; a PointNet, que é a ferramenta adotada para o aprendizado da tarefa alvo; e os demais temas complementares a esses (registro, redes neurais, conjunto de dados e pré-processamento).

2.1 Nuvens de Pontos

As nuvens de pontos são conjuntos de pontos em um espaço tridimensional que representa a geometria de um objeto ou cena por meio de sua superfície. Cada ponto em uma nuvem de pontos têm coordenadas tridimensionais, geralmente expressas como coordenadas cartesianas (x, y, z), e pode estar associado a informações adicionais, como cores, normais ou alguma característica da superfície do objeto (LEVOY; WHITTED, 1985).

As nuvens de pontos podem ser adquiridas por meio de sensores de profundidade. Esses sensores registram a posição espacial de cada ponto em relação a um sistema de coordenadas de referência, permitindo a reconstrução tridimensional do objeto ou ambiente. São várias as formas de aquisição, podendo acontecer por sensores tempo de vôo, onde o princípio por trás desta tecnologia baseia-se na medição da distância aos pontos das superfícies através do tempo que a radiação emitida demora a chegar aos objetos e voltar (WEINMANN, 2016), ou baseado no princípio da luz estruturada, onde uma projeção de um padrão numa cena é mantida e o modo como esse padrão se deforma quando atinge as superfícies possibilita o cálculo da profundidade (SALVI *et al.*, 2004).

Nuvens de pontos são frequentemente utilizadas em diversas áreas, como visão computacional, robótica, realidade virtual e modelagem 3D, por servirem de ponte entre os objetos reais e soluções computacionais com a sua representação digital dos mesmos. Também podemos destacar a sua versatilidade para aquisição, armazenamento e conversão para outras representações geométricas, como malhas triangulares e cubóides *(voxels)* (RUSU; COUSINS, 2011). Entretanto, como na maioria das técnicas de reconstrução digital de objetos e cenários reais, os sensores geram várias amostras de partes diferentes de um objeto e ainda precisamos

lidar com o alinhamento delas para obter a nuvem de pontos completa.

2.2 Registro de Nuvens de Pontos

O registro de nuvens de pontos é o processo de estimação de uma transformação espacial que permite sobrepor duas ou mais nuvens de pontos para que elas compartilhem uma mesma orientação e, portanto, se conjuguem formando uma superfície única (SCHARSTEIN *et al.*, 2002). Dessa maneira, podemos, por exemplo, alinhar adequadamente nuvens de pontos parciais de um objeto ou cena, capturadas com algum sensor, e obter uma representação completa da superfície em questão. Existem vários algoritmos para realizar o alinhamento de nuvens de pontos, sendo um dos mais conhecidos o *Iterative Closest Point* (ICP).

No contexto do ICP, duas nuvens de pontos são manipuladas a fim de estimar a melhor transformação de alinhamento entre elas. Uma das nuvens é denominada nuvem de origem e a outra, nuvem alvo. A nuvem alvo se mantém estática no espaço e recebe esse nome por ser a superfície de objetivo a qual queremos aproximar os pontos da nuvem de origem. O primeiro passo do ICP é encontrar.

O primeiro passo do ICP envolve encontrar correspondências entre os pontos das nuvens de pontos de entrada. Isso é feito comparando cada ponto de uma nuvem com todos os pontos da outra nuvem em busca da melhor correspondência. O ICP busca encontrar uma transformação rígida (rotação e translação) que minimize a distância entre os pontos correspondentes das nuvens de pontos. Para fazer isso, é necessário iterativamente ajustar a transformação e recalcular as correspondências entre os pontos (ZHANG, 1994). Esse processo pode se tornar computacionalmente custoso, especialmente em nuvens com um significativo número de pontos na superfície, onde muitas iterações são necessárias para obter a convergência, pois requer um alto número de comparações.

O ICP possui ainda vulnerabilidades relacionadas à sensibilidade à inicialização, ambiguidade nas nuvens em alinhamento, ruídos e pontos isolados. Apesar das suas suscetibilidades a erros, ele se mantém eficaz para a maioria dos casos e, embora já existam variantes do algoritmo para melhorar sua eficiência, a sua otimização ainda é objeto de estudo (ZHANG *et al.*, 2021). Uma das estratégias de otimização que pode-se citar é se basear em informações prévias das nuvens de pontos para auxiliar o algoritmo do ICP, como descritores geométricos e características-chave (RUSU *et al.*, 2009).

2.3 Descritores Geométricos

Descritores geométricos são características ou representações matemáticas que capturam informações sobre a geometria de objetos ou cenários, sendo bastante utilizados na identificação de correspondências entre pontos de nuvens distintas no contexto de registro (ZHANG *et al.*, 2019). Existem diversos tipos de descritores geométricos, cada um com suas características e aplicabilidades específicas. Alguns exemplos incluem descritores baseados em histogramas de pontos, descritores baseados em formas e descritores baseados em características locais. Dentre os descritores baseados em características locais, podemos destacar os descritores geométricos da matriz de covariância por sua versatilidade em se desdobrar em outros que exprimem diferentes características (HACKEL *et al.*, 2016).

A matriz de covariância é uma matriz simétrica que descreve a dispersão dos pontos em torno de sua média. Para cada ponto em uma nuvem de pontos, uma matriz de covariância é calculada a partir das coordenadas dos pontos vizinhos ao ponto de referência. Os descritores geométricos são derivados dessa matriz de covariância para caracterizar as propriedades geométricas dos pontos em um determinado contexto local, podendo ser calculada a partir da Equação 2.1:

$$[Cov]_{ij} = \frac{1}{n}\sum_{i=1}^{n}(A_i - \bar{A_i})(A_j - \bar{A_j}), \tag{2.1}$$

em que n é o número de observações, j $\in$ [1,2,3] e $\bar{A_j}$ é a média de todas as observações na respectiva dimensão.

Com as matrizes de covariância calculadas, em seguida realizamos a decomposição da matriz de covariância na sua forma canônica para extrair os autovalores λ_1, λ_2 e λ_3 e os autovetores e_1, e_2 e e_3 para calcular os descritores geométricos.

Com os autovalores e autovetores definidos, podemos calcular, a exemplo: a omnivariância, que mede a distribuição volumétrica da superfície; a autoentropia, que descreve a ordem ou desordem dos pontos dentro de uma vizinhança; a planaridade, que descreve a suavidade da superfície; a anisotropia, que mede a mudança de padrões entre pontos em termos de sua direção (AGAPAKI; NAHANGI, 2020). Neste trabalho, o descritor que será alvo da predição é a omnivariância, que pode ser calculada a partir da Equação 2.2, por conta do seu carácter não-linear, onde poderemos avaliar a capacidade da rede de simular tal função.

$$O = \sqrt[3]{\lambda_1 * \lambda_2 * \lambda_3} \tag{2.2}$$

Apesar dos descritores geométricos serem utilizados como uma alternativa de otimização para algoritmos de correspondência como o ICP, eles também podem ser custosos computacionalmente, pois, para cada ponto, vários cálculos são realizados, comprometendo o objetivo de serem utilizados para melhorar a eficiência do mesmo.

Nessa perspectiva, podemos utilizar um aproximador de funções como uma Rede Neural Artificial (*Artificial Neural Network* – ANN) para obter as informações dos descritores. Para tarefas mais complexas e não lineares, onde padrões são difíceis de descrever com funções analíticas, as redes neurais podem, em alguns contextos, oferecer maior desempenho por se aproximar do resultado esperado de uma função, mas sem calculá-la passo a passo (HAN *et al.*, 2016).

2.4 Redes Neurais

Uma rede neural é um modelo computacional inspirado no funcionamento do cérebro humano que consiste em um conjunto interconectado de unidades de processamento chamadas de neurônios artificiais. Os neurônios artificiais realizam operações matemáticas em dados de entrada para produzir uma saída, onde cada um deles possuem diferentes pesos em suas funções, produzindo diferentes resultados com uma mesma operação (RUSSELL; NORVIG, 1995). Dessa maneira, definindo adequadamente a quantidade, a forma como eles se conectam, suas operações matemáticas e seus pesos, podemos fazer com que redes neurais realizem tarefas de variadas naturezas (RUSSELL; NORVIG, 1995). Para facilitar o seu estudo, pode-se dividi-las em sua arquitetura e funções de treinamento.

2.4.1 Arquitetura

A estrutura de uma rede neural é composta por camadas de neurônios, onde sua organização mais básica possui três tipos principais: camada de entrada, camadas ocultas e camada de saída. A camada de entrada recebe os dados de entrada, em seguida as camadas ocultas processam e/ou extraem características dos dados, e por seguinte a camada de saída produz o resultado final, o resultado da tarefa que está sendo realizada pela rede.

Cada conexão entre os neurônios têm um peso associado, que representa a importância daquela conexão para a rede. Além disso, cada neurônio tem um valor de viés que ajusta o valor de saída do neurônio. Os pesos e vieses são ajustados durante o treinamento da rede neural

para otimizar o desempenho do modelo.

Cada neurônio aplica uma função de ativação ao somatório ponderado das entradas que recebe, juntamente com o viés (GOODFELLOW *et al.*, 2016). A função de ativação introduz não-linearidade na rede e permite que ela aprenda relações complexas nos dados. Exemplos de funções de ativação comumente usadas incluem a função sigmoide, função unidade linear retificada (*Rectified Linear Unit* - ReLU) e função tangente hiperbólica (NIELSEN, 2015).

2.4.2 Funções do Treinamento

A propagação direta é o processo em que os dados de entrada são transmitidos através da rede, camada por camada, começando pela camada de entrada e passando pelas camadas ocultas até a camada de saída (GOODFELLOW *et al.*, 2016). Cada neurônio calcula a soma ponderada das suas entradas, aplica a função de ativação e passa o resultado para os neurônios da próxima camada.

A função de perda é utilizada para estimar o quão bem o modelo está realizando a tarefa desejada, medindo a diferença entre os valores calculados pela rede e aqueles esperado durante a fase de treinamento (GOODFELLOW *et al.*, 2016). Ela compara a saída produzida pela rede com o valor verdadeiro ou esperado e gera um valor que representa o erro ou a discrepância entre a saída da rede e o valor desejado.

O objetivo do treinamento de uma rede neural é ajustar os pesos e vieses para minimizar o erro calculado pela função de perda. A retropropagação do erro é um algoritmo que calcula o gradiente do erro em relação aos pesos e vieses em toda a rede, camada por camada, a partir da camada de saída até a camada de entrada (NIELSEN, 2015). Esses gradientes são usados para atualizar os pesos e vieses por meio de um algoritmo de otimização, como o gradiente descendente, de forma a reduzir o erro progressivamente (NIELSEN, 2015).

O processo de retropropagação e atualização dos pesos e vieses é repetido várias vezes até que a rede neural alcance uma performance satisfatória na tarefa em questão, exigindo várias amostras de dados para a realização desse processo (NIELSEN, 2015). Durante o treinamento, os pesos e vieses são ajustados gradualmente, permitindo que a rede aprenda a reconhecer padrões nos dados e melhore seu desempenho. Em tarefas mais difíceis, geralmente o treinamento com o mesmo conjunto de dados é repetido em iterações, também chamadas de épocas.

Entre as camadas ocultas de uma rede neural, podem ser executadas normalizações

em lote para tentar acelerar o treinamento e melhorar o desempenho dos modelos (GOODFELLOW *et al.*, 2016). Durante o treinamento de redes neurais, especialmente em camadas profundas, os valores de ativação podem variar significativamente à medida que os dados passam pela rede. Isso pode levar a problemas como a degradação do gradiente e a desaceleração do treinamento (GOODFELLOW *et al.*, 2016). A normalização em lote mitiga esses problemas normalizando as ativações de cada camada através de uma normalização estatística (GOODFELLOW *et al.*, 2016).

Apesar de conseguirmos enquadrar a maioria das tarefas realizadas por redes neurais em classificação, regressão e síntese, existem várias arquiteturas diferentes para lidar com tarefas mais difíceis, que exigem soluções mais específicas, e os diferentes tipos de dados manipulados. Utilizar redes neurais para lidar com nuvem de pontos diretamente não é uma tarefa trivial, por conta da estrutura desordenada dos pontos e tamanhos diversos das nuvens. Entretanto, em 2017, Charles et al. (2017) criou a PointNet para tarefas de classificação e segmentação, abrindo precedentes para utilização de redes neurais para outras tarefas.

2.5 PointNet

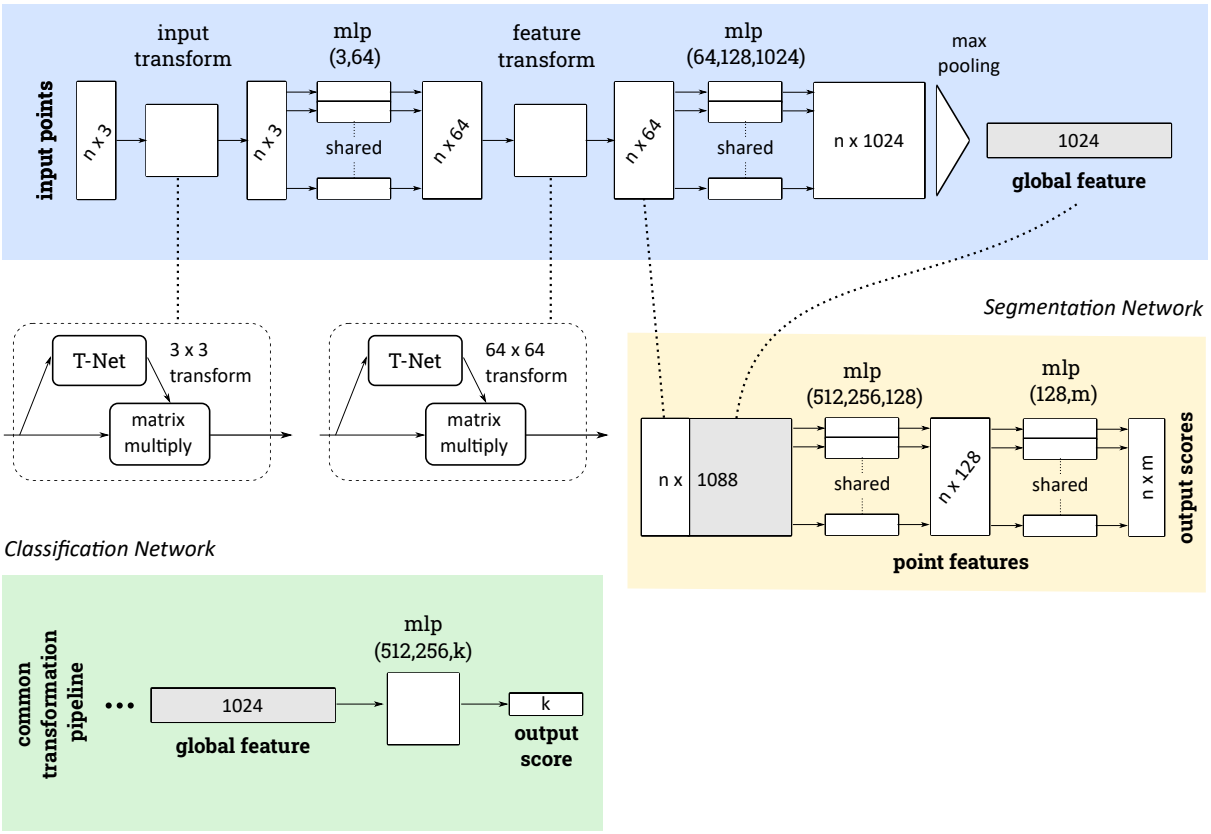
A PointNet é uma arquitetura de rede neural desenvolvida para processar diretamente nuvens de pontos, sem a necessidade de uma etapa prévia de conversão para representações volumétricas ou de malha, eliminando o tempo de pré-processamento desses casos. A principal ideia por trás da PointNet é tratar inicialmente cada ponto da nuvem como uma unidade de processamento independente e em seguida condensá-los em um vetor de atributos que representa a nuvem.

Podemos destacar duas principais operações na PointNet: a aplicação de transformações pontuais, para ajustar os pontos e aumentar a dimensionalidade dos pontos; e da agregação global de informações *(global features)*, para obter uma representação resumida da nuvem de pontos (CHARLES *et al.*, 2017). Dessa maneira, as *global features* são utilizadas para tarefas de classificação e a combinação deles com os pontos agora representados em alta dimensão são usados para tarefas de segmentação por partes.

2.5.1 Estrutura Básica

A PointNet possui duas arquiteturas: uma para tarefas de classificação e outra para segmentação, mas ambas são redes neurais que desdobram de uma estrutura básica comum. As etapas dessa estrutura consistem em: transformação de entrada *(input transform)*, aumento de dimensionalidade para 64, transformação de atributo *(feature transform)*, aumento de dimensionalidade para 1024 e agregação global (CHARLES *et al.*, 2017).

Figura 1 – Arquitetura da PointNet



Fonte: elaborada pelo autor.
As redes de classificação e regressão utilizam a mesma estrutura base de transformação. Cada ponto da nuvem é submetido a essa estrutura, onde os *n* pontos passam pelas etapas de *input transform*, *feature transform* e *max pooling* para gerar a *global feature*. "mlp" significa *multi-layer perceptron*, os números entre colchetes são tamanhos de camada. Normalização em lote é usado para todas as camadas com ReLU. As camadas de *dropout* são usadas para o último mlp na rede de classificação e segmentação.

A *input transform* é realizada por uma sub-rede chamada T-Net que é treinada junto com a PointNet para realizar uma transformação pontual. Uma transformação pontual em nuvens de pontos consiste na aplicação de uma função a cada ponto separadamente, sem levar em consideração os pontos vizinhos ou a estrutura global da nuvem. A função da T-Net que é aprendida e adaptada para ajustar a nuvem de pontos para a tarefa em questão que a PointNet

está sendo treinada (CHARLES *et al.*, 2017).

Em seguida, os pontos são submetidos a uma rede neural que realiza uma transformação pontual para aumentar a dimensão dos pontos de 3 para 64. Essa rede neural possui uma arquitetura simples com apenas 3 camadas totalmente conectadas: a camada de entrada, que recebe x, y, z de cada ponto; uma camada oculta de tamanho 64, que realiza uma convolução de uma dimensão e normalização em lote; e uma camada de saída de também tamanho 64 com função de ativação ReLU (CHARLES *et al.*, 2017).

As etapas de *feature transform* e aumento de dimensionalidade para 1024 realizam, respectivamente, as mesmas funções das duas etapas anteriores com seus devidos ajustes. A *feature transform* utiliza outra T-Net para ajustar os pontos (agora em dimensão 64). Em seguida, os pontos são submetidos a outra rede neural totalmente conectada, mas dessa vez com uma arquitetura: camada de entrada de tamanho 64, uma camada oculta de 64, outra camada oculta 128 e uma camada de saída de 1024, com as funções correspondentes à arquitetura anterior respectiva (CHARLES *et al.*, 2017).

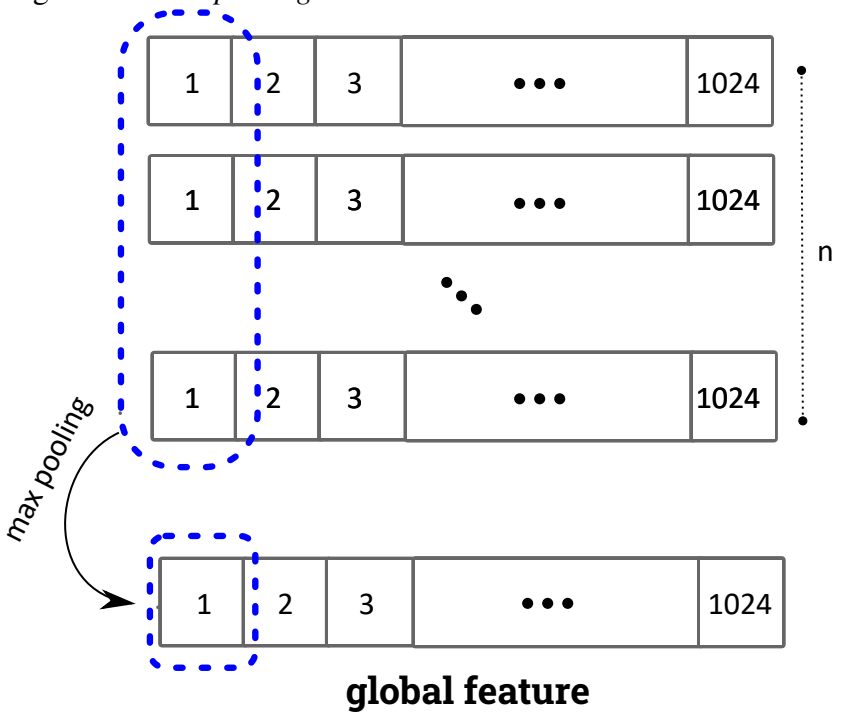
Após essas quatro etapas, é realizada uma operação de agregador de máximo *(max pooling)* para resumir os pontos (agora em dimensão 1024) em um vetor de atributos de tamanho 1024, chamado *global feature*. A agregação global é uma operação utilizada em redes neurais para reduzir a dimensionalidade de vetores ou matrizes de atributos gerados por alguma camada convolucional, combinando as informações das mesmas em um vetor ou matriz de tamanho menor através de alguma operação. Se a operação realizada na agregação global for assimétrica, como no caso do *max pooling*, que extrai o maior valor do espaço de seleção, ela garante ser invariante a ordenação, pois independente da ordem o máximo vai ser o mesmo (CHARLES *et al.*, 2017). O *max pooling* é aplicado para sobre cada i-ésima posição, onde $i : [1, 1024]$, nos n pontos da nuvem. Dessa maneira, a PointNet garante aceitar qualquer tamanho da nuvem, pois em qualquer tamanho o *max pooling* vai reduzi-la para o tamanho 1024 da *global feature*.

2.5.2 T-Net

A T-Net, ou Transform-Net, é uma rede neural totalmente conectada que atua como um módulo de transformação, permitindo que a rede PointNet aprenda a alinhar e orientar as nuvens de pontos de entrada de forma adequada antes de prosseguir com as etapas de processamento subsequentes.

A função da T-Net é permitir que a rede PointNet seja robusta a transformações

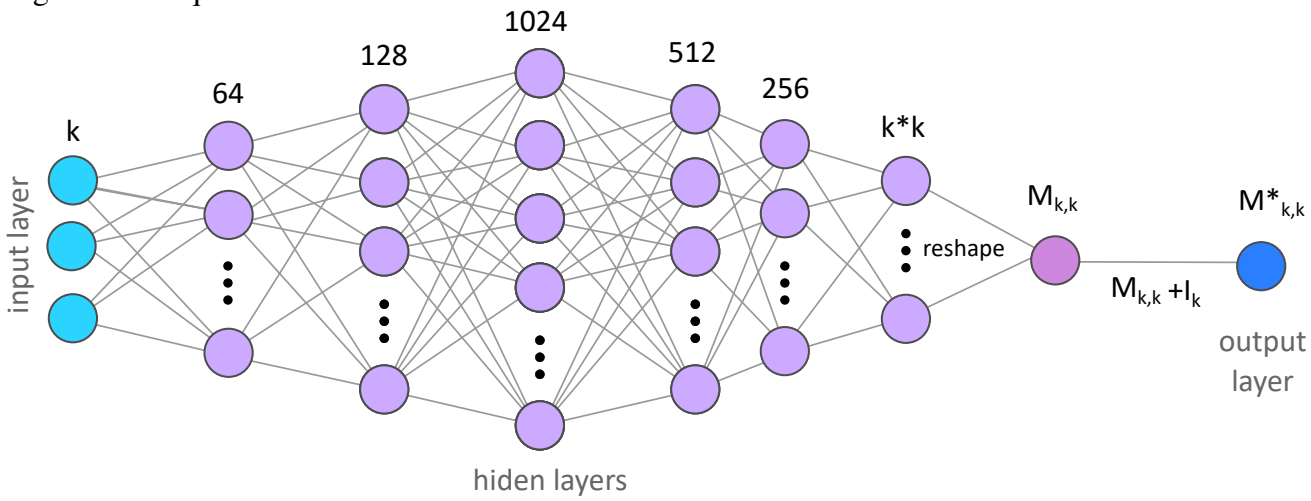
Figura 2 – *Max pooling*



Fonte: elaborada pelo autor.
O agrupamento de máxima é feito considerando cada i-ésima posição, onde i:[1,2024], dos *n* pontos da nuvem em dimensão 1024, resultando em um único vetor de tamanho 1024 que é a *global feature*.

espaciais, garantindo que a representação da nuvem de pontos seja robusta a rotação e invariante a escala e translação. A T-Net, portanto, é o elemento da PointNet que proporciona a arquitetura em lidar com nuvens de pontos em diferentes orientações e posições, tornando-a mais flexível e aplicável em várias tarefas de naturezas diferentes (CHARLES *et al.*, 2017).

Figura 3 – Arquitetura da T-Net



Fonte: elaborada pelo autor.
Os números nas colunas representam o número de neurônios na respectiva camada oculta da rede. A rede recebe os pontos em dimensão k, aumenta sua dimensão até 1024 com convoluções e depois diminui para a dimensão para $k * k$ através de transformações lineares. Normalização em lote é usado para todas as camadas com ReLU. Em seguida, o vetor $k * k$ é transposto em uma matriz $M_{k,k}$, onde esta é somada com a sua matriz identidade.

A arquitetura da T-Net se assemelha com a da própria PointNet, seguindo a estratégia de: aumentar a dimensionalidade dos pontos com redes neurais simples totalmente conectadas, mas sem a etapa de ajuste dos pontos, que é a própria T-Net; também realiza o *max pooling* e em seguida reduz a dimensionalidade com transformações lineares.

A arquitetura da T-Net consiste nas respectivas camadas em sequência: 3 redes neurais simples totalmente conectadas, *max pooling*, 3 camadas de transformações lineares, rearranjo de vetor para matriz e operação de soma com sua respectiva matriz identidade. A arquitetura da T-Net possui um parâmetro k que define o tamanho da camada de entrada e saída, dessa maneira, ela é usada em duas etapas diferentes da arquitetura: a primeira como estimadora da transformação dos pontos (x, y, z); na segunda, como igual normalizadora, mas agora de um conjunto de dimensão superior (atributos locais, explicados com mais detalhes a seguir).

As 3 primeiras camadas da T-Net, que são outras redes neurais simples, seguem a mesma arquitetura mudando apenas as dimensões das camadas. A arquitetura delas consiste em apenas 3 camadas totalmente conectadas: a camada de entrada, que recebe os pontos em dimensão 3, 64 ou 128 respectivamente para cada rede; uma camada oculta de tamanho 64, 128 ou 1024 que realiza uma convolução de uma dimensão e normalização em lote; e uma camada de saída com os mesmos tamanhos da respectiva camada anterior (64, 128 e 1024) com função de ativação ReLU.

Realizadas as 3 primeiras camadas, um *max pooling* é feito nos pontos (agora em dimensão 1024) seguindo a mesma execução da operação feita na PointNet.

Em seguida, as 3 camadas de redução de dimensionalidade por transformação linear são executadas, fazendo reduções do tamanho 1024 (tamanho depois do *max pooling*) para 512, 256 e k^2 respectivamente em cada camada, onde as duas primeiras realizam normalização em lote. Por fim, o vetor de tamanho k^2 é rearranjado para uma matriz $M_{k,k}$ para então ser somado com uma matriz identidade, sendo a matriz resultante a saída da T-Net.

A T-Net é treinada em conjunto com a rede PointNet para aprender automaticamente as transformações espaciais necessárias para melhorar o desempenho da rede. A matriz gerada pela T-Net realinha e orienta os pontos da nuvem de pontos de acordo com as transformações oportunas aprendidas, dando a rede robustez a transformações afim (CHARLES *et al.*, 2017). Durante o treinamento, os parâmetros da T-Net são ajustados para maximizar o desempenho da rede PointNet na tarefa em questão, como classificação ou segmentação.

2.5.3 Tarefas

Dada a estrutura básica da PointNet realizada, temos o conjunto de pontos da nuvem em alta dimensionalidade (atualmente no tamanho 1024) e temos o vetor de atributos *global feature* com informações resumidas sobre a mesma. Charles et al. (2017) argumentam que, a operação de *max pooling*, ao extrair vários pontos de máximo dos pontos em alta dimensão, podemos destacar os pontos-chave da nuvem. Ao destacar vários pontos-chave, cria-se um vetor que representa uma espécie de esqueleto da nuvem, servindo assim para conseguirmos discriminá-la realizando alguma classificação.

Charles et al. (2017) também argumentam que, ao aumentarmos a dimensionalidade dos pontos e aplicando-os as mesmas transformações, os pontos espacialmente próximos apresentarão informações semelhantes/próximas em seu vetor de alta dimensão. Dessa maneira, temos pontos que, mesmo tendo sido processados isoladamente, guardam informações locais da nuvem. Ao juntarmos essas informações com as informações globais da *global feature* podemos realizar tarefas que essencialmente necessitam de informações locais, como segmentação.

A arquitetura da PointNet para classificação segue um esquema básico de classificação com redes neurais, sendo parecida com a etapa de redução de dimensionalidade da T-Net, com incrementos para a referida tarefa. A arquitetura consiste em: 3 camadas de redução de dimensionalidade por transformação linear, fazendo reduções do tamanho 1024 da *global feature* para 512, 256 e k, respectivamente em cada camada, sendo k o número de classes, onde a primeira e a terceira realizam normalização em lote e a segunda *dropout*; encerrando com uma camada de *logsoftmax* para indicar a probabilidade de cada classe.

Na arquitetura da PointNet para segmentação, para unir as informações globais às locais, são concatenados aos pontos da etapa de *feature transform* (em dimensão 64) com o vetor de *global features*, resultando em um vetor de tamanho 1088. A sua estratégia segue um esquema de redução de dimensionalidade para cada ponto até o tamanho k de classes da segmentação. A arquitetura consiste em: 4 camadas para redução de dimensionalidade por convolução, onde as três primeiras realizam normalização em lote e ReLU, e uma última camada de *logsoftmax*.

Graças a estratégia de unir as informações locais e globais da PointNet para realizarmos segmentação em cada ponto, também podemos tentar estender suas funções para tarefas de regressão realizando as devidas modificações. Para isso, uma das modificações cruciais que devemos fazer é substituir a função de perda de probabilidade de log negativo ou interseção média sobre união, utilizadas em segmentação por funções de perda para tarefas de regressão.

2.6 Funções de perda para regressão

Uma função de perda é uma medida usada para quantificar a discrepância entre as previsões de um modelo de aprendizado de máquina e os valores reais dos dados de treinamento. Ela é fundamental para treinamento de um modelo, pois fornece informações sobre o quão bem o modelo está performando.

A função de perda recebe como entrada as previsões do modelo e os rótulos verdadeiros dos dados de treinamento e retorna um valor escalar que representa a magnitude do erro do modelo. O objetivo é minimizar esse valor de perda durante o treinamento para ajustar os pesos e vieses da rede, através da retropropagação do erro, ajustando a acuracidade do modelo em realizar previsões mais próximas do real.

Como funções de perda comumente utilizadas para tarefas de regressão, podemos citar: *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE) e *Mean Absolute Error* (MAE). Essas métricas são baseadas no erro absoluto entre o valor predito pela rede e o valor real que esperamos atingir, cada uma acentuando alguma característica da relação entre a diferença desses valores com as suas fórmulas.

Essas funções de perda podem ser utilizadas tanto para tarefas de regressão que se almeja estimar apenas um valor (apenas 1 saída) bem como estimar múltiplos valores. Dependendo da natureza da tarefa de regressão multissaída e também quantidade valores para predição, as métricas baseadas no erro absoluto podem não refletir adequadamente a precisão do modelo preditivo. Para contornar isso, algumas métricas consideram e comparam as distribuições dos valores preditos e reais, como a função *Gaussian Negative Log Likelihood Loss* (GNLLL).

2.6.1 Mean Squared Error

O MSE é uma medida de erro quadrático médio, calculado como a média das diferenças quadráticas entre as previsões e os valores reais, representado na Equação 2.3:

$$MSE = (\frac{1}{n}) * \sum (y_i - \bar{y_i})^2 \tag{2.3}$$

onde n é o número de amostras ou instâncias, y_i é o valor real observado, $\bar{y_i}$ é a previsão do modelo.

O MSE penaliza erros maiores de forma quadrática, o que significa que erros maiores têm um impacto mais significativo na função de perda. Isso pode ser útil quando desejamos dar mais importância a previsões muito erradas.

2.6.2 Root Mean Squared Error

O RMSE é uma versão do MSE em que a raiz quadrada é aplicada ao resultado para fornecer uma medida do erro médio, representado na Equação 2.4. É útil quando queremos ter uma noção mais intuitiva do erro, pois está na mesma escala que os valores de destino.

$$RMSE = \sqrt{MSE} \tag{2.4}$$

O RMSE é calculado da mesma maneira que o MSE, mas sua unidade é a mesma que a variável de destino. Isso facilita a interpretação e comparação do erro com os valores reais.

2.6.3 Mean Absolute Error

O MAE é uma medida de erro médio absoluto, calculado como a média das diferenças absolutas entre as previsões e os valores reais, representado na Equação 2.5:

$$MAE = (\frac{1}{n}) * \sum |y_i - \bar{y_i}| \tag{2.5}$$

2.6.4 Gaussian Negative Log Likelihood

A função de perda GNLLL é uma métrica que mede a diferença entre as previsões de um modelo e as distribuições gaussianas dos valores reais (MURPHY, 2012). É frequentemente usada em problemas de regressão quando se assume que os erros seguem uma distribuição gaussiana (MURPHY, 2012). Em vez de simplesmente prever um único valor, a GNLLL também leva em consideração a incerteza das previsões. O objetivo é não apenas minimizar a diferença entre as previsões e os valores reais, mas também capturar a incerteza associada a essas previsões (MURPHY, 2012).

A função de perda é baseada no conceito de verossimilhança, que mede a probabilidade de observar os valores reais dadas as previsões do modelo e uma distribuição gaussiana, representada na Equação 2.6. A verossimilhança é maximizada quando as previsões se aproximam dos valores reais de acordo com a distribuição gaussiana assumida (MURPHY, 2012).

$$GaussianNLLLoss = (\frac{1}{2}) * log(2\pi\sigma^2) + (\frac{1}{2\sigma^2}) * (y - \mu)^2 \tag{2.6}$$

onde π é o número pi, σ é o desvio padrão da distribuição gaussiana, μ é a média da distribuição gaussiana e y é o valor real observado.

2.6.5 Diferença entre as métricas

O MSE e o RMSE são mais adequados quando queremos penalizar erros maiores de forma mais significativa, enquanto o MAE é preferível quando queremos tratar todos os erros com igual importância. Assim como o MSE, o MAE mede a magnitude do erro. No entanto, em vez de penalizar os erros quadrados, o MAE trata todos os erros com igual importância, pois usa o valor absoluto. O MAE é menos sensível a diferenças discrepantes do que o MSE, uma vez que não amplifica os erros grandes.

Já a GNLLL é uma medida útil quando se assume que os erros seguem uma distribuição gaussiana. No entanto, é importante ressaltar que a escolha dessa função de perda depende do problema específico e das suposições feitas sobre a distribuição dos erros, não sendo em qualquer problema de regressão multissaída adequada para o uso.

A escolha entre MSE, RMSE, MAE e GNLLL depende do contexto da tarefa e de preferências do modelador para inferir sobre desempenho e as capacidades da rede. Em problemas pouco explorados, como regressão ponto-a-ponto sob nuvens de pontos, há poucas informações sobre o desempenho das funções de perca mais comuns nessas tarefas, tornando oportuno a realização de testes. Testes para escolha de uma função de perda foram desenvolvidos na Sessão 4.1 e seu uso discutido na Sessão 6.1.

2.6.6 Conjunto de dados

Os conjuntos de dados são coleções estruturadas de informações usadas para alimentar as redes neurais durante a fase de treinamento, bem como outros algoritmos de aprendizado de máquina. A rede neural aprende a reconhecer padrões e a fazer previsões com base nos exemplos apresentados nos dados.

O desempenho de uma rede neural depende da sua capacidade de generalizar os padrões aprendidos durante o treinamento para novos dados. Por isso, um conjunto de dados abrangente e representativo ajuda a melhorar a generalização do modelo, permitindo que ele faça previsões precisas em diferentes outros cenários não contemplados durante a fase de treinamento.

Coletar e catalogar amostras reais pode ser um trabalho oneroso e custoso financeiramente, além de incorrer em questões éticas em função dos dados coletados. Como alternativa, alguns conjuntos de dados são criados artificialmente para possibilitar o estudo e desenvolvimento de soluções com aprendizado de máquina. Por conta disso, algumas instituições de

pesquisa disponibilizam gratuitamente alguns de seus conjuntos de dados utilizados em seus trabalhos. Assim, alguns conjuntos de dados se tornam referência em algumas áreas de pesquisa, sendo utilizados frequentemente. Para objetos tridimensionais, podemos citar como conjuntos de dados de referência o ModelNet40 e o ShapeNet.

2.6.7 ModelNet40

O ModelNet40 é um conjunto de dados criado pela universidade de Princeton que é amplamente utilizado para pesquisas com objetos 3D. O ModelNet40 possui mais de 12 mil modelos 3D em formato de malha. Ele consiste em uma coleção de 40 categorias, incluindo itens comuns como cadeiras, mesas, carros e camas, que foram modelados computacionalmente. Ele também possui um conjunto com apenas 10 categorias de objetos mais simples com quase 5 mil amostras, a ModelNet10, que será utilizado nos testes do presente trabalho (WU *et al.*, 2015).

2.6.8 ShapeNet

O ShapeNet é outro conjunto de dados amplamente utilizado para pesquisas com objetos 3D, tendo sido criado como um esforço colaborativo pela universidade de *Princeton, Stanford e Toyota Technological Institute at Chicago*. O ShapeNet é um conjunto de dados bem grande com mais de 300 milhões de modelos sintéticos, possuindo vários subconjuntos com diferentes tipos de rótulos para uma variedade de tarefas. Um desses subconjuntos é o ShapeNet Part, com mais de 31 mil modelos 3D formato de malhas, categorizados em 16 classes e possuindo rótulos para segmentação por partes dos objetos. Existe ainda outra versão da ShapeNet Part para nuvens de pontos com quase 17 mil modelos já convertidos em nuvens de pontos com tamanho em cerca de 2500, com cada ponto da nuvem devidamente rotulado (CHANG *et al.*, 2015).

2.7 Pré-processamento

O pré-processamento de dados é uma etapa essencial na análise de dados e na construção de modelos de aprendizado de máquina. Essa etapa consiste em um conjunto de técnicas utilizadas para limpar, transformar e organizar os dados brutos, tornando-os adequados para análise e treinamento dos modelos. No caso dos dados do ModelNet10 e ShapeNet Part, não precisamos nos preocupar com limpezas pois os objetos 3D foram modelos computacionalmente e

não escaneados, mas ainda temos alguns procedimentos para realizar. Os objetos da ModelNet10 contém malhas representadas por vértices e faces triangulares, então precisamos fazer uma conversão para nuvem de pontos. A PointNet aceita nuvem de pontos de tamanho diferentes, mas para treinarmos vários dados ao mesmo tempo em *Graphics Processing Unit* (GPU), precisamos agrupá-los, e para manter propriedades de matrizes ao juntarmos os dados em lotes, precisamos que as nuvens tenham o mesmo tamanho. Para garantir maior equidade na comparação entre o modelos treinados com conjuntos de dados diferentes, os objetos também tiveram a sua escala normalizada.

2.7.1 Conversão de malhas para nuvens de pontos

No pré-processamento da ModelNet10, realizamos tanto a conversão de vértices e faces da malha para nuvem de pontos quanto a reamostragem desses pontos para o tamanho desejado no mesmo algoritmo:

1. As áreas dos triângulos dos vértices são calculadas e alocados em um vetor com mesma ordenação das faces;
2. Um vetor com o mesmo tamanho da amostragem dos pontos escolhida é criado;
3. O vetor criado é preenchido aleatoriamente com as faces usando como peso probabilístico a área dos triângulos calculados anteriormente. Dessa maneira, as faces que vão se repetir no sorteio são as que possuem as maiores áreas;
4. As coordenadas baricêntricas dos triângulos do vetor de faces aleatórias são calculadas para estimar os pontos da nuvem. Durante o cálculo, é utilizado uma interpolação linear com pesos aleatórios de forma a estimar um ponto sempre dentro da amostragem do seu respectivo triângulo. Dessa maneira, os pontos estimados de triângulos repetidos dificilmente vão ter a mesma posição, se distribuindo nessa superfície de maior área e melhorando a amostragem.

2.7.2 Normalização em Esfera Unitária

Dada a conversão e a reamostragem das nuvens de pontos realizadas, convêm normalizarmos a escala dos pontos. Para isso, podemos utilizar a normalização em esfera unitária com o algoritmo:

1. A média dos pontos é calculada para se encontrar o ponto de origem da nuvem;
2. Subtrai-se em cada ponto a média calculada para centralizar a nuvem de pontos em torno

da origem;

3. O maior valor da norma euclidiana (magnitude) entre todos os pontos é calculado;
4. Divide-se os pontos pelo maior valor da norma euclidiana calculado. Isso garante que todos os pontos tenham uma magnitude máxima de 1. Dessa maneira temos os pontos centralizados na origem e com escala normalizada.

2.8 Considerações finais

Neste capitulo foi apresentado o referencial teórico que serviu de base para o desenvolvimento do presente trabalho. Foi apresentado o conceito de descritores geométricos e de omnivariância, que tentaremos predizer. Foram introduzidas os principais componentes e etapas das redes neurais, que serão a base dos métodos do trabalho. Foram fundamentos a arquitetura e principais conceitos evolvendo a PointNet, que serão utilizados para as contribuições do projeto. Também foram expostos os conjuntos de dados e processamento realizado neles, sendo estes utilizados para o treinamento dos modelos propostos. Dado os elementos supracitados discutidos, no próximo capítulo serão desenvolvidos testes para nortear algumas decisões sobre o treinamento dos modelos da PointNet para regressão.

3 EXPERIMENTOS PARA CLASSIFICAÇÃO

Antes de iniciar os experimentos para predição de descritores de fato, primeiramente foram realizados alguns experimentos com a PointNet para classificação, com o objetivo de verificar seus resultados e nortear os experimentos para predição dos descritores. Os testes consistiram em examinar o efeito do uso de diferentes tamanhos para nuvens de pontos na *global feature* e na acurária da classificação. Um último teste foi feito mudando a arquitetura da rede para verificar o impacto no seu tempo de execução treinamento e inferência.

3.1 Ambiente de Testes

Os experimentos foram realizados em notebook Lenovo Ideapad L340, com as configurações:

- **CPU:** Intel CoreTM i5-9300HF 2.40GHz
- **RAM:** 8GB
- **GPU:** Nvidia GeForce GTX 1050 3GB
- **HD:** SSD NVME M.2 1T
- **SO:** Windows 10 Home

Ambiente de execução dos testes:

- **Liguagem:** Python 3.10.6
- **Ambiente de execução:** Jupyter-Lab 3.5.3
- **Navegador:** Edge 114.0.1823.58
- **Driver GPU:** CUDA 11.6

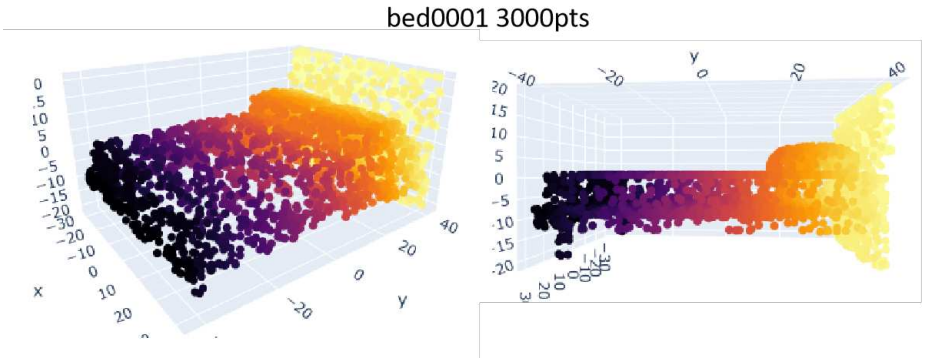
3.2 Global Feature para nuvens de diferentes tamanhos

Nesse primeiro teste, foram examinados os efeitos na definição das *global features* para nuvens de diferentes tamanhos em redes PointNet treinadas com diferentes tamanhos de nuvens. As redes PointNet foram treinadas com a ModelNet10 com os mesmos dados de treinamento, mudando apenas a amostragem das nuvens usadas em dois subconjuntos: um com nuvens com 1024 pontos e outro com nuvens com 600 pontos.

3.2.1 Global Feature para treinamento com nuvens de 1024 pontos

O objeto para inspeção foi selecionado arbitrariamente, sendo o primeiro objeto da classe cama do ModelNet10, a bed0001, que podemos ver na Figura 4.

Figura 4 – Nuvem de pontos do objeto bed0001 com amostragem de 3000 pontos

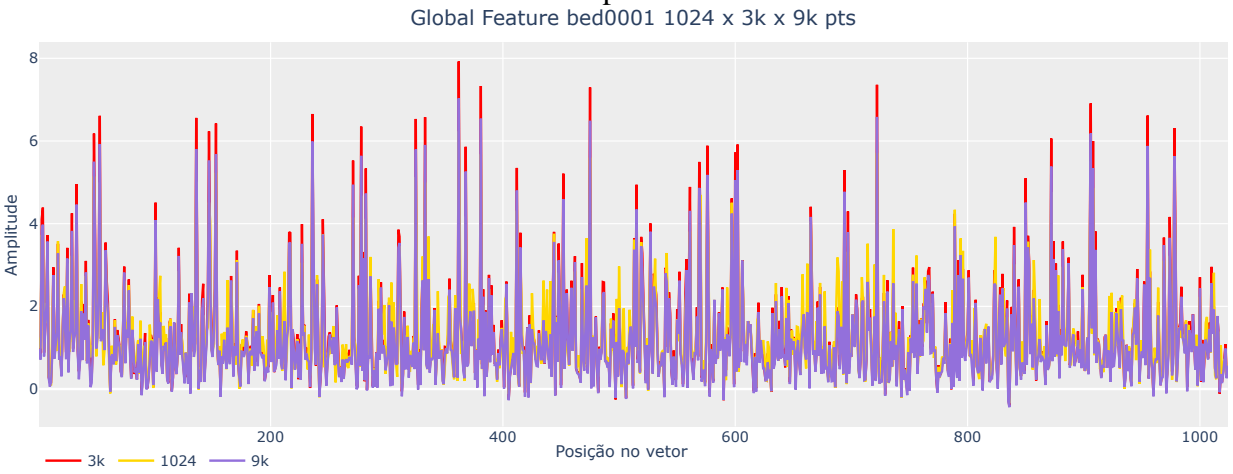


Fonte: elaborada pelo autor.

Para comparar graficamente os resultados das *global features*, foi escolhido utilizar gráficos de linha. Apesar da *global feature* ser um vetor de atributos, e não uma série temporal, ao ilustrar elas com gráfico de linha podemos visualizar mais facilmente a sua diferença para com as outras. Foram comparadas *global features* para o objeto bed0001 de tamanho 1024, 3000 e 9000.

Como mostra a Figura 5, há bastante sobreposição nas linhas dos três tamanhos, mostrando que as três *global features* possuem valores bem semelhantes.

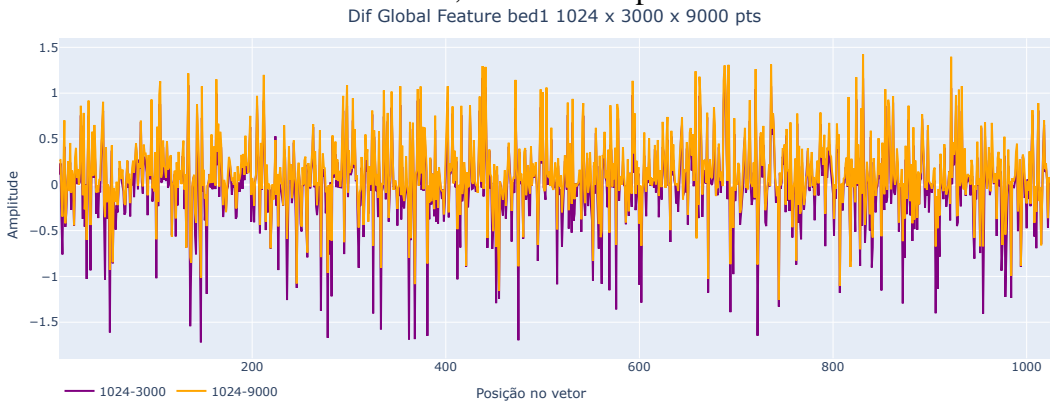
Figura 5 – Comparação entre *global features* de bed0001 com amostragens de 1024, 3000 e 9000 pontos



Fonte: elaborada pelo autor.

Para verificar de maneira mais apropriada as diferenças entre as três *global features*, foram gerados os gráficos de linha da diferença entre a *global feature* da nuvem de 1024 e 3000 pontos, e entre a nuvem de 1024 e 9000 pontos. Como mostra a Figura 6, a dispersão das diferenças entre as *global features* se mantém baixa. A faixa de diferença entre a nuvem de 1024 e 9000 ficou em torno de -0,5 e 0,5 considerando os seus picos, o que representa uma diferença absoluta de 1, sendo seus valores mais frequentes no entorno de -0,25 e 0,25, ou uma diferença absoluta de 0,5. Já a faixa de diferença entre a nuvem de 1024 e 3000 apresentou picos maiores. O RMSE da diferença entre a nuvem de 1024 e 3000 pontos foi 0,31 e de 1024 e 9000 foi 0,44.

Figura 6 – Comparação da diferença entre *global features* do objeto bed0001 com amostragens de 1024, 3000 e 9000 pontos



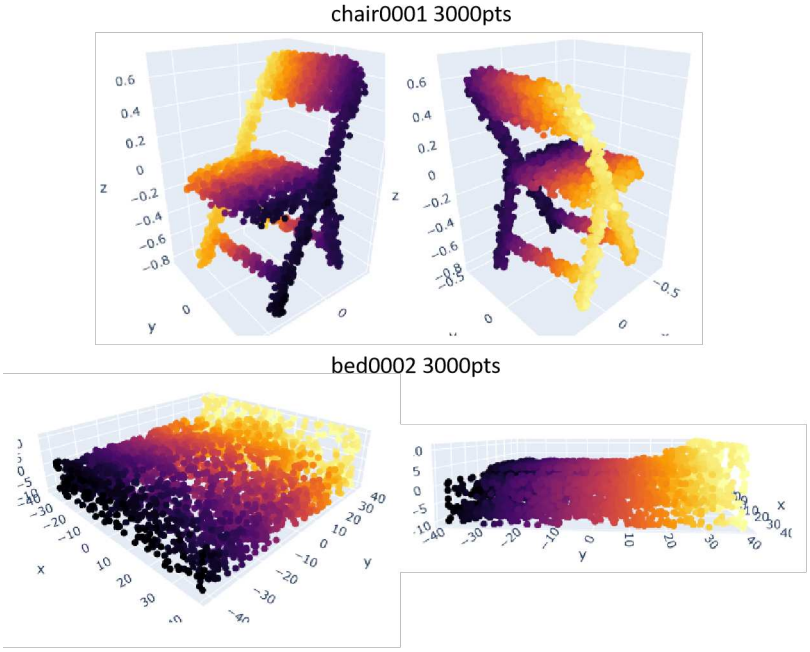
Fonte: elaborada pelo autor.

Apesar de termos visto graficamente que a diferença entre a *global feature* de diferentes tamanhos ser pequena, convém realizarmos essa inspeção para a diferença entre objetos de duas classes diferentes e também da mesma classe, para vermos a magnitude da sua diferença e compararmos com a inspeção anterior.

Para a comparação entre classes, foi selecionada uma nuvem de tamanho 3000, também arbitrariamente: o primeiro objeto da classe cadeira, o chair0001, e dessa mesma classe foi selecionado o segundo objeto da classe mesa, o bed0002. Podemos vê-los na Figura 7.

Analisando as Figuras 8 e 9, podemos ver que os objetos da mesma classe possuem a *global feature* bem próxima por conta da sobreposição das linhas, já as nuvens de pontos de classes possuem formas bem distintas. Inspecionando a Figura 8, com as diferenças entre a *global feature* do bed0001 e bed0002, podemos confirmar a análise anterior e também a primeira análise sobre os diferentes tamanhos de nuvem. A diferença da *global feature* do bed0001 e bed0002 mostra-se bem pequena em comparação com a de bed0001 e chair0001. Além de que a magnitude da diferença entre bed0001 e bed0002 ser bem semelhante à magnitude da diferença

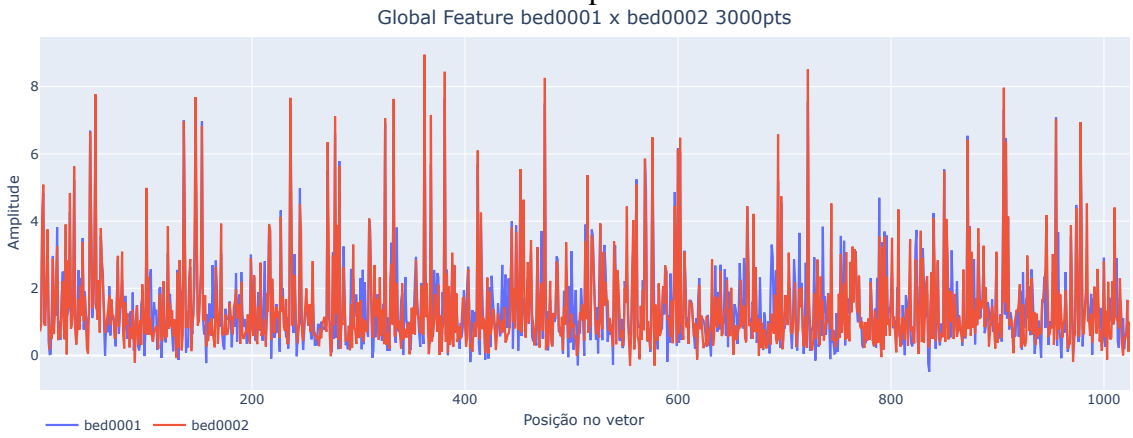
Figura 7 – Objetos chair0001 e bed0002 com amostragem 3000 pontos



Fonte: elaborada pelo autor.

entre os tamanhos 1024,3000 e 9000 do bed0001.

Figura 8 – Comparação entre *global features* dos objetos bed0001 e bed0002 com amostragens de 3000 pontos

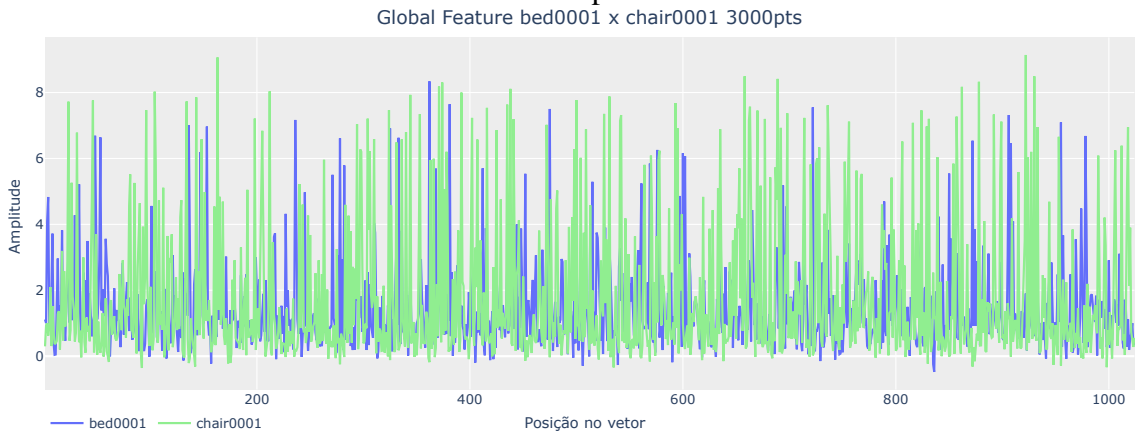


Fonte: elaborada pelo autor.

3.2.2 Global Feature para treinamento com nuvens de 6000 pontos

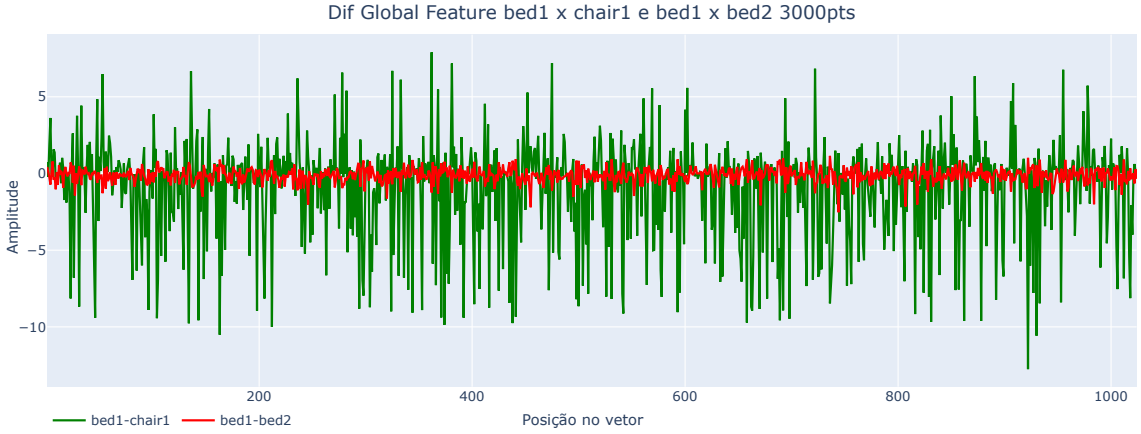
Para verificar se os mesmos resultados se reproduzem em uma rede PointNet treinada com mais pontos, os mesmos gráficos foram gerados para *global features* resultantes de uma rede treinada com nuvens de 6000 pontos. de 6000 pontos. Para tentar notar alguma diferença mais brusca, o tamanho das nuvens foi aumentado para 6000, quase seis vezes o tamanho das nuvens do treinamento anterior, ao invés de fazermos uma progressão com incrementos menores.

Figura 9 – Comparação entre *global features* dos objetos bed0001 e chair0001 com amostragens de 3000 pontos



Fonte: elaborada pelo autor.

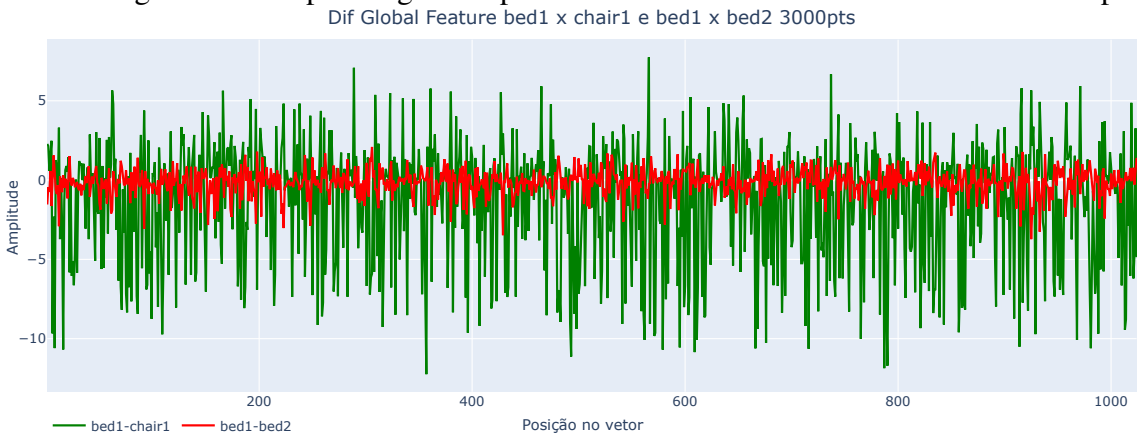
Figura 10 – Comparação da diferença entre *global features* dos objetos bed0001 e chair0001 com amostragens de 3000 pontos



Fonte: elaborada pelo autor.

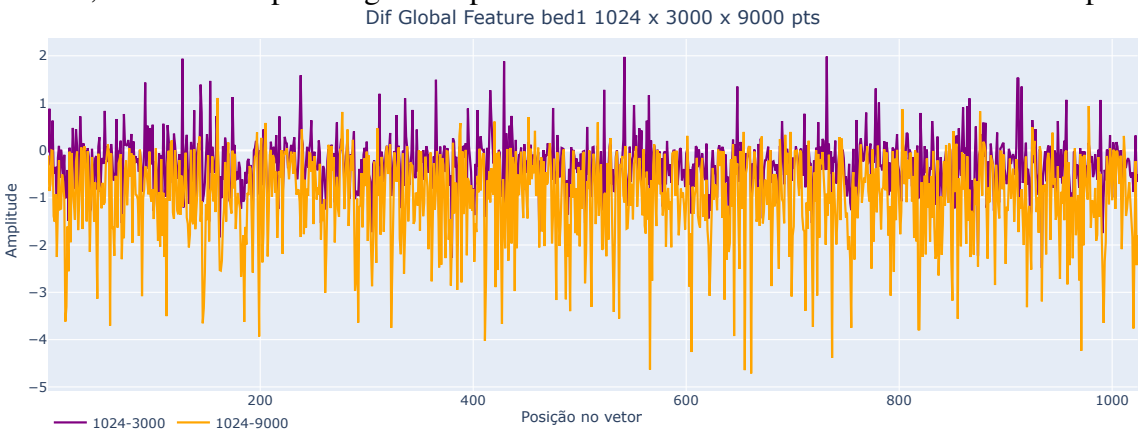
Como já sabemos que objetos de uma mesma classe possuem *global feature* semelhante e de classes diferentes possuem padrões distintos. Por concisão, convém analisarmos apenas os gráficos das diferenças. Analisando a Figura 11, e comparando a Figura 10, podemos ver pelas *global features* geradas que a rede treinada com nuvens de tamanho 6000 aumentou a sua diferenciabilidade entre os objetos. Também podemos verificar essa afirmação através do RMSE das diferenças consideradas: para bed0001 e chair001, foi 3,98 contra 4,20; e para bed0001 e bed0002, foi 0,48 contra 0,73. Inspecionando a Figura 12 e comparando com a Figura 6, vemos que esse efeito se repete para nuvens de uma mesma classe mas com tamanhos diferentes. Também podemos verificar essa afirmação através do RMSE das diferenças consideradas: para 1024 e 3000 pontos, foi 0,31 contra 0.41; e para 1024 e 9000 pontos, foi 0,44 contra 0.54.

Figura 11 – Comparação da diferença entre *global features* dos objetos bed0001 e chair0001 com amostragens de 3000 pontos geradas por uma PointNet treinada com nuvens de 6000 pontos



Fonte: elaborada pelo autor.

Figura 12 – Comparação da diferença entre *global features* do objeto bed0001 com amostragens de 1024, 3000 e 9000 pontos geradas por uma PointNet treinada com nuvens de 6000 pontos



Fonte: elaborada pelo autor.

3.3 Acurácia das redes PointNet treinadas com diferentes tamanhos

Verificamos anteriormente os efeitos de nuvens de diferentes tamanhos na *global feature*, apesar de conseguirmos extrair alguns padrões desses efeitos, devemos verificar o impactos deles no desempenho das redes. Para verificar o desempenho das redes, foi computada a acurácia média das classes separadamente, geradas em uma matriz de confusão. Para cada rede, a validação foi feita com o conjunto de dados de teste da ModelNet10 com nuvens de tamanho 1024 e 6000.

Analisando a Tabela 1, podemos ver que a PointNet treinada com nuvens de tamanho 6000 teve um desempenho superior em ambos os testes, o que pode ser explicado pelo aumento na diferenciação de objetos que verificamos anteriormente.

Comparando as matrizes de confusão da Figura 13 para os resultados das redes treinadas com nuvens de 1024 e 6000 e validadas com nuvens de 1024, podemos ver que existem

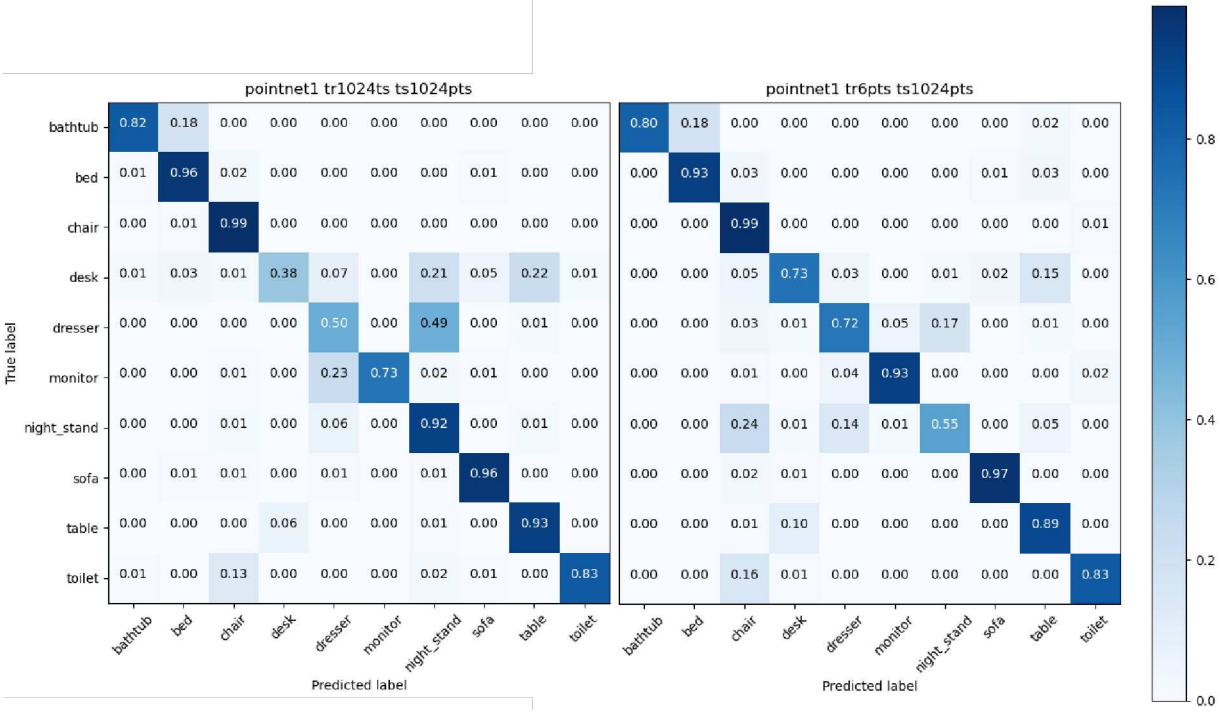
Tabela 1 – Comparação entre os modelos treinados para classificação com nuvens de pontos de tamanho 1024 e 6000

Modelo	Acurácia Ts 1024	Acurácia Ts 6000
PointNet Tr 1024	0,80	0,80
PointNet Tr 6000	**0,83**	**0,83**

Fonte: elaborada pelo autor.
Tr:utilizado no treinamento, Ts:utilizado no teste/validação

classes que a rede treinada com 1024 se confundia e a rede treinada com 6000 aprendeu a distingui-las melhor apenas com os dados, sem necessidade de se alterar a arquitetura da rede.

Figura 13 – Matrizes de confusão para os modelos treinados com nuvens de pontos de tamanho 1024 e 6000



Fonte: elaborada pelo autor.
À esquerda, a matriz de confusão de um modelo PointNet para classificação treinado e testado com nuvens de tamanho 1024. À direita, a matriz de confusão de outro modelo PointNet para classificação treinado com nuvens de tamanho 6000 e testado com nuvens de tamanho 1024.

3.4 Mudança de tamanho da Global Feature

Um dos objetivos do presente trabalho, além de conseguir utilizar a PointNet para predição de descritores geométricos, é conseguir fazê-la ter um desempenho melhor do que o cálculo iterativo do mesmo. Para verificar como algumas modificações poderiam alterar o tempo de execução da rede, foi realizada uma mudança no tamanho da *global feature*. Foram alterados:

o tamanho da saída da última rede antes do *max pooling* para gerar um vetor de dimensão 2048 ao invés de 1024 e a entrada rede de classificação para o mesmo tamanho.

Para termos alguma referência de tempo, também foram registrados os tempos de execução das redes treinadas com nuvens de tamanho 1024 e 3000, além do próprio cálculo dos descritores geométricos.

Analisando a Tabela 2, podemos ver que apenas aumentar o tamanho da *global feature*, o tempo de execução da rede aumenta 1,9 vez, praticamente dobrndo o tempo. Era esperado que a sua acurácia fosse menor que as outras redes, pois, ao mudar apenas a dimensão de 1024 para 2048, a convolução que realiza esse aumento recebe um vetor de tamanho 512, então o aumento muito brusco diminui a resolução/qualidade da operação. Entretanto, ainda assim, o seu tempo é bem inferior ao cálculo dos descritores.

A execução das redes foi realizada em CPU, em vez de GPU, para conseguirmos comparar adequadamente os tempos com o cálculo dos descritores. Podemos destacar como não esperado o tempo médio de execução das redes em nuvens de pontos de tamanho 6000, mesmo sendo executadas em momentos diferentes.

Tabela 2 – Resultado dos testes nos modelos treinados com nuvens de pontos de tamanhos diferentes e com tamanho da *global feature* aumentado

Modelo	Acurácia Ts 1024	Acurácia Ts 6000	Tempo médio (s) Ts 1024	Tempo médio (s) Ts 6000
PointNet Tr 1024	0,80	0,80	0,0231	0,0249
PointNet Tr 6000	**0,83**	**0,83**	**0,0226**	**0,0229**
PointNet GF2048 Tr 1024	0,79	0,78	0,0422	0,0438
Omnivariância calculada	-	-	0,3458	0,3560

Fonte: elaborada pelo autor.
GF2048: *global feature* tamanho 2048

3.5 Considerações Finais

Com as análises feitas em redes treinadas com nuvens de diferentes tamanhos, conseguimos verificar a tendência da rede em aumentar a diferenciabilidade dos objetos refletidos nas *global features*, consequentemente aumentando a sua acurácia para classificação. Vale destacar que, por conta das etapas de *input transform* e *feature transform*, além da própria natureza das redes neurais, o treinamento ajusta os parâmetros da rede a se ajustarem para a tarefa em questão. Nessa perspectiva, a conclusão do aumento de diferenciabilidade das *global*

features deve ser associado a tarefas de classificação, pois a rede pode criar outras tendências para outras tarefas. Ainda assim, considerando que há precedentes para melhorar o desempenho das redes PointNet ao aumentar o tamanho das nuvens utilizadas no seu treinamento, convém utilizarmos essa estratégia para os testes com a PointNet para predição de descritores.

4 EXPERIMENTOS PARA REGRESSÃO

Para verificar se a arquitetura da PointNet pode ser utilizada para tarefas de regressão ponto-a-ponto para predizermos os descritores geométricos de uma nuvem de pontos, algumas modificações na arquitetura foram realizadas e testadas. Ao todo, foram propostas cinco arquiteturas com três estratégias diferentes: segmentação adaptada, sem *global feature* e segmentação avançada. Dessa maneira, podemos avaliar a capacidade de adaptação da PointNet para regressão, bem como se aprofundar nas funções das etapas de sua arquitetura.

Os testes das arquiteturas foram realizados primeiramente utilizando o conjunto de dados do ModelNet10 com nuvens de tamanho 2500, para em seguida comparar seus resultados e eleger a arquitetura com maior potencial para a tarefa de regressão.

Dada uma arquitetura escolhida, foram realizados outros testes com os conjuntos de dados ShapeNet e ModelNet40 com nuvens de tamanhos diferentes. Dessa forma, torna-se possível avaliar o desempenho da rede ao ser treinada com volumes de dados e objetos diferentes, além da capacidade para lidar com esses objetos. No primeiro teste, com a arquitetura de segmentação adaptada, também foi realizado o teste de eficácia de funções de perda diferentes para elegermos a mais apropriada para os testes seguintes.

Na falta de uma métrica destinada para avaliar regressão ponto-a-ponto em nuvens de pontos, por conta da sua dimensionalidade e dificuldade de sintetizar a relação do valor predito com a geometria do objeto, será utilizado a mediana do erro absoluto entre o descritor predito e o calculado, que será referenciado no decorrer do texto como MEA.

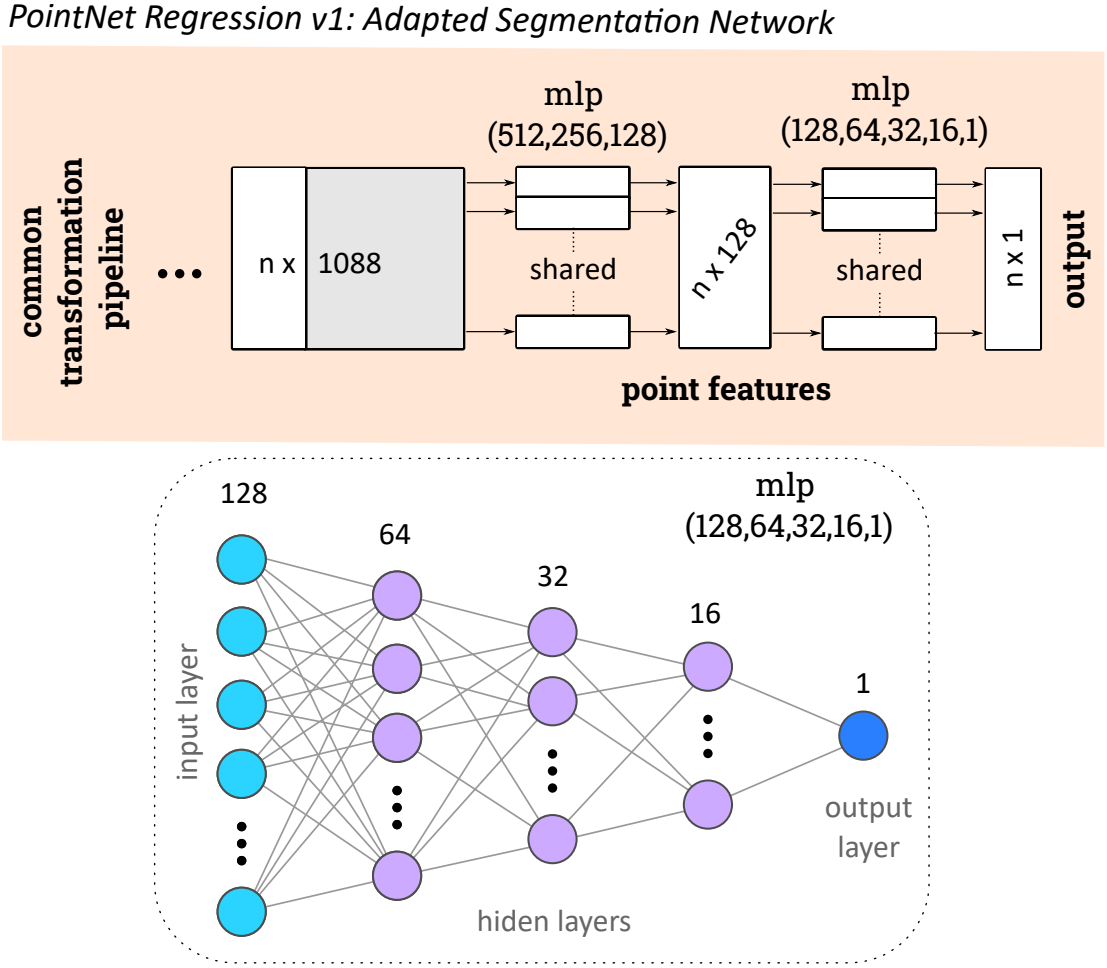
4.1 PointNet para Segmentação Adaptada (v1)

A rede de segmentação da PointNet foi desenvolvida para realizar segmentação por partes, que pode ser considerado como uma classificação local na nuvem de pontos. A rede associa informações gerais da nuvem com informações locais para fazer essa segmentação, abrindo precedentes para utilizarmos essa associação para outras tarefas. Dessa maneira, a primeira arquitetura proposta para regressão utiliza a mesma da segmentação com as devidas adaptações.

Para a arquitetura de segmentação classificar os pontos nas suas respectivas partes da nuvem, a camada de saída gera um vetor de tamanho m classes, como vimos na Figura 1. Cada posição do vetor de saída é uma probabilidade do referido ponto pertencer a uma parte da nuvem,

que em seguida é maximizado por uma função LogSoftmax para indicar a classe mais provável. Para adaptarmos essa rede para regressão, precisamos eliminar essa última etapa de maximização da probabilidade e acrescentar camadas para reduzir a dimensionalidade da camada de saída para 1, como vemos na Figura 14. Dessa maneira, fazemos uma redução de dimensionalidade mais compassada, melhorando a resolução da operação, obtendo como resposta na camada de saída apenas o valor do descritor predito no referido ponto.

Figura 14 – Arquitetura Rede de Segmentação Adaptada (v1)



Fonte: elaborada pelo autor.
Recorte da adaptação feita na rede segmentação da PointNet após o *common transformation pipeline*, iniciando a rede com a concatenação dos pontos em dimensão 64 após a etapa de *feature transform* com o vetor de *global feature*. Normalização em lote é usado para todas as camadas com ReLU.

Para o treinamento, dos 4899 objetos da ModelNet10, foram separados 3991 amostras para o treinamento e 908 para o teste, correspondendo, respectivamente, à 81,5% e 18,5% do conjunto de dados. Essa separação já vem definida no próprio conjunto de dados, onde os dados de treino e teste já estão separados em pastas.

Para avaliar o desempenho de uma rede neural, podemos começar analisando o

desempenho do seu treinamento pelo decaimento dos valores da função de perda escolhida. Uma rede neural começa predizendo valores bem distantes do real esperado, mas essa diferença diminui ao decorrer das épocas de treinamento, como visto no Capítulo 2. A velocidade com que os resultados da função de perda diminuem (para funções de perda de erro) indicam que os resultados previstos pela rede neural estão convergindo para os valores esperados. Dessa maneira, podemos avaliar o desempenho de funções de perda diferentes ao analisarmos o decaimento dos seus valores no decorrer das épocas. Com a análise do treinamento também podemos verificar a quantidade de épocas adequadas para o treinamento, pois, se treinarmos a rede com muita épocas, podemos levá-la ao super ajuste nas previsões, perdendo capacidade de generalização da mesma.

Os valores gerados pelas funções de perda pertencem ao domínio da sua própria função, então mesmo tendo desempenhos parecidos, os valores estarão em escalas diferentes e não poderemos compará-los diretamente. Normalizar os valores também não seria tão efetivo, pois funções de perda mais sensíveis ao erro podem apresentar variações mais bruscas e comparar com outra função menos sensível poderia enviesar a análise. Por conda disso, em cada treinamento das funções de perda, também foram computados as outras funções que não foram utilizadas, dessa maneira, poderemos comparar entre si de forma mais adequada os seus desempenhos.

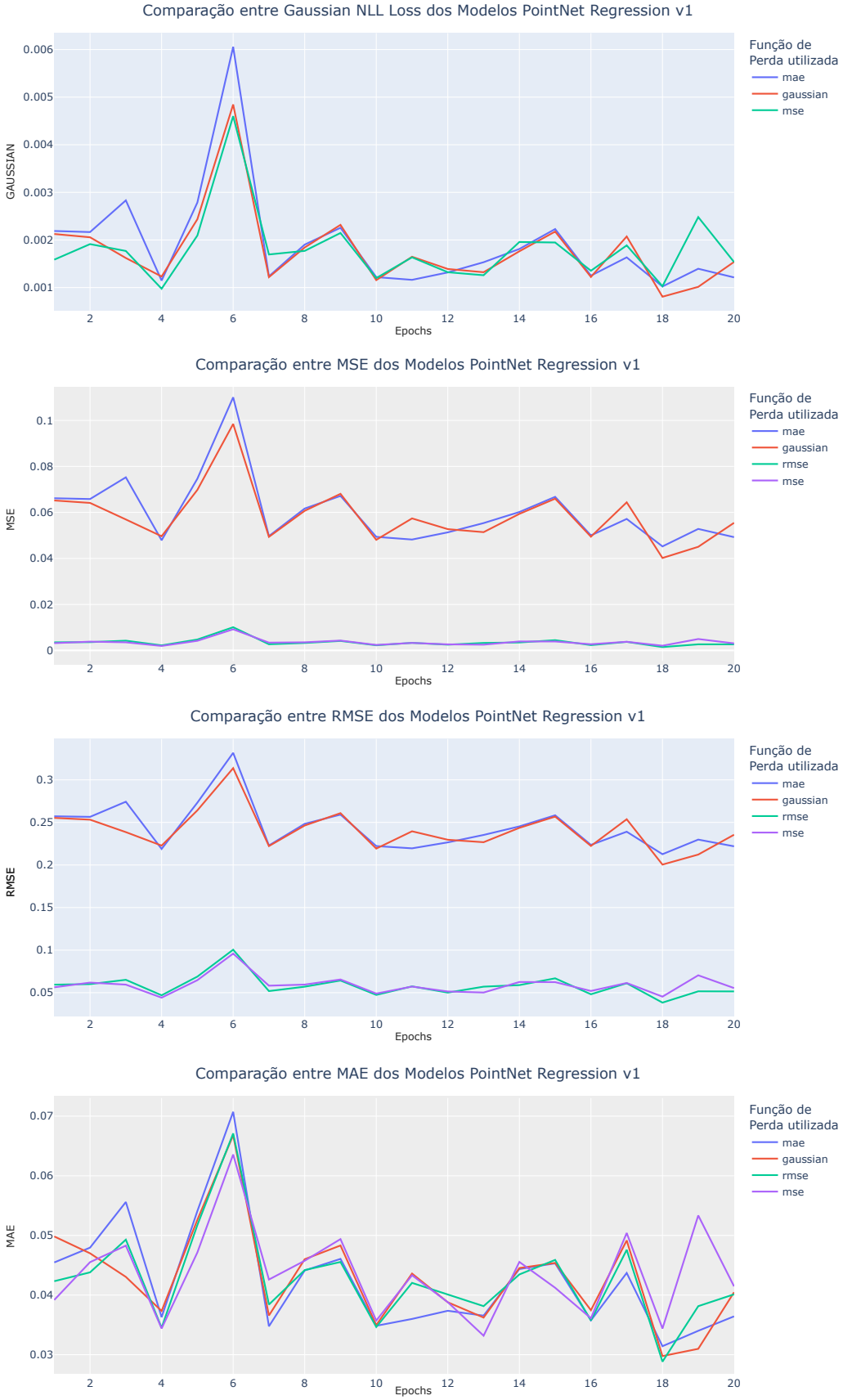
Analisando a Figura 15, podemos ver que o decaimento das funções de perda é bem lento, mas suas magnitudes não são tão altas, considerando que MAE, MSE e RMSE são baseados no erro absoluto, mas, no caso da Gaussian NLL Loss, com natureza estatística mais complexa, não podemos fazer a mesma interpretação sem outra referência. No caso das funções de perda baseadas no erro absoluto, como estamos considerando a média de uma média, os erros podem estar sendo amortizados e assim não sabemos o que realmente está acontecendo na predição dos descritores.

Como estamos tentando predizer a omnivariância dos pontos de uma nuvem, pequenas porções locais com valores mais baixos do descritor podem estar sendo preditos de maneira errada, com valores mais baixos, e isso pode não aparecer nas métricas por conta da amortização.

Como não podemos realizar uma extensa inspeção visual dos resultados pelo número de testes realizados, a análise para a escolha da função de perda será simplificada considerando apenas o desempenho do treinamento e da validação dos modelos.

Para a quantidade de épocas treinadas, podemos ver os menores valores das funções

Figura 15 – Arquitetura Rede de Segmentação Adaptada



Fonte: elaborada pelo autor.
O RMSE foi retirado do primeiro plot pois o eu erro elevado prejudicou a escala.

de perda ocorrem na época 18, dessa maneira, a validação dos modelos foi realizada utilizando seus estados nessa época. Para verificar se algum modelo aprendeu a generalizar melhor que os outros, testes também foram realizados no conjunto de dados ShapeNet, com 16 classes e 2874 objetos no conjunto de validação do mesmo, com classes que não existem na ModelNet10.

Avaliando a Tabela 3, podemos ver que, apesar da natureza das funções de perda baseado na diferença absoluta serem mais simples, a Gaussian NLL Loss apresentou o desempenho mais baixo se considerarmos a média dos seus resultados. Analisando as medianas, os resultados foram bem próximos. Ao considerarmos os resultados obtidos nos testes com os dados da ShapeNet, é possível perceber que os resultados ficaram bem mais distantes do que os com a ModelNet10.

Tabela 3 – Comparação entre os modelos v1 treinados com funções de perda e conjunto de dados diferentes

v1 Tr	Média MEA	Mediana MEA
ModelNet10 GAUS	0,116	0,046
ModelNet10 MAE	**0,049**	**0,043**
ModelNet10 MSE	0,058	0,042
ModelNet10 RMSE	0,051	0,044
ShapeNet GAUS	0,343	0,289
ShapeNet MAE	0,295	0,278
ShapeNet MSE	0,295	0,287
ShapeNet RMSE	0,293	0,287

Fonte: elaborada pelo autor.

Dentre as funções de perda consideradas, embora não haja diferença significativa, a MAE foi que demonstrou melhores resultados em ambos os testes. A MAE pode ter sido privilegiada por conta da semelhança com o cálculo utilizado para estimar os erros. Como mencionado anteriormente, para cada nuvem, foi calculada a mediana dos erros absolutos, que é bem próxima da média dos erros absolutos, a MAE. Mesmo assim, como o resultado das outras funções de perda não foram distantes da MAE, é possível afirmar que essa coincidência não favoreceu demais a mesma.

Dessa maneira, mesmo com resultados bem próximos, é necessário eleger uma função de perda para a realização dos próximos experimentos. Por conta dos seus resultados menores na validação feita com ModelNet10 e ShapeNet, a função escolhida foi a MAE.

Para finalizarmos os experimentos com a PointNet para segmentação adaptada, convém testarmos o achado anterior: aumentar o tamanho das nuvens de pontos no treinamento

melhora a acurácia para classificação e há possibilidade de também melhorar na regressão.

Foi treinada uma rede com a mesma arquitetura, mas utilizando nuvens de tamanho 5000 da ModelNet10, e foram validados em nuvens de tamanho 2500 como nos testes anteriores. Analisando a Tabela 3 e comparando com a Tabela 4, podemos ver que, apesar de ser esperado uma melhora com base nos testes de classificação, os resultados pioraram ao utilizarmos nuvens de tamanhos maiores na segmentação.

Tabela 4 – Validação dos modelos v1 treinados com nuvens de tamanho 2500 sob nuvens de 5000

v1 Tr 5000	Média MEA	Mediana MEA
Ts ModelNet10 2500	**0,072**	**0,047**
Ts ShapeNet 2500	0,301	0,292

Fonte: elaborada pelo autor.

4.2 PointNet sem Global Feature

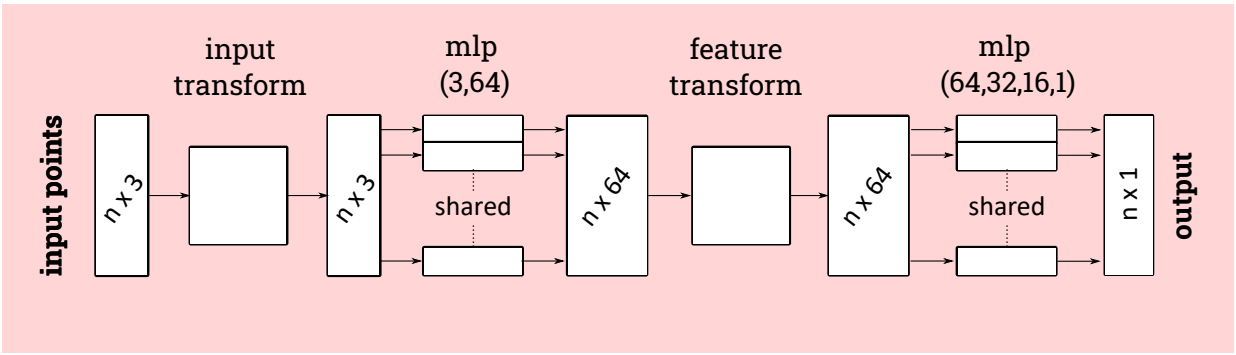
Na estratégia anterior, foi adaptada a arquitetura de segmentação da PointNet com as devidas modificações para regressão ponto-a-ponto, onde é associado informação global e local no vetor de entrada, mas, é importante notar uma característica intrínseca à arquitetura da PointNet no que diz respeito aos atributos usados para regressão: do total de atributos, a maioria é oriunda de informação global.. O vetor de entrada da rede de segmentação possui tamanho 1088, onde 1024 é da *global feature* e apenas 64 é da informação local do ponto, correspondendo, respectivamente, à 94% e 6% dos 1088. Dessa maneira, como o valor de um descritor geométrico é essencialmente local, ter mais informação global pode estar enviesando a rede a realizar a predição baseando-se na forma geral da nuvem. Nessa perspectiva, considerando que há pouca representatividade da informação local, uma estratégia considerando apenas a informação local foi proposta.

Como explicado na fundamentação teórica, *Charles et al*.(2017) argumenta que, ao aumentarmos a dimensionalidade dos pontos e aplicando-os às mesmas transformações, os pontos espacialmente próximos apresentarão informações semelhantes. Dessa maneira, para considerarmos apenas a informação local dos pontos, podemos utilizar os pontos em alta dimensão juntamente com a arquitetura de regressão proposta anteriormente para realizar a predição do descritor. Para validar essa estratégia, foram pensadas três arquiteturas utilizando os pontos em dimensões diferentes.

4.2.1 PointNet Parcial 64 (v2 64)

Nessa primeira arquitetura, foi utilizada a mesma estrutura da PointNet até a etapa de *feature transform* juntamente com uma rede neural para regressão, a mesma da etapa anterior, retirando apenas a primeira camada de 128 para 64. A Figura 16 ilustra essas modificações na arquitetura.

Figura 16 – Arquitetura Parcial da PointNet (v2 64)



Fonte: elaborada pelo autor.

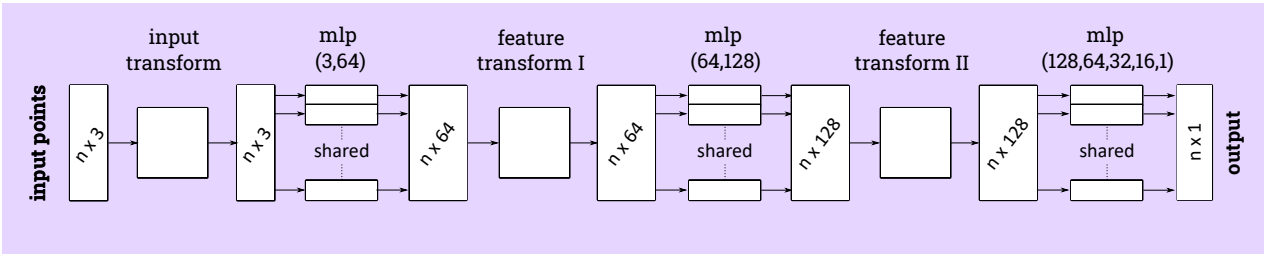
4.2.2 PointNet Parcial 128 (v2 128)

Nessa segunda arquitetura, também foi utilizada a mesma estrutura da PointNet até a etapa de *feature transform*, entretanto, foi adicionado uma convolução de 64 para 120 e outra *feature transform* de tamanho 128, juntamente com a mesma rede neural para regressão da PointNet para segmentação adaptada. *Charles et al.* (2017) argumenta que a T-Net é uma arquitetura modular que pode ser utilizada para alinhar pontos em diferentes dimensões, sendo utilizada nas etapas de *input transform* e *feature transform*, que tem dimensão 3 e 64 respectivamente. Dessa maneira, uma segunda *feature transform* foi adicionada para verificar se essa capacidade de alinhamento pode melhorar o desempenho da rede. A Figura 17 ilustra essas modificações na arquitetura.

4.2.3 PointNet Parcial 1024 (v2 1024)

Nessa terceira arquitetura, foi utilizada a mesma arquitetura da PointNet até a etapa anterior ao *max pooling* juntamente com uma rede de regressão, no mesmo formato das outras apresentadas anteriormente, adicionando apenas mais camadas para comportar o tamanho da

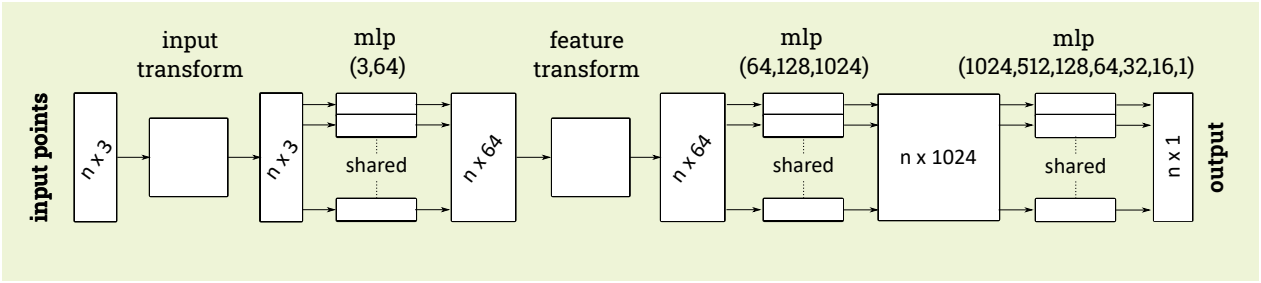
Figura 17 – Arquitetura Rede de Segmentação Adaptada (v2 128)
PointNet Regression v2_128: Parcial PointNet Network



Fonte: elaborada pelo autor.

entrada. Nessa arquitetura, não foi adicionado outra etapa de *feature transform*, apenas mantendo parte da PointNet original, pois assim poderemos verificar se, seguindo a mesma, removendo-se apenas a *global feature*, teremos resultados melhores. A Figura 18 ilustra essas modificações na arquitetura.

Figura 18 – Arquitetura Rede de Segmentação Adaptada (v2 1024)
PointNet Regression v2_1024: Parcial PointNet Network



Fonte: elaborada pelo autor.

4.3 PointNet para Segmentação Avançada Adaptada (v3)

No suplemento do artigo de concepção da PointNet, *Charles et al.*(2017) define a arquitetura de segmentação apresentada na Figura 1 como uma arquitetura básica de segmentação e apresenta outra arquitetura mais robusta, que nesse texto será designada como arquitetura avançada de segmentação. A terceira estratégia utilizada para regressão de preditores foi utilizar uma adaptação dessa arquitetura avançada de segmentação.

Como podemos ver pela Figura 19, a arquitetura avançada de segmentação segue a mesma estratégia de concatenar informação global com local, mas concatena ainda mais informação local e aumenta o tamanho da *global feature*. Nela, foram adicionadas mais etapas a partir das camadas em dimensão 128, com uma rede totalmente conectada de 128 a 128 e

uma etapa de *feature transform* para o seu tamanho. A *feature transform* feita nos pontos em dimensão 64 foi deslocada para ser realizada nos pontos em dimensão 128, optando por realizar o ajuste dos pontos de forma mais tardia. Logo após a concatenação dos pontos nas diversas dimensões com a *global feature*, os passos da rede que realizam a segmentação não foi definida adequadamente no artigo, podemos inferir que há uma redução de dimensionalidade de 3024 para m classes, mas ainda se faz necessário contemplar a escolha pelo espaçamento dessa operação.

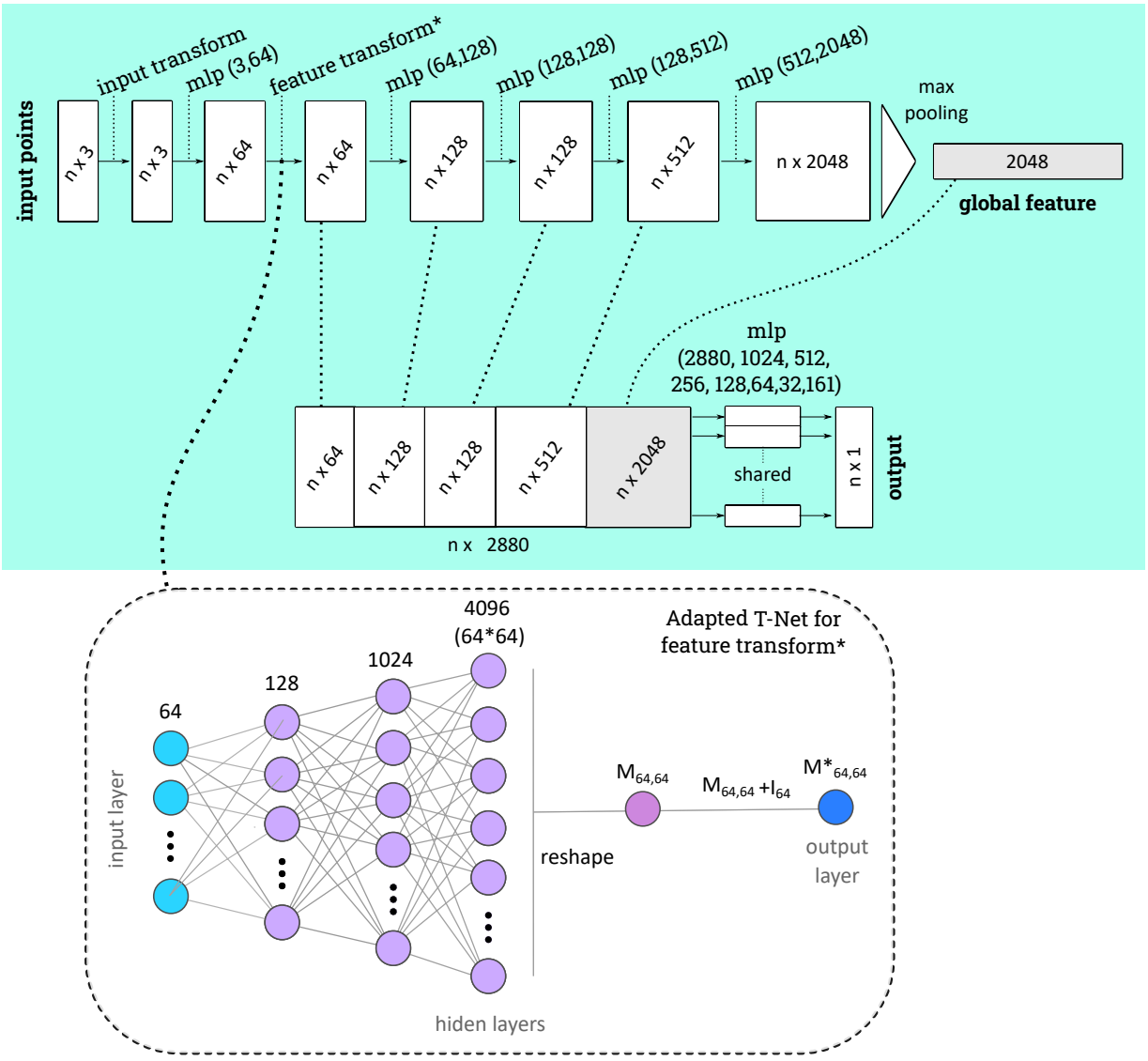
Na adaptação dessa arquitetura, em vez de apenas modificar a última rede de segmentação para regressão, algumas outras mudanças foram feitas com base nas observações feitas dos outros testes. A primeira modificação proposta foi retirar a etapa de *feature transform* em dimensão 128 e realizá-la novamente em 64.

Apesar de *Charles et al.*(2017) apresentar a T-Net como uma rede modular para o ajuste dos pontos em qualquer dimensão, observando com atenção a última etapa de mudança de dimensionalidade, de 256 para $k * k$, é possível notar que, para determinados valores de k, a mudança de dimensão se torna muito brusca. Ao utilizarmos com, por exemplo, $k = 64$, essa mudança ocorrerá de 256 para 4096 $(64 * 64)$, e com $k = 128$, de 256 para 16384. Mudanças de dimensionalidade muito bruscas podem resultar em perda de informação, tornando a eficiência da T-Net questionável para determinados valores de k. Nessa perspectiva, a segunda mudança idealizada foi alterar a arquitetura da T-Net, ajustando-a para gerar uma matriz 64x64 com uma mudança menos brusca de dimensão, que pode ser vista na Figura 19.

4.4 Considerações finais

Como podemos ver pelas arquiteturas propostas, há várias formas de articular o uso da informação local contida nos pontos através das transformações, seja levando essa informação para uma dimensão maior ou combinando-a com informações globais da nuvem. No próximo capítulo, poderemos considerar o efeito dessas mudanças através dos resultados nos testes.

Figura 19 – Arquitetura da Rede de Segmentação Avançada Adaptada (v3)
Advanced Segmentation Network Adapted



Fonte: elaborada pelo autor.
A *feature transform** é uma adaptação da rede T-Net. Recebe os pontos em dimensão 64 e aumenta seu tamanho até 4096 através de convoluções. Em seguida, o vetor 4096 é transposto em uma matriz $M_{64,64}$, onde esta é somada com a sua matriz identidade.

5 RESULTADOS

A validação dos modelos foi feita utilizando o conjunto de testes da ModelNet10 e executados em GPU. Analisando a Tabela 5, podemos ver que todos os resultados foram bem parecidos, onde, por diferença de milésimos, o modelo com a arquitetura PointNet Parcial 64 (v2 64) treinada com nuvens de pontos de tamanho 2500 foi a melhor. Entretanto, o modelo treinado com a arquitetura de segmentação avançada adaptada teve resultados quase idênticos, se diferindo apenas por 1 milésimo na média das medianas da diferença absoluta e no seu tempo minimamente superior.

Tabela 5 – Comparação entre os modelos treinados das arquiteturas propostas e tamanhos de nuvens de pontos diferentes.

Modelo	Média MEA	Mediana MEA	Tempo médio (s)
v1 Tr 2500	0,049	0,043	$9,972 \cdot 10^{-4}$
v1 Tr 5000	0,072	0,047	$9,972 \cdot 10^{-4}$
v2 64 Tr 2500	**0,044**	**0,039**	$\mathbf{9,684 \cdot 10^{-4}}$
v2 64 Tr 5000	0,051	0,045	$\mathbf{9,684 \cdot 10^{-4}}$
v2 128 Tr 2500	0,050	0,045	$9,972 \cdot 10^{-4}$
v2 128 Tr 5000	0,052	0,046	$9,972 \cdot 10^{-4}$
v2 1024 Tr 2500	0,051	0,042	$9,972 \cdot 10^{-4}$
v2 1024 Tr 5000	0,045	0,041	$9,972 \cdot 10^{-4}$
v3 Tr 2500	0,045	**0,039**	$9,972 \cdot 10^{-4}$
v3 Tr 5000	0,046	**0,039**	$9,972 \cdot 10^{-4}$

Fonte: elaborada pelo autor.

Apesar do tempo de processamento da rede ser sido praticamente o mesmo em todos os modelos, o tempo de treinamento não foi. Por conta do tamanho da rede, a arquitetura de segmentação avançada teve o maior tempo de treinamento, enquanto a PointNet Parcial 64 teve o menor tempo. Apesar da diferença mínima nas métricas apresentadas, precisamos eleger um modelo para continuar os testes nos outros conjuntos de dados. Devido à praticidade do tempo de treinamento menor, a arquitetura escolhida foi a PointNet Parcial 64.

Para o tempo de processamento dos modelos pela GPU ter praticamente o mesmo, mesmo embora tenham arquiteturas de tamanhos diferentes, o tempo registrado deve ser equivalente apenas ao tempo de carregamento dos dados e retorno dos resultados.

5.1 Comparação dos resultados do modelo elegido treinado com diferentes conjuntos de dados

Para analisar a capacidade de generalização do modelo treinado com a arquitetura PointNet Parcial 64 e conjunto de ModelNet10, precisamos comparar com outros modelos treinados com conjunto de dados diferentes. Para relembrar, a ModelNet10 possui 4.899 objetos (Tr:3.991, Ts:908) e 10 classes, a ModelNet40 possui 12.310 objetos (Tr:9.843, Ts:2.467) e 40 classes, e a ShapeNet possui 16.881 (Tr:14.007, Ts:2.874) e 16 classes. Além disso, foi retirada uma classe da ModelNet40 para validação, sendo uma classes de objetos nunca vista por nenhum dos modelos. Assim, poderemos analisar tanto o desempenho dos modelos quanto a capacidade da arquitetura.

A Tabela 6 corresponde ao desempenho dos modelos nos testes dos seus respectivos conjuntos de dados. Podemos ver que, embora a diferença seja pequena, os modelos treinados com a ModelNet40 e Shapenet tiveram desempenhos melhores do que o treinado com a ModelNet10. Podemos relacionar a melhora do desempenho com o volume de dados utilizados no treinamento dos modelos, já que a ModelNet40 tem 2x mais dados do que a ModelNet10 e a ShapeNet tem 3,5x mais dados. Outra observação que pode-se fazer é que mesmo a ShapeNet tendo mais dados do que a ModelNet40, o seu desempenho não foi melhor. Entretanto, não podemos generalizar que mais classes diferentes no treinamento é mais importante do que o volume, pois as classes da ShapeNet são consideravelmente desbalanceadas.

Tabela 6 – Validação dos modelos v2 64 em seus respectivos conjuntos de teste

Modelo	Média MEA	Mediana MEA
v2 64 Tr ModelNet40	**0,040**	**0,033**
v2 64 Tr ShapeNet	0,042	0,035

Fonte: elaborada pelo autor.

Para o teste final, como comentado, foi submetido aos modelos uma classe em que nenhum dos modelos teve contato com algo parecido no seu treinamento, a classe Person da ModelNet40, com 108 amostras. Essa classe possui modelos de seres humanos e humanoides em poses diferentes. Analisando a Tabela 7 que compila esse teste, podemos ver que a ModelNet10 conseguiu prevalecer nos seus resultados.

Dado que a análise das métricas foi concluída, podemos realizar uma inspeção gráfica dos descritores preditos. Analisando as Figuras 20,21,22,23 e 24 podemos identificar

Tabela 7 – Validação dos modelos v2 64 na classe Person

Modelo	Média MEA	Mediana MEA
v2 64 Tr ModelNet40	**0,036**	**0,029**
v2 64 Tr ModelNet10	0,049	0,031
v2 64 Tr ShapeNet	0,064	0,031

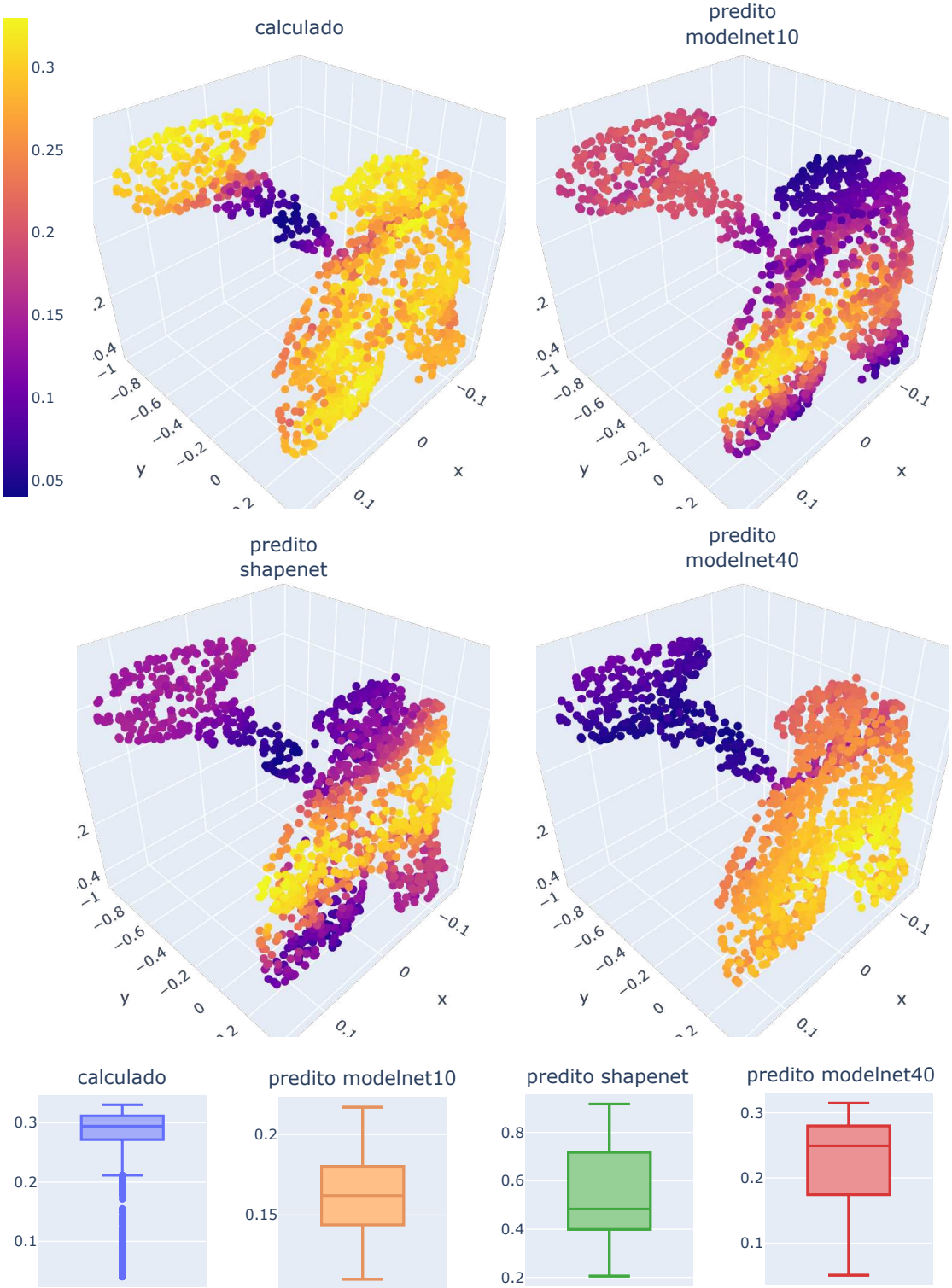
Fonte: elaborada pelo autor.

que, apesar dos resultados das suas métricas não serem tão distantes, nos levando a pensar que o resultado de suas predições seriam próximos, o modelos demonstraram padrões bem diferentes. Mesmo os modelos apresentando padrões diferentes em suas predições, podemos perceber que nenhum deles teve sucesso em se aproximar dos padrões gerados do cálculo da omnivariância, mostrando assim que não conseguiram generalizar sua solução.

5.2 Considerações finais

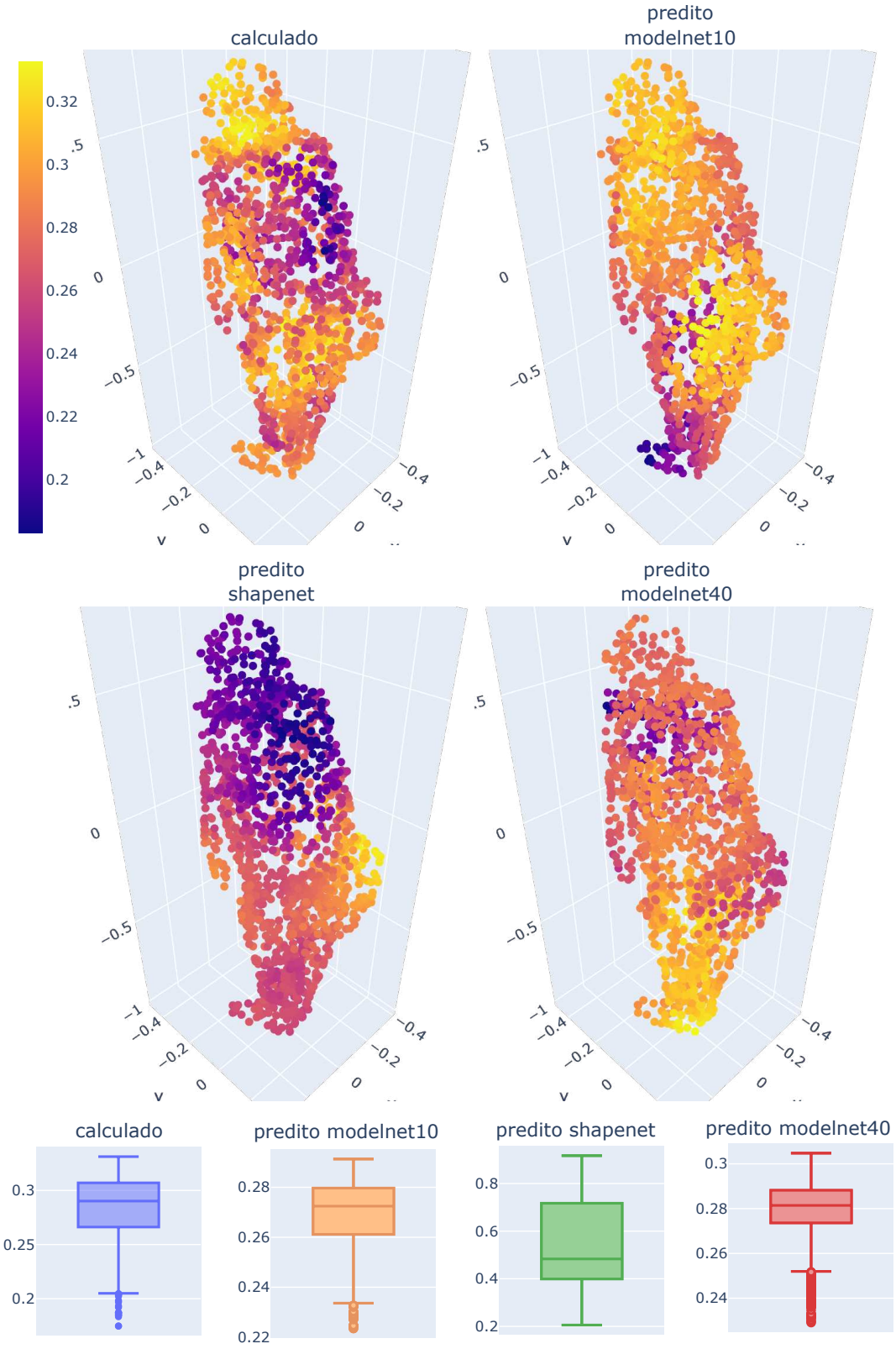
Neste Capítulo vimos que, sob as métricas utilizadas para avaliação, os modelos não apresentaram resultados destoantes entre si. Ainda assim, na validação dos modelos da arquitetura v2 64, elegida a melhor pelos testes, os resultados das sua predições não foram parecidos e nenhum conseguiu se aproximar da reprodução do cálculo da omnivariância de fato. Entretanto, apesar dos resultados, dado os testes feitos sob diferentes condições(conjuntos de dados, funções de perda, tamanhos das nuvens e arquiteturas), ainda podemos extrair observações significativas da PointNet e do seu uso para a tarefa de regressão.

Figura 20 – Predição do objeto person001 pelos modelos



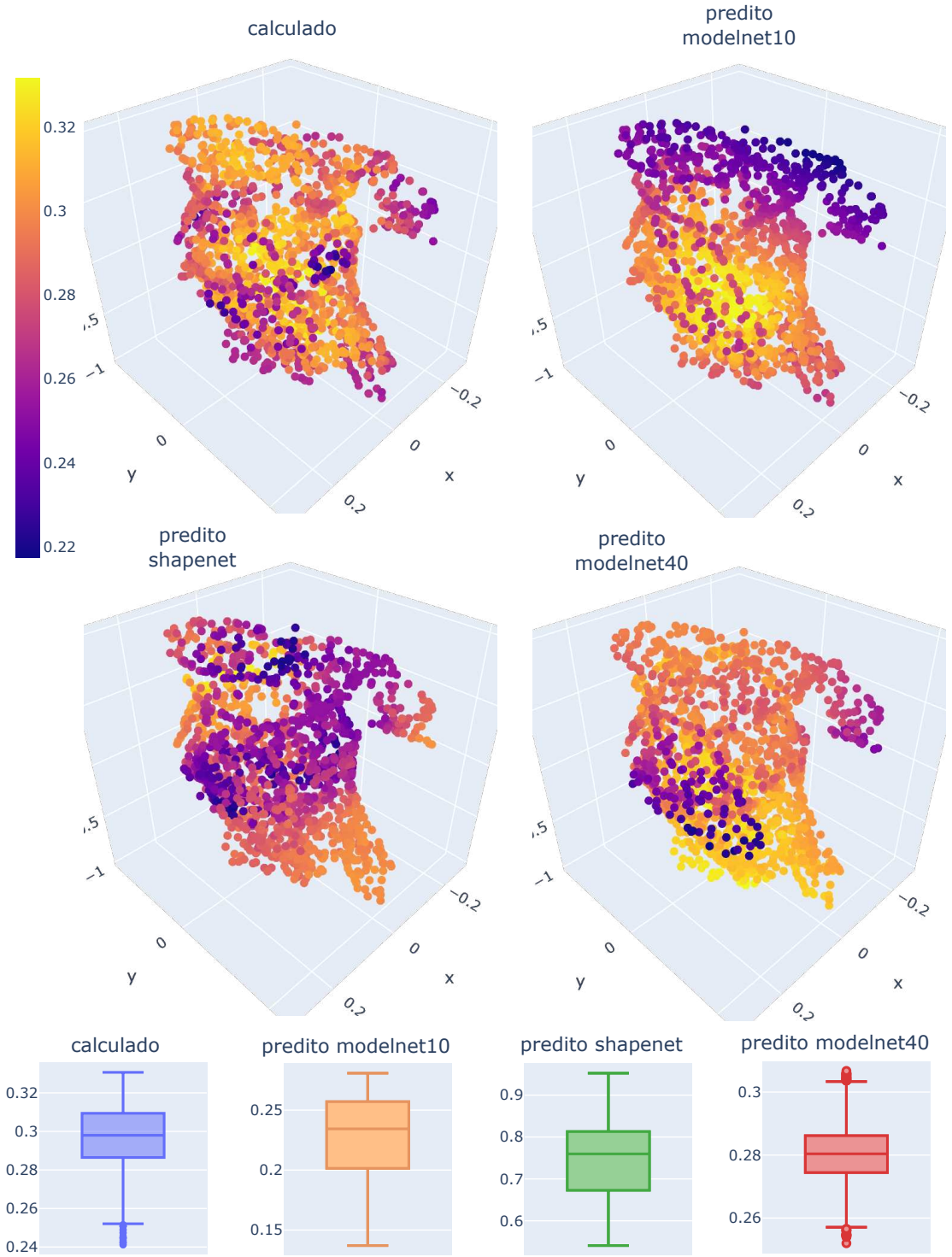
Fonte: elaborada pelo autor.

Figura 21 – Predição do objeto person002 pelos modelos



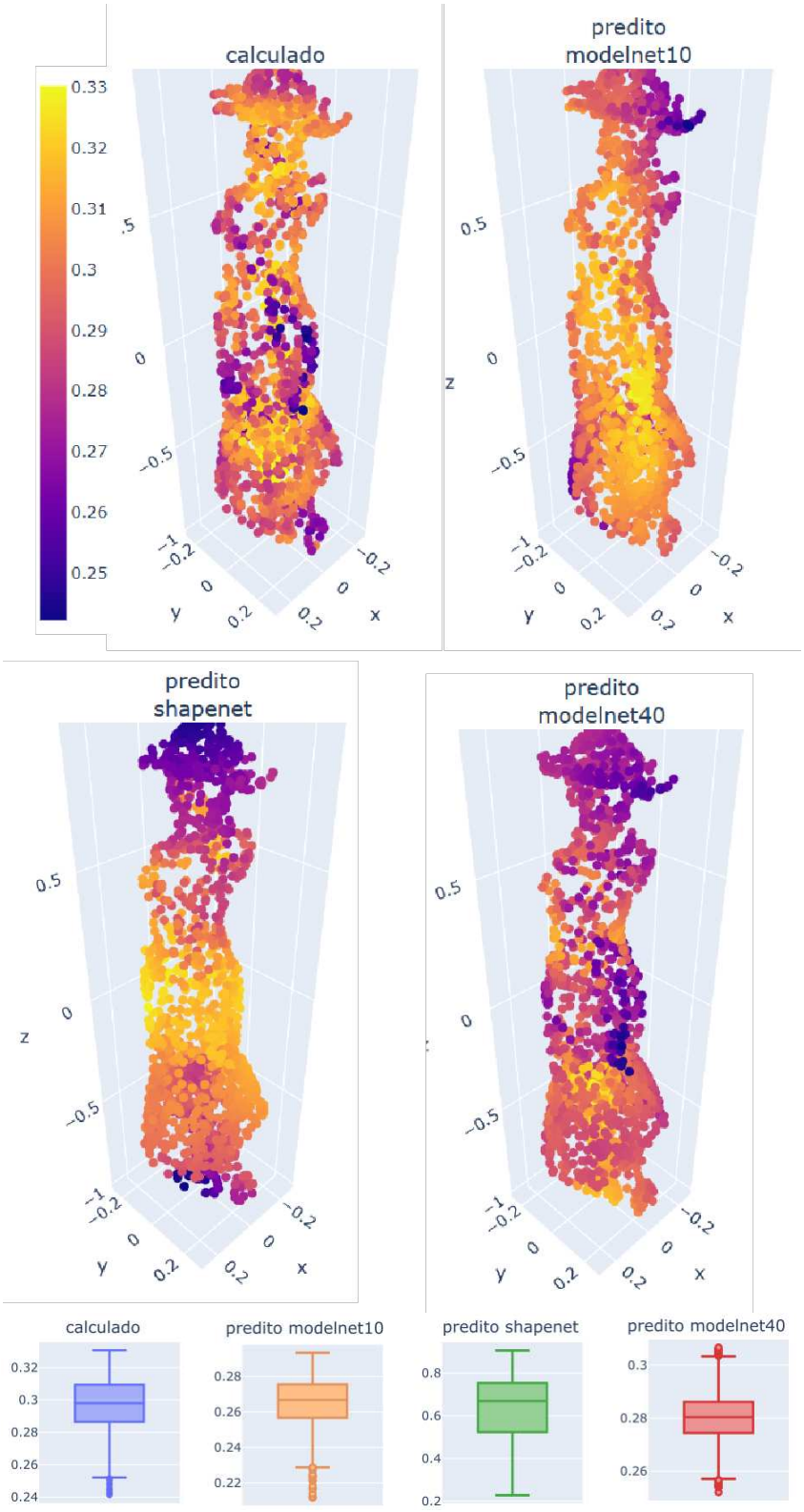
Fonte: elaborada pelo autor.

Figura 22 – Predição do objeto person003 pelos modelos



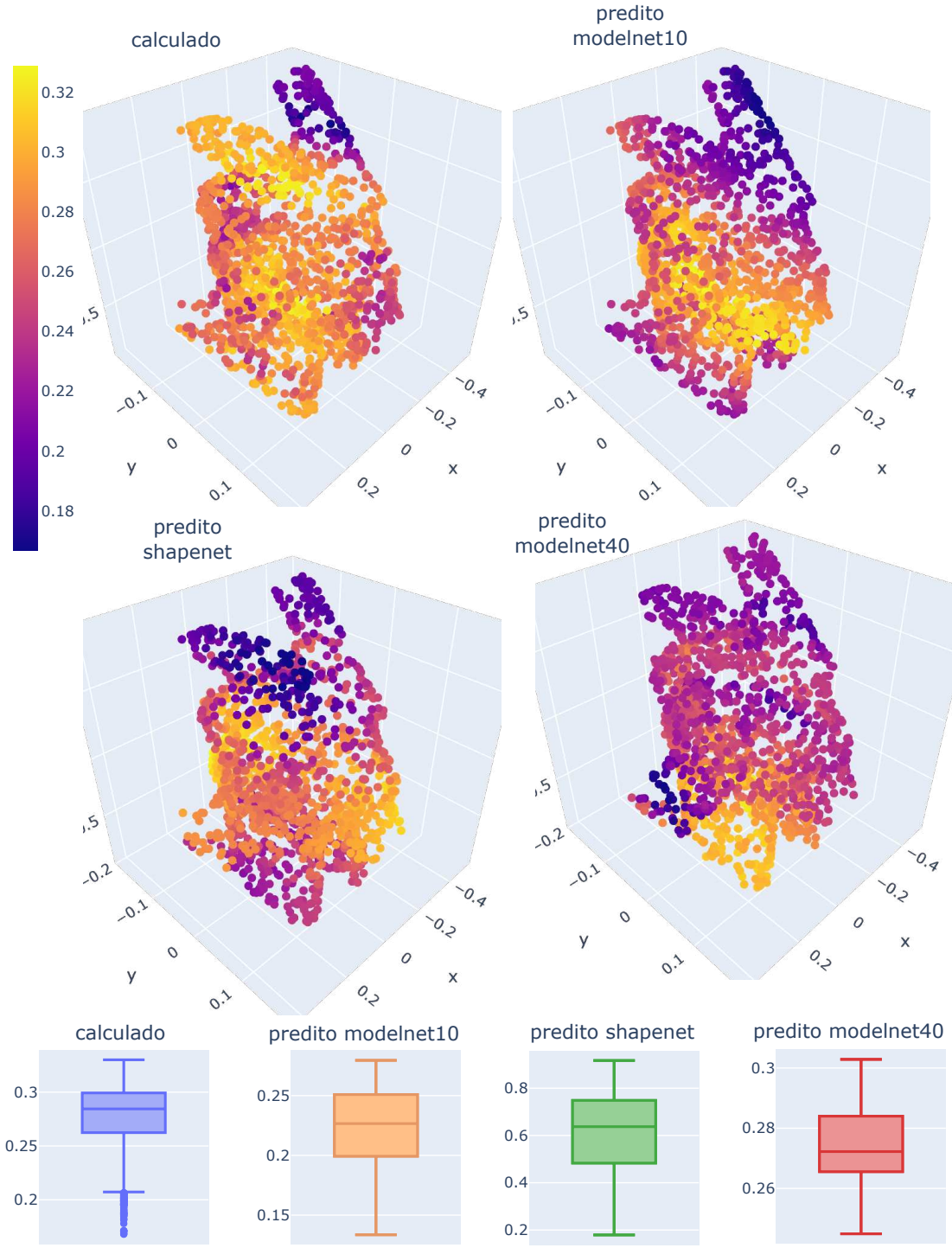
Fonte: elaborada pelo autor.

Figura 23 – Predição do objeto person004 pelos modelos



Fonte: elaborada pelo autor.

Figura 24 – Predição do objeto person005 pelos modelos



Fonte: elaborada pelo autor.

6 REFLEXÕES GERAIS

6.1 Incompletude das Funções de Perda e Métricas para avaliação para regressão ponto-a-ponto

Como vimos no referencial teórico, as funções de perda para tarefas de regressão mais comuns baseiam-se na média de alguma operação sob o erro absoluto entre o valor real e predito. Quando passamos para tarefas de regressão multissaída, ainda podemos utilizá-las, mas, quanto maior for a quantidade de regressões a serem feitas, menor fica a concisão e interpretabilidade dos resultados. Para a análise dos resultados das predições dos descritores geométricos não foi diferente, onde há mais uma variável: a topologia dos objetos em questão da predição.

Realizar inspeções sob o plot dos descritores preditos nas nuvens de pontos e compará-los com o valor real calculado preenche a lacuna de entender de como as predições estão ocorrendo. Entretanto, analisar todos os objetos do conjunto de dados dessa forma é inviável, e, ao analisarmos poucas amostras, não podemos generalizar as conclusões, além de que o processo de análise direta sobre as previsões pode conter viés.

Na falta de uma métrica específica para avaliar regressão ponto-a-ponto em nuvens de pontos, pode ser oportuno criar uma baseando-se nas necessidades da tarefa. No caso de descritores locais, que dependem da configuração dos seus pontos vizinhos, poderíamos computar alguma métrica utilizando a mesma estratégia. Seguindo essa lógica, poderíamos até conseguir uma métrica com maior interpretabilidade, entretanto, ainda estaríamos resumindo o desempenho de toda a predição de uma nuvem com milhares de pontos, e, dessa maneira, provavelmente alguma informação seria perdia. Além de que, ao avaliarmos todo um conjunto de dados, inevitavelmente temos que resumir novamente essas informações através de uma média ou mediana.

Na dificuldade de sintetizar o desempenho de uma predição ponto-a-ponto, e ainda manter alta interpretabilidade, podemos criar e utilizar várias métricas diferentes para análise. Assim, mesmo uma métrica resumindo e perdendo algum tipo de informação, haverá outra preenchendo essa lacuna, e o contrário também. Dessa maneira, podemos contornar a dificuldade da análise, mas ainda há a dificuldade de encontrar uma função de perda que melhor reflete os propósitos da tarefa.

6.2 Aprendizado de informações locais de nuvens de pontos e viés de geometria

Como vimos no Capítulo 2 de Fundamentação Teórica, as redes neurais possuem vários mecanismos para ajustar o pesos dos seus neurônios de forma a se adequar aos resultados de uma função de perda. Dessa maneira, ainda que precisemos de uma função de perda que melhor reflete os propósitos da tarefa em questão, a forma com que o modelo aprende e ajusta seus pesos depende majoritariamente da sua arquitetura, afinal, é ela que define a quantidade e a relação dos neurônios.

A PointNet foi idealizada para tarefas de classificação e segmentação por partes, tarefas estas que dependem fortemente de informação global da nuvem. Sua arquitetura contém componentes, como a T-Net, e etapas, como o *max pooling*, que tem forte efeito positivo nas tarefas citadas. Entretanto, como foi possível inferir através dos experimentos, tarefas que necessitam essencialmente de informação local podem ser mais difíceis de serem realizadas por conta da dificuldade de desassociar da rede a informação global das nuvens.

Uma das estratégias adotadas foi retirar a informação global ao deixarmos de concatenar a *global feature* com os pontos em alta dimensionalidade, mas, embora o modelo com melhor desempenho tenha sido com essa estratégia, sob a ótica das métricas utilizadas, os resultados não apresentaram mudanças significativas. Nessa perspectiva, se analisarmos com mais atenção as arquiteturas, podemos perceber que ainda há informação global nos componentes da T-Net. Além disso, as próprias redes neurais mais simples também podem ser enviesadas por conta do ajuste em função dos objetos vistos no treinamento.

Corroborando com essa análise de viés, podemos destacar o desempenho do modelo da arquitetura PointNet Parcial 64 treinada com a ModelNet40, que, dos conjuntos utilizados, é o que possui maior diversidade de classes.

6.3 Conjuntos de dados e treinamento

O conjunto de dado utilizado no treinamento de uma rede neural possui uma parcela considerável de importância no desempenho da mesma, afinal, é que ele que a rede vai aprender a executar a referida tarefa. No contexto da utilização da PointNet para predição de descritores geométricos, apesar do objetivo ser criar uma função aproximadora do cálculo dos mesmos, servindo para qualquer objeto, como não podemos desassociar o viés dos objetos vistos no treinamento pela própria natureza da arquitetura. Dessa maneira, uma das alternativas é expor

ao modelo, durante o seu treinamento, uma variedade de categorias suficientemente grande para que os casos da referida tarefa seja apenas um subdomínio do domínio solução da função aproximadora. Embora a ideia de que os casos da referida tarefa seja apenas um subdomínio do domínio solução da função aproximadora, na prática ela pode ser inviável. Para realizar essa ideia, precisaríamos treinar o modelo com diferentes tipos de nuvens, como visões parciais, tamanhos diferentes, com diversos tipos de ruídos e objetos desalinhados, e ainda teríamos que calcular e armazenar a informação dos descritores geométricos. Dessa maneira, o volume de dados pode dificultar ou até mesmo inviabilizar o processo de treinamento.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram propostos cinco arquiteturas com três estratégias diferentes para regressão ponto-a-ponto baseadas na PointNet. Apesar dos modelos desenvolvidos não terem atingido o objetivo de criar uma função aproximadora para o cálculo de descritores geométricos, o processo de modelagem e testes renderam observações importantes acerca da utilização de redes neurais em nuvens de pontos.

Ao longo dos testes foi possível perceber os pontos fortes e limitações na arquitetura na PointNet. A PointNet lida bem com tarefas de classificação e segmentação por conseguir capturar de forma eficiente informações globais das nuvens de pontos. Entretanto, para tarefas que essencialmente lidam com informações locais, o seu desempenho já não é tão satisfatório quanto nas outras tarefas citadas. Dessa maneira, com as modificações e testes propostos no presente trabalho, não foi possível utilizar a predição de descritores geométricos da PointNet para a filtragem de regiões de interesse para auxiliar o algoritmo ICP.

Por meio da T-Net e da própria natureza das redes neurais, a PointNet consegue se ajustar aos dados de treinamento e otimizar sua eficiência lidando com eles. Essa característica é bastante oportuna para a tarefa de classificação, onde se lida com tipos de objetos já conhecidos durante o treinamento, mas para tarefas que necessitam de generalização ela atrapalha.

Os modelos treinados não conseguiram generalizar a sua solução para outros tipos de objetos, mostrando a dificuldade de desassociar informações globais da arquitetura. As arquiteturas sem concatenação de informações locais e globais mostraram que a PointNet até consegue capturar informações locais processando os pontos isoladamente, mas também mostraram que suas transformações possuem um limite de significância na reprodução das informações locais. Devido ao ajuste da PointNet ao conjunto de treinamento, a utilização dela para regressão ponto-a-ponto fica limitada a aplicações que não precisem de generalizações. Ainda assim, a resolução nos resultados não é tão alta devido a tendência de realizar predições mais homogêneas pera falta de informação local para aumentar a discriminação dos pontos.

Nessa perspectiva, sugere-se para trabalhos futuros encontrar maneiras concisas, e ainda assim informativas, de se avaliar regressão ponto-a-ponto, bem como funções de perda que reflitam melhor o desempenho de redes neurais para essa tarefa. Outro ponto importante de se investigar é em maneiras de aumentar a representatividade de informações locais das nuvens ainda processando seus pontos isoladamente, pois assim outras soluções para nuvens de pontos envolvendo redes neurais poderão ser viabilizadas.

REFERÊNCIAS

AGAPAKI, E.; NAHANGI, M. **Scene understanding and model generation**. [*S. n.*], 2020. 65–167 p. Disponível em: https://doi.org/10.1016/b978-0-12-815503-5.00003-6.

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**. 2015.

CHARLES, R. L.; SU, H.; KAICHUN, M.; GUIBAS, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: . [*S. n.*], 2017. Disponível em: https://doi.org/10.1109/cvpr.2017.16.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S. l.*]: MIT Press, 2016.

HACKEL, T.; WEGNER, J. D.; SCHINDLER, K. Contour Detection in Unstructured 3D Point Clouds. In: . [*S. n.*], 2016. Disponível em: https://doi.org/10.1109/cvpr.2016.178.

HAN, S.; MAO, H.; DALLY, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In: . [*S. n.*], 2016. Disponível em: https://forum.stanford.edu/events/posterslides/DeepCompressionCompressingDeepNeuralNetworkswithPruning, TrainedQuantizationandHuffmanCoding.pdf.

HAN, X.-F.; SUN, S.-J.; SONG, X.-Y.; XIAO, G.-Q. **3D Point Cloud Descriptors in Hand-crafted and Deep Learning Age: State-of-the-Art**. 2020.

LEVOY, M.; WHITTED, T. The Use of Points as a Display Primitive. **Technical Report**, v. 85-022, 1 1985. Disponível em: https://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf.

MEKURIA, R.; BLOM, K.; CESAR, P. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. **IEEE Transactions on Circuits and Systems for Video Technology**, Institute of Electrical and Electronics Engineers, v. 27, n. 4, p. 828–842, 4 2017. Disponível em: https://doi.org/10.1109/tcsvt.2016.2543039.

MEN, H.; GEBRE, B. A.; POCHIRAJU, K. Color point cloud registration with 4D ICP algorithm. In: . [*S. n.*], 2011. Disponível em: https://doi.org/10.1109/icra.2011.5980407.

MURPHY, K. P. **Machine Learning**. [*S. l.*]: MIT Press, 2012.

NIELSEN, M. A. **Neural Networks and Deep Learning**. [*S. l.*: *s. n.*], 2015.

RUSSELL, S. D.; NORVIG, P. Artificial intelligence: a modern approach. **Choice Reviews Online**, Association of College and Research Libraries, v. 33, n. 03, p. 33–1577, 11 1995. Disponível em: https://doi.org/10.5860/choice.33-1577.

RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast Point Feature Histograms (FPFH) for 3D registration. In: . [*S. n.*], 2009. Disponível em: https://doi.org/10.1109/robot.2009.5152473.

RUSU, R. B.; COUSINS, S. B. 3D is here: Point Cloud Library (PCL). In: . [*S. n.*], 2011. Disponível em: https://doi.org/10.1109/icra.2011.5980567.

RUSU, R. B.; MARTON, Z.-C.; BLODOW, N.; DOLHA, M. E.; BEETZ, M. Towards 3D Point cloud based object maps for household environments. **Robotics and Autonomous Systems**, Elsevier BV, v. 56, n. 11, p. 927–941, 11 2008. Disponível em: https://doi.org/10.1016/j.robot.2008.08.005.

SALVI, J.; PAGèS, J. F.; BATLLE, J. Pattern codification strategies in structured light systems. **Pattern Recognition**, Elsevier BV, v. 37, n. 4, p. 827–849, 4 2004. Disponível em: https://doi.org/10.1016/j.patcog.2003.10.002.

SCHARSTEIN, D.; SZELISKI, R.; ZABIH, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In: . [*S. n.*], 2002. Disponível em: https://doi.org/10.1109/smbv.2001.988771.

SCHAUL, T.; HORGAN, D.; GREGOR, K.; SILVER, D. Universal Value Function Approximators. In: . [*S. n.*], 2015. p. 1312–1320. Disponível em: http://proceedings.mlr.press/v37/schaul15.pdf.

WEINMANN, M. **Reconstruction and Analysis of 3D Scenes**. [*S. n.*], 2016. Disponível em: https://doi.org/10.1007/978-3-319-29246-5.

WU, Z.; SONG, S.; KHOSLA, A.; YU, F.; ZHANG, L.; TANG, X.; XIAO, J. **3D ShapeNets: A Deep Representation for Volumetric Shapes**. 2015.

ZHANG, J.; YAO, Y.; DENG, B. Fast and Robust Iterative Closest Point. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society, p. 1, 1 2021. Disponível em: https://doi.org/10.1109/tpami.2021.3054619.

ZHANG, S. High-speed 3D shape measurement with structured light methods: A review. **Optics and Lasers in Engineering**, Elsevier BV, v. 106, p. 119–131, 7 2018. Disponível em: https://doi.org/10.1016/j.optlaseng.2018.02.017.

ZHANG, Z. Iterative point matching for registration of free-form curves and surfaces. **International Journal of Computer Vision**, Springer Science+Business Media, v. 13, n. 2, p. 119–152, 10 1994. Disponível em: https://doi.org/10.1007/bf01427149.

ZHANG, Z.-X.; SUN, L.; ZHONG, R.; CHEN, D.; XU, Z.; WANG, C.; QIN, C.-Z.; SUN, H.; LI, R. 3-D Deep Feature Construction for Mobile Laser Scanning Point Cloud Registration. **IEEE Geoscience and Remote Sensing Letters**, Institute of Electrical and Electronics Engineers, v. 16, n. 12, p. 1904–1908, 12 2019. Disponível em: https://doi.org/10.1109/lgrs.2019.2910546.