UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

CARLOS MATHEUS RODRIGUES MARTINS

APLICAÇÃO DE DEEP LEARNING PARA SEGMENTAÇÃO E DETECÇÃO DE DETRITOS EM NUVENS DE PONTOS

RUSSAS

2026

CARLOS MATHEUS RODRIGUES MARTINS

APLICAÇÃO DE DEEP LEARNING PARA SEGMENTAÇÃO E DETECÇÃO DE DETRITOS EM NUVENS DE PONTOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Profa. Dra. Rosineide Fernando da Paz

RUSSAS

2026

À minha família, por todo o investimento em minha educação e por me proporcionar o privilégio da dedicação exclusiva aos estudos durante boa parte da minha trajetória. À minha mãe, por todo o carinho, zelo e incentivo incondicional. Ao meu pai, por sua presença constante, que sempre me transmitiu segurança e força. E à minha namorada e futura esposa, por todo o suporte e amor que tem me oferecido.

AGRADECIMENTOS

Ao meu orientador(a), Prof(a). Dra Rosineide Fernando da Paz , pela paciência, confiança, orientação e por todo o conhecimento compartilhado durante o desenvolvimento deste Trabalho de Conclusão de Curso.

Aos laboratórios que tornaram este trabalho possível ao Laboratório de Tecnologias Inovadoras (LTI), pela disponibilização do espaço e pelo ambiente de trabalho que sustentou cada etapa desta pesquisa, e ao Laboratório de Reabilitação e Durabilidade das Construções (LAREB), pela disponibilização das nuvens de pontos LiDAR da Igreja Nossa Senhora do Rosário, em Aracati (CE), que constituíram a base empírica de todos os experimentos realizados neste trabalho.

Aos professores do campus da Universidade Federal do Ceará (UFC) - Russas, que contribuíram diretamente para a minha formação acadêmica, técnica e profissional, me preparando para os desafios da área.

Aos meus amigos e colegas de curso, em especial, pelos anos de parceria, pelas discussões técnicas, projetos compartilhados e por tornarem essa jornada muito mais leve.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Aos bibliotecários da Universidade Federal do Ceará: Francisco Edvander Pires Santos, Juliana Soares Lima, Izabel Lima dos Santos, Kalline Yasmin Soares Feitosa e Eliene Maria Vieira de Moura, pela revisão e discussão da formatação utilizada neste *template*.

Ao aluno Thiago Nascimento do curso de Ciência da Computação da Universidade Estadual do Ceará que elaborou o *template* do qual este trabalho foi adaptado para a Universidade Federal do Ceará.

À minha família. Ao meu pai, Antonio Carlos, por ser meu exemplo e me ensinar o verdadeiro valor do trabalho e do esforço na vida de um homem. À minha mãe, Maria Rodrigues, por ser a fonte de afeto, carinho e apoio constante em toda a minha trajetória. À minha irmã, Gabriele Rodrigues, por toda a ajuda e amparo que sempre me ofereceu. E a Lauana Cartaxo, o amor da minha vida, que esteve ao meu lado durante toda a graduação, sendo meu porto seguro e me ajudando a superar todos os desafios.

RESUMO

A detecção automática de fissuras em nuvens de pontos *Light Detection and Ranging* / Detecção e Medição por Luz (LiDAR) de estruturas de concreto enfrenta um desafio prático crítico: a escassez de dados rotulados, pois anotar manualmente cada ponto como fissura ou superfície normal é inviável em larga escala. Este trabalho propõe e avalia oito métodos de aprendizado de máquina para esse problema, cobrindo o espectro completo de supervisão supervisionado, semi-supervisionado e não supervisionado, todos treinados em modo global e avaliados sob o protocolo *Leave-One-Cloud-Out* (LOCO) em um *dataset* de 64 nuvens de pontos da Igreja Nossa Senhora do Rosário (Aracati-CE), digitalizadas com o escâner *Leica BLK 360*.

Os resultados demonstram que o treinamento global, modalidade em que os dados de todas as nuvens de treino são concatenados em um único conjunto, é o fator mais determinante para a detecção eficaz, superando diferenças de complexidade arquitetural. O GMMScalar, um perceptron multicamadas treinado nas 18 *features* escalares e geométricas, obteve o melhor AUROC médio ($0{,}976 \pm 0{,}023$), superando o PTv3-Binary (0,966) e o ScalarGAT (0,929). A migração do treinamento por nuvem para o modo global elevou o AUROC dos métodos GMM em até 36 pontos percentuais. A análise diagnóstica identificou dois regimes de discriminabilidade no *scalar field*, nuvens com separação clara (*gap*) e nuvens com sobreposição (*overlap*), que explicam a variabilidade dos resultados e quantificam o custo da ausência de supervisão ($\Delta\text{AUROC} = 0{,}35$).

Palavras-chave: nuvens de pontos; detecção de fissuras; aprendizado de máquina; *LiDAR*; detecção de anomalias; estudo comparativo.

ABSTRACT

The automatic detection of cracks in LiDAR point clouds of concrete structures faces a critical practical challenge: the scarcity of labeled data, since manually annotating each point as crack or normal surface is infeasible at scale. This work proposes and evaluates eight machine learning methods for this problem, covering the full supervision spectrum supervised, semi-supervised, and unsupervised, all trained in global mode and evaluated under the *Leave-One-Cloud-Out* (LOCO) protocol on a dataset of 64 point clouds from the Igreja Nossa Senhora do Rosário (Aracati, Brazil), acquired with the *Leica BLK 360* scanner.

The results demonstrate that global training, in which data from all training clouds are concatenated into a single set, is the most determining factor for effective detection, outweighing differences in architectural complexity. The GMMScalar, a multilayer perceptron trained on 18-dimensional scalar and geometric features, achieved the best mean AUROC (0.976 ± 0.023), surpassing PTv3-Binary (0.966) and ScalarGAT (0.929). Migrating from per-cloud to global training improved the AUROC of the GMM methods by up to 36 percentage points. Diagnostic analysis identified two discriminability regimes in the scalar field, clouds with clear separation (*gap*) and clouds with overlap (*overlap*), which explain the variability of the results and quantify the cost of absent supervision ($\Delta\text{AUROC} = 0.35$).

Keywords: point clouds; crack detection; machine learning; LiDAR; anomaly detection; comparative study

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AE	*Autoencoder* / Autoencoder
AP	*Average Precision* / Precisão Média
ASPP	*Atrous Spatial Pyramid Pooling* / Pooling Piramidal Espacial com Dilatação
AUROC	*Area Under the Receiver Operating Characteristic Curve* / Área sob a Curva Característica de Operação do Receptor
BCE	*Binary Cross-Entropy* / Entropia Cruzada Binária
BIM	*Building Information Modelling* / Modelagem da Informação da Construção
BLK	*Black* / Linha de equipamentos Leica BLK
CNN	*Convolutional Neural Networks* / Redes Neurais Convolucionais
CUDA	*Compute Unified Device Architecture* / Arquitetura Unificada de Dispositivos de Computação
DAGMM	*Deep Autoencoding Gaussian Mixture Model* / Modelo Profundo Autoencoder de Mistura Gaussiana
DCGAN	*Deep Convolutional Generative Adversarial Network* / Rede Generativa Adversária Convolucional Profunda
E57	Formato de arquivo ASTM E57 para dados tridimensionais
ELBO	*Evidence Lower Bound* / Limite Inferior da Evidência
EM	*Expectation-Maximization* / Maximização da Expectativa
EMA	*Exponential Moving Average* / Média Móvel Exponencial
F1	Medida F1
FID	*Fréchet Inception Distance* / Distância de Inception de Fréchet
FPFH	*Fast Point Feature Histograms* / Histogramas Rápidos de Características de Pontos
FPS	*Farthest Point Sampling* / Amostragem do Ponto Mais Distante
GAN	*Generative Adversarial Network* / Rede Adversária Generativa
GANs	*Generative Adversarial Networks* / Redes Adversárias Generativas

GAT	*Graph Attention Network* / Rede de Atenção em Grafos
GB	*Gigabyte* / Gigabyte
GMM	*Gaussian Mixture Model* / Modelo de Mistura Gaussiana
GMM-AE	*Gaussian Mixture Model Autoencoder* / Autoencoder com Modelo de Mistura Gaussiana
GMM-VAE	*Gaussian Mixture Model Variational Autoencoder* / Autoencoder Variacional com Modelo de Mistura Gaussiana
GPU	*Graphics Processing Unit* / Unidade de Processamento Gráfico
GRU	*Gated Recurrent Unit* / Unidade Recorrente com Portões
IS	*Inception Score* / Pontuação Inception
KL	*Kullback-Leibler* / Divergência de Kullback-Leibler
KPFCNN	*Kernel Point Fully Convolutional Neural Network* / Rede Neural Totalmente Convolucional de Pontos-Chave
LiDAR	*Light Detection and Ranging* / Detecção e Medição por Luz
LOCO	*Leave-One-Church-Out* / Deixar Uma Igreja de Fora
LocSE	*Local Spatial Encoding* / Codificação Espacial Local
LSTM	*Long Short-Term Memory* / Memória de Longo Curto Prazo
MAE	*Masked Autoencoder* / Autoencoder Mascarado
MLP	*Multi-Layer Perceptron* / Perceptron Multicamadas
NBR	Norma Brasileira
NLL	*Negative Log-Likelihood* / Log-Verossimilhança Negativa
P95	Percentil 95
PCA	*Principal Component Analysis* / Análise de Componentes Principais
PLY	*Polygon File Format* / Formato de Arquivo Poligonal
R-CNN	*Region-Based Convolutional Neural Network* / Rede Neural Convolucional Baseada em Regiões
RAM	*Random Access Memory* / Memória de Acesso Aleatório
ReLU	*Rectified Linear Unit* / Unidade Linear Retificada
RGB	*Red, Green and Blue* / Vermelho, Verde e Azul

ROC	*Receiver Operating Characteristic* / Característica de Operação do Receptor
RPN	*Region Proposal Network* / Rede de Propostas de Região
S3DIS	*Stanford Large-Scale 3D Indoor Spaces Dataset* / Conjunto de Dados Stanford de Espaços Internos 3D em Grande Escala
TLS	*Terrestrial Laser Scanner* / Scanner a Laser Terrestre
U-NET	Arquitetura de Rede Neural Convolucional para Segmentação de Imagens
VAE	*Variational Autoencoder* / Autoencoder Variacional
XYZ	Coordenadas espaciais nos eixos X, Y e Z

LISTA DE SÍMBOLOS

α	Taxa de aprendizado
β	Parâmetros de momentum no otimizador
B	Batch size / Tamanho do lote
C	Número de classes
d	Dimensionalidade do vetor de características
D	Discriminador em redes adversárias generativas
e_i	Erro de reconstrução para o ponto i
ε	Valor pequeno para estabilidade numérica
F	Vetor de características
G	Gerador em redes adversárias generativas
Γ	Coeficiente de reflexão
h	Função de ativação ou altura
i, j, k	Índices de iteração
K	Número de vizinhos mais próximos
λ	Parâmetro de peso para balanceamento de loss
L	Função de perda (Loss function)
L_{adv}	Perda adversarial
L_{class}	Perda de classificação
L_{recon}	Perda de reconstrução
L_{total}	Perda total combinada
N	Número total de pontos na nuvem
N_x, N_y, N_z	Componentes do vetor normal
P	Ponto na nuvem de pontos
Q, K, V	Query, Key, Value em mecanismos de atenção
R, G, B	Componentes de cor (Red, Green, Blue)
σ	Parâmetro de raio em convoluções espaciais

$\sum$	Operador de somatório
θ	Parâmetros do modelo neural
w_i	Peso atribuído à classe i
x, y, z	Coordenadas espaciais tridimensionais
$\hat{x}_i$	Valor reconstruído para o ponto i
x_i	Valor original do ponto i
y_i	Rótulo verdadeiro da classe i
$\hat{y}_i$	Probabilidade predita para a classe i
∇	Operador gradiente
$\|\cdot\|_2$	Norma euclidiana (L2)

SUMÁRIO

1 INTRODUÇÃO

A inspeção e o monitoramento de estruturas de concreto são tarefas fundamentais para a segurança de edificações, pontes e demais obras civis. Fissuras superficiais constituem um dos primeiros indicadores de degradação estrutural e, quando não detectadas a tempo, podem evoluir para falhas graves (LAXMAN *et al.*, 2023). Tradicionalmente, esse processo é realizado por técnicos de campo que percorrem as superfícies manualmente, um procedimento lento, dispendioso e sujeito a inconsistências entre inspetores (SONY *et al.*, 2019). A automação dessa tarefa, apoiada em sensores digitais e algoritmos de aprendizado de máquina, representa um caminho promissor para torná-la mais eficiente, reprodutível e escalável (JR *et al.*, 2019; FLAH *et al.*, 2021).

Entre os sensores disponíveis para inspeção estrutural, o LiDAR destaca-se por capturar a geometria tridimensional de superfícies com alta fidelidade, gerando nuvens de pontos que preservam coordenadas espaciais, intensidade de reflexão e atributos físicos da superfície (MIRZAEI *et al.*, 2022). Em particular, o *scalar field* , um atributo escalar gerado pelo *scanner* durante a varredura, apresenta valores sistematicamente distintos em regiões de fissura e em superfícies íntegras de concreto (CHEN; CHO, 2022). Explorar esse sinal físico como *feature* discriminativa é a motivação central deste trabalho.

Apesar do potencial do LiDAR, a detecção automática de fissuras enfrenta um desafio prático crítico: a escassez de dados rotulados. Anotar manualmente cada ponto de uma nuvem como “fissura” ou “normal” exige inspeção visual especializada e é inviável em larga escala, situação particularmente evidente em estruturas históricas como a Igreja Nossa Senhora do Rosário (Aracati-CE), cujas paredes de concreto apresentam padrões de fissura heterogêneos e distribuídos ao longo de extensas superfícies digitalizadas com o escâner *Leica BLK 360*. Essa limitação motiva o estudo comparativo de métodos com diferentes graus de dependência de rótulos: desde abordagens supervisionadas, que exploram ao máximo os rótulos disponíveis (LIN *et al.*, 2017; WU *et al.*, 2024), até abordagens não supervisionadas, capazes de operar sem nenhum rótulo de avaria durante o treinamento (PANG *et al.*, 2021; ZONG *et al.*, 2018; DINH *et al.*, 2017).

Este trabalho propõe e avalia oito métodos para a detecção automática de fissuras em nuvens de pontos LiDAR de estruturas de concreto, todos treinados em modo global, com dados de todas as nuvens de treino concatenadas, e avaliados sob o protocolo LOCO (*Leave-One-Cloud-Out*). A contribuição metodológica central é a demonstração de que o treinamento

global é o fator mais relevante para a detecção eficaz, superando diferenças de complexidade arquitetural: o GMMScalar, um perceptron multicamadas treinado diretamente nas 18 *features* escalares e geométricas, obteve o melhor AUROC médio ($0{,}976 \pm 0{,}023$), superando o PTv3 com *backbone* congelado pré-treinado no S3DIS (WU *et al.*, 2024) (0,966) e o ScalarGAT com atenção em grafos (VELIČKOVIĆ *et al.*, 2018) (0,929).

Os oito métodos avaliados cobrem diferentes paradigmas de supervisão: GMMScalar, GMM-AE, GMM-VAE, ScalarGAT e PTv3 (supervisionados), ScalarMAE (semi-supervisionado: pré-treino não supervisionado seguido de *fine-tuning* com rótulos (HE *et al.*, 2022; PANG *et al.*, 2022)), e NormFlow e MemBank (não supervisionados). Essa diversidade permite quantificar o custo da ausência de supervisão e identificar os limites do sinal escalar como *feature* discriminativa.

As contribuições deste trabalho são:

a) Coleta e documentação de um dataset de 64 nuvens de pontos LiDAR de estruturas de concreto, com 22 nuvens de superfície íntegra, 32 de avaria e 10 de teste;
b) Análise do *scalar field* como *feature* discriminativa, identificando dois regimes de separabilidade: nuvens com separação clara (*gap*) e nuvens com sobreposição (*overlap*) escalar;
c) Protocolo de treinamento global com avaliação LOCO, demonstrando ganho de até 36 pontos de AUROC em relação ao modo de treinamento por nuvem;
d) Benchmark comparativo de oito métodos sob protocolo LOCO global, reportando AUROC, F1 e AP para cada nuvem e em média;
e) Análise diagnóstica dos limites do *scalar field* como sinal discriminativo, com implicações para trabalhos futuros na área.

Este trabalho está organizado em sete capítulos. O presente capítulo contextualiza o problema e apresenta as contribuições. O Capítulo 2 define os objetivos geral e específicos. O Capítulo 3 apresenta a fundamentação teórica, abordando nuvens de pontos, aprendizado contrastivo, *memory banks* para detecção de anomalias, redes de atenção em grafos e autoencoders mascarados. O Capítulo 4 discute os trabalhos relacionados. O Capítulo 5 descreve o dataset, o protocolo de avaliação e todos os métodos propostos. O Capítulo 6 apresenta os resultados experimentais. Por fim, o Capítulo 7 expõe as conclusões, limitações e propostas de trabalhos futuros.

2 OBJETIVOS

2.1 Objetivo geral

Realizar um estudo comparativo de métodos de aprendizado de máquina para a detecção automática de fissuras em nuvens de pontos de estruturas de concreto, cobrindo diferentes paradigmas de supervisão, supervisionado, semi-supervisionado e não supervisionado, e avaliando cada método sob o mesmo protocolo experimental.

2.2 Objetivos específicos

- Coletar e documentar um dataset de nuvens de pontos de estruturas de concreto, contendo superfícies íntegras e superfícies com fissuras, com anotações por ponto derivadas do *scalar field*;
- Implementar e avaliar oito métodos de detecção de fissuras cobrindo o espectro completo de supervisão: GMMScalar, GMM-AE, GMM-VAE, ScalarGAT e PTv3-Binary (supervisionados), ScalarMAE (semi-supervisionado), e NormFlow e ScalarMemBank (não supervisionados);
- Definir um protocolo de avaliação uniforme baseado em LOCO (*Leave-One-Cloud-Out*) com treinamento global, garantindo comparabilidade justa entre métodos de diferentes paradigmas;
- Reportar as métricas AUROC, F1 e AP para cada nuvem de avaria do dataset e em média, permitindo análise por nuvem e análise agregada;
- Analisar os limites do *scalar field* como sinal discriminativo, identificando os dois regimes de separabilidade presentes no dataset e suas implicações para cada paradigma de supervisão.

3 FUNDAMENTAÇÃO TEÓRICA

Detectar fissuras automaticamente em nuvens de pontos LiDAR implica dominar duas áreas de conhecimento distintas: a estrutura particular desses dados tridimensionais e os paradigmas de aprendizado profundo desenvolvidos para processá-las. Nas seções iniciais deste capítulo, traça-se a trajetória histórica dos algoritmos de segmentação 3D desde *PointNet* ao *SuperPoint Transformer*, mostrando como cada proposta emergiu em resposta às limitações de sua predecessora. Em seguida, são abordados os paradigmas de baixa supervisão que fundamentam as contribuições deste trabalho: detecção de anomalias por modelagem generativa, *memory banks* contrastivos, redes de atenção em grafos e *autoencoders* mascarados. O *Point Transformer V3* (PTv3) é discutido no decorrer deste capítulo: onde faz parte da linha histórica de algoritmos e integra a base de referência da parte supervisionada nos experimentos.

3.1 Nuvens de Pontos e Representações 3D

Nuvens de pontos são coleções de amostras discretas no espaço tridimensional, em que cada ponto registra ao menos suas coordenadas (x, y, z) e, frequentemente, atributos adicionais como intensidade de retorno do laser, cor RGB ou normais de superfície Liu *et al.* (2021). A principal tecnologia de aquisição é o *scanner* LiDAR (*Light Detection and Ranging*), que emite pulsos de laser e mede o tempo de retorno para calcular distâncias com precisão milimétrica. Na modalidade de *Terrestrial Laser Scanner* / Scanner a Laser Terrestre (TLS) (*Terrestrial Laser Scanning*), o dispositivo é posicionado sobre tripés e varre o ambiente em múltiplos posicionamentos sequenciais, produzindo nuvens densas com dezenas a centenas de milhões de pontos por cena Poux e Billen (2019). Alternativas incluem a fotogrametria por imagens que reconstrói geometria 3D a partir de fotografias sobrepostas processadas por *Structure from Motion* e câmeras de profundidade baseadas em luz estruturada, mais comuns em aplicações embarcadas de menor alcance.

As nuvens de pontos apresentam três características intrínsecas que as distinguem das imagens convencionais, impondo desafios significativos ao desenvolvimento de métodos baseados em redes neurais para sua representação e processamento. A primeira é a *irregularidade*: os pontos não estão distribuídos em uma grade regular, pois a densidade varia com a distância ao *scanner* e com a geometria da superfície. A segunda é a *ausência de ordem*: não existe uma sequência canônica entre os N pontos, qualquer permutação representa a mesma geometria, o

que invalida operações que assumem um índice espacial fixo. A terceira é a *esparsidade global*: embora a nuvem possa ser localmente densa, grandes regiões do volume 3D permanecem vazias, tornando a representação volumétrica uniforme extremamente ineficiente em memória.

Essas características tornam inaplicável a abordagem direta das redes convolucionais desenvolvidas para imagens. A voxelização é a subdivisão do espaço em células cúbicas regulares a abordagem foi a primeira alternativa adotada, mas cresce cubicamente com a resolução: dobrar a precisão eleva o número de células por um fator de oito, inviabilizando cenas de escala urbana sem compressão drástica como mencionado em Alzubaidi *et al.* (2021). A projeção em múltiplas vistas 2D preserva parte das informações geométricas, mas descarta a profundidade e introduz ambiguidades em superfícies ocludidas. A limitação de ambas as estratégias motivou o desenvolvimento de arquiteturas capazes de processar os pontos diretamente, na representação bruta (x, y, z), sem conversões intermediárias.

Para avaliar e comparar essas arquiteturas, a comunidade consolidou alguns *datasets* de referência. O ModelNet40 ilustrado na Figura 1 reúne objetos 3D sintéticos para classificação é amplamente utilizado em estudos de capacidade representacional. O S3DIS cobre ambientes internos de edifícios reais com anotações semânticas ponto a ponto, tornando-se o principal *benchmark* para segmentação de interiores. O ScanNet amplia esse escopo com cenas reconstruídas por câmeras RGB-D, e o SemanticKITTI disponibiliza sequências *outdoor* de escala urbana anotadas em 28 classes.

Figura 1 – Exemplo nuvens de pontos conjunto de dados público *modelnet40*

Fonte: Elaborado pelo autor.

.

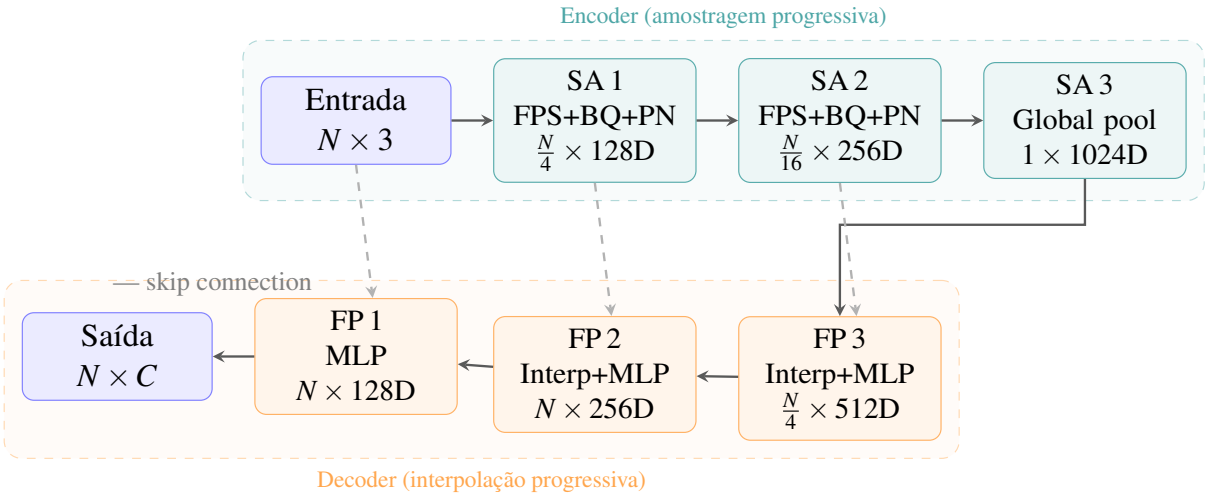
3.2 Evolução dos Algoritmos de *Deep Learning* para Segmentação de Nuvens de Pontos

O *PointNet* introduzido por Qi *et al.* (2017a), foi o primeiro algoritmo a processar nuvens de pontos diretamente nas coordenadas (x, y, z), sem convertê-las em *grids* ou *voxels*. A ideia central era aplicar um *Multi-Layer Perceptron* / Perceptron Multicamadas (MLP) compartilhado a cada ponto de forma independente e, em seguida, reunir todas as representações por meio de um *max-pooling* global simétrico que garante invariância à permutação dos pontos na entrada. Por tratar cada ponto isoladamente antes da agregação, o *PointNet* não tinha mecanismo para capturar estrutura de vizinhança: dois pontos adjacentes influenciavam a representação final apenas de forma indireta, pelo *pooling* de toda a nuvem. Essa limitação tornou-se evidente em tarefas de segmentação semântica, nas quais distinguir classes próximas exige relações locais de curta distância.

Qi *et al.* (2017b) estendeu esta abordagem para o *PointNet++*, e introduziu uma

hierarquia de *set abstractions*: pontos candidatos são selecionados por *Farthest Point Sampling* (FPS), seus vizinhos dentro de raios fixos são agrupados por *ball query*, e o *PointNet* original é aplicado localmente a cada grupo, extraindo representações progressivamente mais globais à medida que a resolução diminui ao longo do encoder (Figura 2). Essa estrutura hierárquica captava tanto detalhes de superfície em baixo nível quanto contexto semântico de alto nível, consolidando o *PointNet++* como referência dominante por vários anos. A contrapartida era o custo do FPS, de complexidade $\mathcal{O}(n^2)$ com o número de pontos, que tornava o método inadequado para nuvens com milhões de amostras.

Figura 2 – Arquitetura hierárquica do *PointNet++*: encoder com camadas SA (*Set Abstraction*) que reduzem progressivamente o número de pontos via FPS+*ball query*+PointNet local; decoder com camadas FP (*Feature Propagation*) que restauram a resolução por interpolação, conectadas ao encoder por *skip connections* (tracejado).



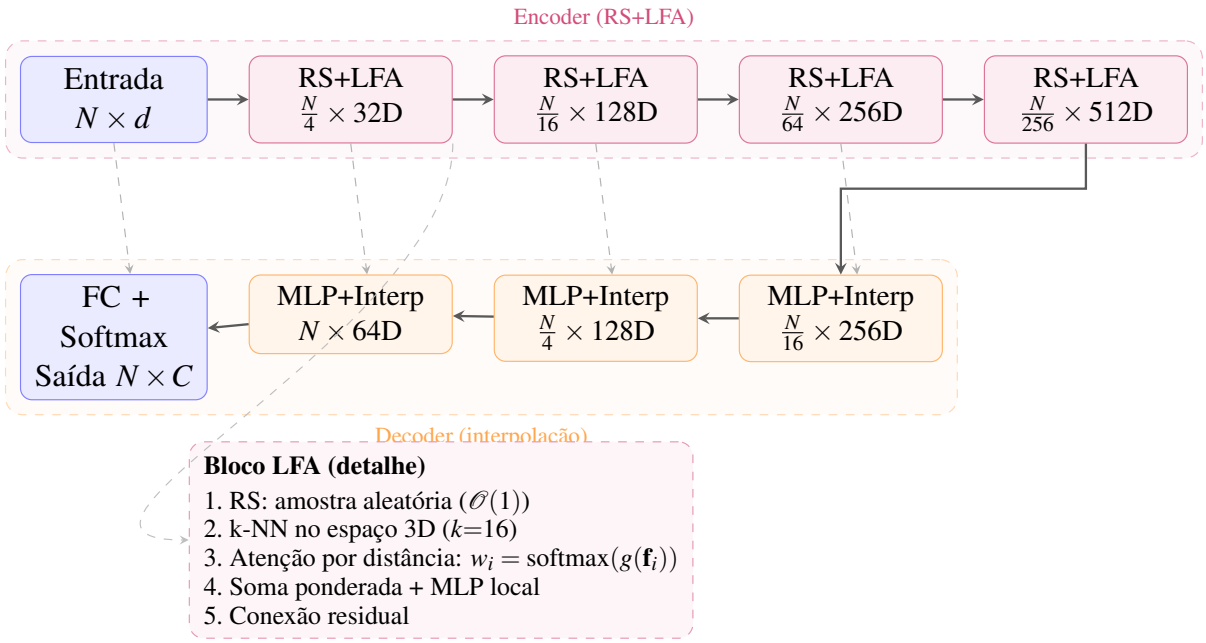
Fonte: Elaborado pelo autor.

Um caminho alternativo aberto por Wang *et al.* (2019) com o *DGCNN* (*Dynamic Graph CNN*). Em vez de fixar o grafo de vizinhança nas coordenadas 3D originais, o *DGCNN* reconstruía esse grafo dinamicamente a cada camada, no espaço de *features* aprendidas pela camada anterior. A operação central, denominada *EdgeConv*, concatenava a representação de cada ponto com a diferença em relação a seus k vizinhos mais próximos no espaço de *features* atual, capturando relações de vizinhança progressivamente mais semânticas conforme o processamento avançava pelas camadas. Com isso, um ponto fisicamente próximo de outro na entrada podia deixar de ser considerado vizinho em camadas intermediárias, se suas representações divergissem tornando o grafo de relações adaptável à tarefa em vez de prescrito pela geometria bruta.

A ideia de aplicar convolução discreta foi trazida para o domínio de pontos irregulares por Thomas *et al.* (2019) com o *KPConv* (*Kernel Point Convolution*). O operador definia um conjunto de pontos de *kernel* no espaço 3D contínuo, cujas influências sobre os pontos vizinhos decaíam com a distância de forma análoga ao filtro de uma convolução discreta, mas sem exigir regularidade de grade. Uma variante deformável permitia que as posições dos pontos de *kernel* fossem ajustadas durante o treinamento para se adaptar à geometria local, aumentando a expressividade do operador em superfícies com curvatura variável. Combinado em uma arquitetura encoder-decoder denominada *KPFCNN*, o *KPConv* alcançou resultados competitivos em *benchmarks* de larga escala *outdoor* como o Paris-Lille-3D ao mesmo tempo em que escalava para nuvens com mais de um milhão de pontos, graças à ausência do *Farthest Point Sampling* (FPS) no processo de agrupamento.

O gargalo do *Farthest Point Sampling* (FPS) permanecia, no entanto, um obstáculo prático central para aplicações em cenas de escala urbana. Hu *et al.* (2020) abordaram essa questão com o *RandLA-Net*, substituindo o FPS por amostragem aleatória simples, cuja complexidade é $\mathcal{O}(1)$ por ponto amostrado. O risco inerente era a perda de pontos informativos, já que a amostragem aleatória pode descartar regiões inteiras sem garantia de cobertura. Para compensar, os autores projetaram um módulo de atenção local que condensava, antes de cada rodada de subamostragem, a informação dos vizinhos em representações robustas ao descarte posterior. A Figura 3 ilustra essa arquitetura, cujo custo linear com o número de pontos viabilizou a segmentação semântica em cenas de escala urbana com mais de um milhão de pontos processados em tempo real.

Figura 3 – Arquitetura do *RandLA-Net*: blocos RS+LFA no encoder substituem o FPS por amostragem aleatória ($\mathcal{O}(1)$), compensados por um módulo de atenção local (LFA) que condensa a vizinhança antes de cada subamostragem; o decoder recupera a resolução por interpolação com *skip connections*.

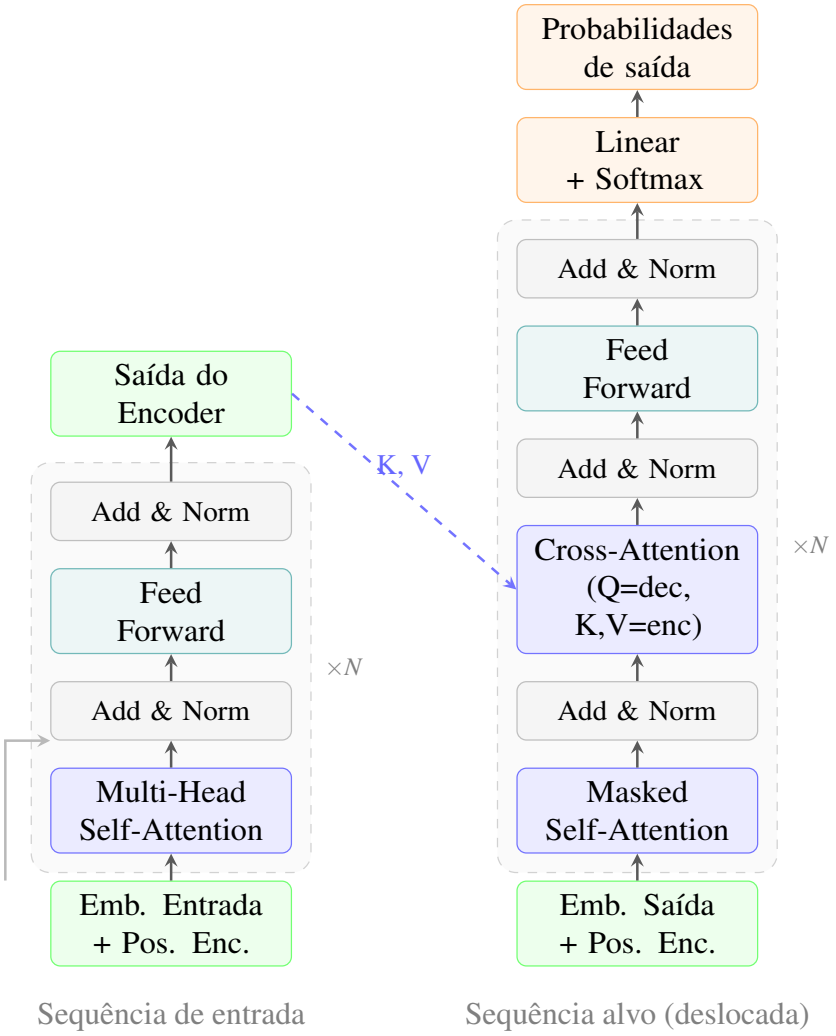


Fonte: Elaborado pelo autor.

3.2.1 Arquitetura Transformers

O mecanismo de atenção proposto por Vaswani *et al.* (2017) para o processamento de linguagem natural substituiu o processamento sequencial das redes recorrentes por um esquema paralelo: cada posição produz três vetores *query* **Q**, *key* **K** e *value* **V** e o resultado é uma combinação ponderada dos valores, cujos pesos refletem a afinidade entre consultas e chaves, como mostra a Figura 4. Após dominar a linguagem natural e, em seguida, a visão 2D com o *Vision Transformer* (ViT), esse paradigma chegou ao processamento de nuvens de pontos com o *Point Transformer*.

Figura 4 – Arquitetura do *Transformer*: codificador (esquerda) com *multi-head self-attention* e FFN empilhados N vezes; decodificador (direita) com *masked self-attention* seguido de *cross-attention* sobre as saídas do codificador ($\mathbf{K}, \mathbf{V}$) e FFN, produzindo distribuição de saída via *softmax*.



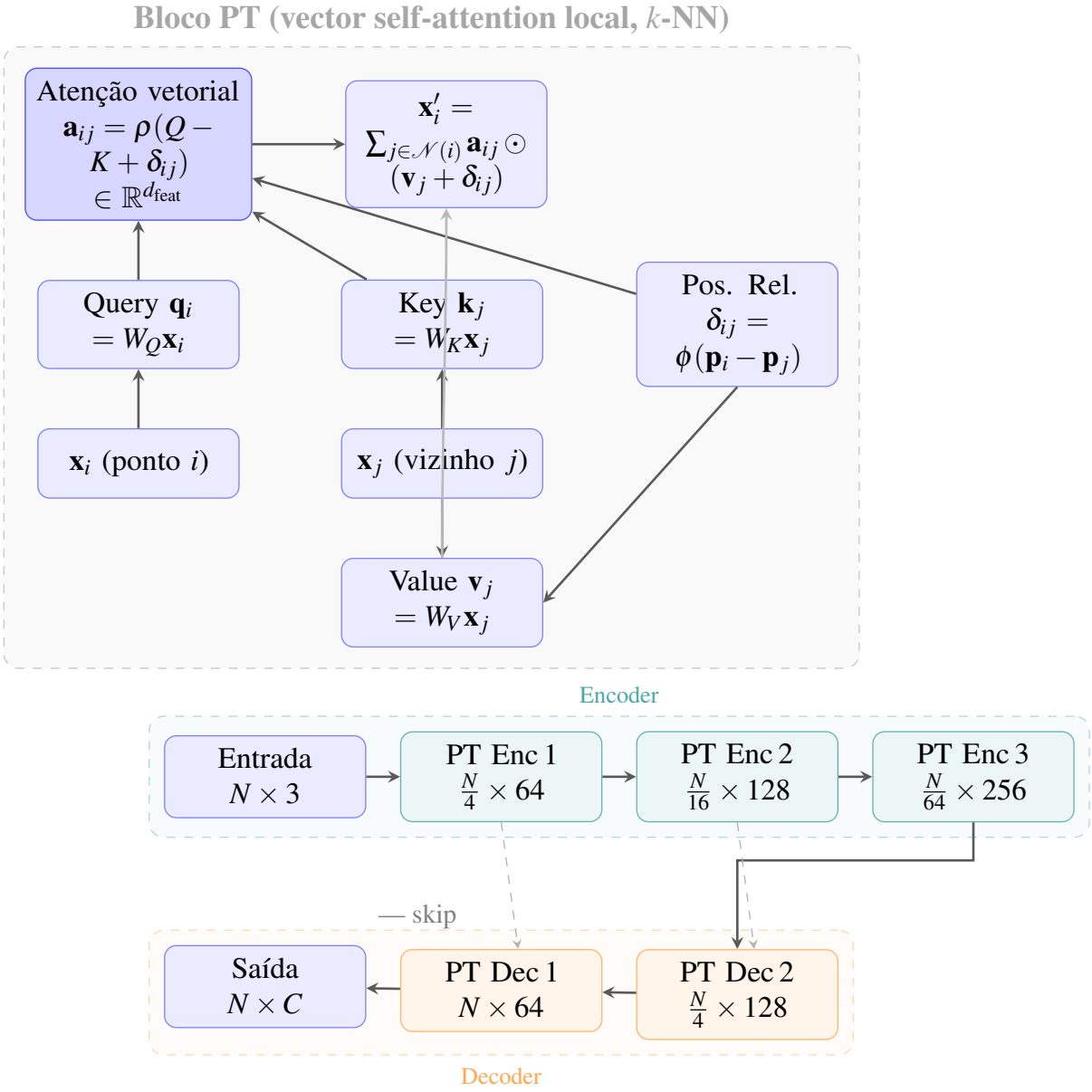
Fonte: Elaborado pelo autor.

3.2.1.1 Point Transformer

Zhao *et al.* (2021) introduziram o *Point Transformer* com uma adaptação fundamental ao domínio 3D: em vez de atenção escalar um único peso por par de elementos, o modelo aplica *vector self-attention*, em que os pesos de atenção são vetores de mesma dimensão que as *features*, permitindo que diferentes dimensões do espaço de representação sejam ponderadas de forma independente para cada vizinhança. A atenção é restrita a janelas de vizinhança local (k-NN nas coordenadas), com codificação posicional relativa adicionada às consultas, chaves e valores o que garante que a posição 3D de cada ponto influencie explicitamente a representação resultante. A arquitetura encoder-decoder composta por esses blocos de atenção, ilustrada na Figura 5,

alcançou resultados de estado da arte em ModelNet40, ShapeNet e S3DIS no momento de publicação, consolidando os *transformers* como paradigma dominante em segmentação 3D.

Figura 5 – Arquitetura do *Point Transformer*: (esquerda) bloco PT com *vector self-attention* local os pesos de atenção $\mathbf{a}_{ij}$ são vetores de mesma dimensão que as *features*, com codificação posicional relativa δ_{ij}; (direita) encoder-decoder com blocos PT substituindo o PointNet local das camadas SA/FP, com *skip connections*.



Fonte: Elaborado pelo autor.

3.2.1.2 Point Transformer V3

O *Point Transformer V3* (PTv3), proposto por Wu *et al.* (2024), representa o estado da arte em segmentação de nuvens de pontos em 2024. A principal inovação arquitetural é a substituição dos grafos k-NN explícitos pela serialização dos pontos via curvas de Hilbert: ao

ordenar os pontos por sua posição na curva, o modelo aplica atenção janelada sobre subsequências contíguas, preservando a localidade espacial sem o custo de construção e armazenamento de grafos. Essa abordagem reduz a complexidade de $\mathcal{O}(n^2)$ para $\mathcal{O}(n \log n)$ e permite processar nuvens com dezenas de milhares de pontos de forma eficiente em GPU.

O PTv3 foi pré-treinado no dataset S3DIS de segmentação de ambientes internos, o que lhe confere representações ricas de geometria 3D antes de qualquer adaptação ao domínio alvo. Neste trabalho, o modelo é submetido a *fine-tuning* para classificação binária por ponto (fissura/normal) com Focal Loss (LIN *et al.*, 2017), servindo como teto de referência supervisionado, isto é, o desempenho máximo alcançável quando rótulos ponto a ponto de avaria estão disponíveis durante o treinamento.

3.3 Detecção de Anomalias em Dados de Alta Dimensão

A detecção de anomalias (*anomaly detection*) é a tarefa de identificar observações que diferem significativamente de um padrão considerado normal, sem que exemplos de anomalias sejam necessários durante o treinamento (PANG *et al.*, 2021). Essa característica torna a abordagem especialmente adequada para cenários onde os dados de avaria são escassos ou inexistentes, exatamente o caso da inspeção de fissuras em estruturas de concreto.

A formulação clássica assume que o modelo aprende a distribuição dos dados normais $p(\mathbf{x}|\text{normal})$ durante o treinamento. Na inferência, um ponto $\mathbf{x}^*$ recebe um *score* de anomalia inversamente proporcional à sua probabilidade sob essa distribuição. Quando o *score* ultrapassa um limiar τ, o ponto é classificado como anômalo:

$$\hat{y} = \begin{cases} \text{anomalia} & \text{se } s(\mathbf{x}^*) > \tau \\ \text{normal} & \text{caso contrário} \end{cases} \tag{3.1}$$

O modelo DAGMM (*Deep Autoencoding Gaussian Mixture Model*), proposto por Zong *et al.* (2018), combina um autoencoder para redução de dimensionalidade com uma Mistura de Gaussianas (GMM) no espaço latente para modelar a distribuição dos dados normais. O *score* de anomalia é a probabilidade de pertencer ao componente "normal" da mistura. Esse paradigma inspira os métodos GMM-AE e GMM-VAE propostos neste trabalho.

3.4 Memory Banks para Detecção de Anomalias

O PatchCore, proposto por Roth *et al.* (2022) para detecção de anomalias em imagens industriais 2D, introduziu uma abordagem baseada em *memory bank*: um conjunto de representações vetoriais extraídas de imagens normais armazenadas durante o treinamento. Na inferência, o *score* de anomalia de um ponto de consulta é sua distância ao vizinho mais próximo no banco:

$$s(\mathbf{q}) = \min_{\mathbf{m} \in \mathscr{M}} \|\mathbf{q} - \mathbf{m}\|_2 \tag{3.2}$$

onde $\mathscr{M}$ é o conjunto de protótipos normais e $\mathbf{q}$ é o *embedding* do patch de consulta. Esse mecanismo é não paramétrico e não requer nenhuma suposição sobre a distribuição dos dados.

Para garantir eficiência e diversidade, o PatchCore aplica *coreset subsampling* (GONZALEZ, 1985): seleciona iterativamente os protótipos mais distantes dos já selecionados, reduzindo o banco em até 90% sem perda significativa de desempenho. O ScalarMemBank, proposto neste trabalho, estende o paradigma do PatchCore para nuvens de pontos LiDAR, adicionando normalização global do *scalar field* e *tokens* multi-escala.

3.5 Aprendizado Contrastivo

O aprendizado contrastivo é um paradigma de auto-supervisão que aprende representações forçando vetores de amostras semelhantes a ficarem próximos no espaço de *embedding*, e vetores de amostras distintas a ficarem distantes (JING; TIAN, 2021). O SimCLR, proposto por Chen *et al.* (2020), é uma das formulações mais influentes: aplica duas transformações aleatórias distintas à mesma amostra (criando dois "aumentos" da mesma instância) e minimiza a *NT-Xent Loss* (*Normalized Temperature-scaled Cross Entropy Loss*):

$$\mathscr{L}_{\text{NT-Xent}} = -\log \frac{\exp\big(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau\big)}{\displaystyle\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{3.3}$$

onde $\mathbf{z}_i$ e $\mathbf{z}_j$ são os *embeddings* dos dois aumentos da mesma amostra (par positivo), N é o tamanho do *batch* e τ é a temperatura que controla a concentração da distribuição. No ScalarMemBank, essa *loss* é usada para treinar o encoder de *tokens* multi-escala, garantindo que

patches semelhantes (normais de diferentes nuvens) produzam *embeddings* próximos no espaço latente de 32 dimensões.

3.6 Redes de Atenção em Grafos

As Redes de Atenção em Grafos (*Graph Attention Networks*, GAT), propostas por Veličković *et al.* (2018), estendem as redes convolucionais de grafos adicionando um mecanismo de atenção que pondera a contribuição de cada vizinho durante a agregação. Dado um grafo $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, a *feature* atualizada do nó i na camada l é:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathscr{N}(i)} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \right) \tag{3.4}$$

onde os pesos de atenção α_{ij} são calculados como:

$$\alpha_{ij} = \frac{\exp\big(\text{LeakyReLU}\big(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]\big)\big)}{\sum_{k \in \mathscr{N}(i)} \exp(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]))} \tag{3.5}$$

Em nuvens de pontos, o grafo é construído por k-NN nas coordenadas XYZ: cada ponto é um nó e os k vizinhos mais próximos formam as arestas. A intuição para detecção de fissuras é que pontos na borda de uma fissura têm vizinhos com *scalar field* muito diferente, o que o mecanismo de atenção pode capturar atribuindo pesos altos a essas transições bruscas.

3.7 Masked Autoencoders

O *Masked Autoencoder* (MAE), proposto por He *et al.* (2022), é um *framework* de auto-supervisão que aprende representações ricas mascarando partes da entrada e forçando o modelo a reconstruí-las. O encoder processa apenas os tokens visíveis (não mascarados) e o decoder recebe as posições mascaradas para reconstruir os tokens originais. A *loss* de pré-treino é:

$$\mathscr{L}_{\text{MAE}} = \frac{1}{|\mathscr{M}|} \sum_{i \in \mathscr{M}} \|\mathbf{t}_i - \hat{\mathbf{t}}_i\|^2 \tag{3.6}$$

onde $\mathscr{M}$ é o conjunto de tokens mascarados e $\hat{\mathbf{t}}_i$ é a reconstrução do decoder. O Point-MAE (PANG *et al.*, 2022) adapta esse paradigma para nuvens de pontos, dividindo-as

em *tokens* por FPS e aplicando mascaramento aleatório de patches. No ScalarMAE proposto neste trabalho, o mascaramento é guiado pelo *scalar field*: patches com valor escalar baixo (potencialmente fissuras) são mascarados com maior probabilidade durante o pré-treino, forçando o modelo a aprender a reconstruir regiões de avaria a partir do contexto de superfícies normais.

4 TRABALHOS RELACIONADOS

Este capítulo revisa os trabalhos mais relevantes para este estudo. Os trabalhos são apresentados em ordem crescente de proximidade com a proposta desta pesquisa: partindo de revisões sobre processamento de nuvens de pontos na construção civil, passando por abordagens de detecção de anomalias por *memory bank*, até os métodos de referência em segmentação 3D utilizados como *baselines*.

4.1 *3D point cloud data processing with machine learning for construction and infrastructure applications: A comprehensive review*

A utilização de nuvens de pontos na construção civil tem sido cada vez mais valorizada como forma de aprimorar a produtividade, qualidade e segurança ao longo do ciclo de vida dos projetos. Entretanto, conforme apontado por Mirzaei *et al.* (2022), essas nuvens carecem de informações semânticas detalhadas, o que limita o aproveitamento pleno de seus benefícios até que sejam devidamente processadas. O processamento manual de nuvens de pontos, além de dispendioso e demorado, é suscetível a erros. Para mitigar esses problemas, o artigo revisa abordagens automáticas baseadas em aprendizado de máquina que vêm sendo aplicadas em projetos de construção e infraestrutura. Tais técnicas têm o potencial de automatizar o processamento das nuvens de pontos, tornando mais eficiente a identificação e segmentação de componentes como paredes, colunas e defeitos estruturais, incluindo rachaduras.

Os autores também destacam lacunas significativas na pesquisa atual: falta de modelos robustos para dados ruidosos e ausência de informações semânticas mais ricas nos modelos automáticos. Essas lacunas têm relação direta com os desafios abordados neste trabalho, em particular a escassez de dados rotulados e a heterogeneidade das superfícies digitalizadas da Igreja Nossa Senhora do Rosário.

4.2 *Semantic Segmentation of indoor point clouds using Convolutional Neural Network*

Com o avanço do *Building Information Modelling* / Modelagem da Informação da Construção (BIM), a demanda por informações semânticas detalhadas nos modelos de edificações tem aumentado significativamente. Contudo, a geração automática de BIM semântico a partir de nuvens de pontos em edificações existentes ainda se encontra em fases iniciais. Segundo Babacan *et al.* (2017), abordagens anteriores dependem de regras codificadas manualmente, que

são limitadas em sua capacidade de generalização e tendem a falhar em diferentes contextos. A abordagem proposta consiste em redes neurais convolucionais para segmentação semântica em nível de voxel, eliminando a dependência de regras fixas. Esses métodos têm sido testados com sucesso em dados de s*canners* móveis e ambientes internos complexos, demonstrando eficácia na extração de superfícies planas em cenários desordenados.

4.3 *CrackEmbed: Point feature embedding for crack segmentation from disaster site point clouds with anomaly detection*

Este estudo propõe uma abordagem não supervisionada para a detecção de rachaduras em nuvens de pontos escaneadas a laser em cenários pós-desastres, como o terremoto no Nepal em 2015. A detecção de rachaduras é crucial para a avaliação de danos estruturais e riscos, porém, métodos tradicionais baseados em intensidade ou normais frequentemente resultam em detecções ruidosas. Métodos de aprendizado profundo oferecem maior precisão, mas necessitam de grandes conjuntos de dados anotados. Para superar essas limitações, o estudo apresenta o *framework CrackEmbed*, que utiliza uma rede neural profunda para extrair características geométricas dos pontos e um algoritmo de detecção de anomalias para segmentar as regiões danificadas. A avaliação do método em nuvens de pontos reais demonstrou que *CrackEmbed* combinado com o algoritmo *Isolation Forest* proporcionou o melhor desempenho em termos de precisão e revocação (CHEN; CHO, 2022).

4.4 *PatchCore: Towards Total Recall in Industrial Anomaly Detection*

O PatchCore, proposto por Roth *et al.* (2022), é o método que fundamenta o paradigma de *memory bank* utilizado no ScalarMemBank. O método extrai *features* de camadas intermediárias de uma rede ResNet pré-treinada para patches de imagens normais, armazena-as em um banco de protótipos e usa a distância ao vizinho mais próximo como *score* de anomalia na inferência. O *coreset greedy* (GONZALEZ, 1985) reduz o banco em até 90% preservando a cobertura geométrica do espaço de *embedding*. O PatchCore atingiu estado da arte em MVTec-AD com AUROC $> 0{,}99$. A principal diferença para o ScalarMemBank é o domínio: imagens 2D com *features* visuais versus nuvens de pontos 3D com atributos físicos do LiDAR.

4.5 *Back to the Feature: Classical 3D Features are (Almost) All You Need*

Horwitz e Hoshen (2023) demonstraram que *features* clássicas 3D (histogramas de normais, curvatura local, FPFH) combinadas com o paradigma de *memory bank* do PatchCore são suficientes para atingir desempenho competitivo em detecção de anomalias em nuvens de pontos. Esse trabalho é relevante por mostrar que *features* físicas interpretáveis, como o *scalar field* explorado neste trabalho, podem substituir *features* aprendidas por redes profundas no paradigma de *memory bank*, com menor custo computacional.

4.6 *Point Transformer V3*

O *Point Transformer V3* (PTv3), proposto por Wu *et al.* (2024), é o estado da arte em segmentação de nuvens de pontos em 2024. Em vez de construir grafos k-NN explícitos, o PTv3 serializa os pontos via curvas de Hilbert e aplica atenção janelada, permitindo processar nuvens com dezenas de milhares de pontos de forma eficiente. O modelo foi pré-treinado no dataset S3DIS e, neste trabalho, é adaptado para classificação binária (fissura / normal) por *fine-tuning* com Focal Loss (LIN *et al.*, 2017). O PTv3-Binary serve como teto de referência supervisionado, estabelecendo o desempenho máximo alcançável com rótulos ponto a ponto disponíveis.

4.7 Quadro de comparação

A Tabela 1 sintetiza as diferenças entre os trabalhos revisados e esta pesquisa, evidenciando que apenas este trabalho combina detecção de fissuras em estruturas reais com comparação sistemática de múltiplos algoritmos *state-of-the-art*.

Tabela 1 – Comparação entre os trabalhos relacionados e esta pesquisa

Trabalho	**Utilizou aprendizado de máquina para nuvens de pontos?**	**Focou em detecção de rachaduras/ avarias?**	**Aplicou segmentação semântica 3D?**	**Utilizou dados de estruturas reais?**	**Comparou múltiplos algoritmos *state-of-the-art*?**
Mirzaei *et al.* (2022)	x	x	–	x	–
Babacan *et al.* (2017)	x	–	x	x	–
Chen e Cho (2022)	x	x	x	x	–
Roth *et al.* (2022)	x	–	–	–	x
Horwitz e Hoshen (2023)	x	–	x	–	x
Wu *et al.* (2024)	x	–	x	x	–
Esta pesquisa	**x**	**x**	**x**	**x**	**x**

Fonte: Elaborado pelo autor com base na revisão bibliográfica.

5 METODOLOGIA

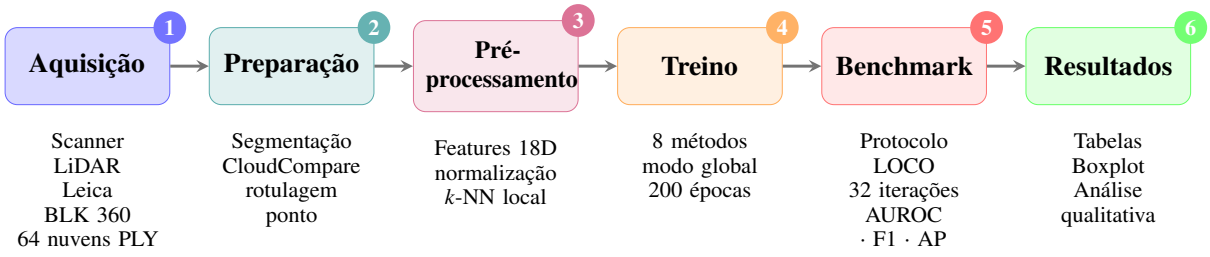
Esta seção descreve a metodologia completa do trabalho: aquisição e composição do dataset, protocolo de avaliação e os oito métodos implementados para detecção de fissuras em nuvens de pontos LiDAR. A presente pesquisa adota uma abordagem experimental e bibliográfica (GIL, 2019; LAKATOS; MARCONI, 2017).

A vertente experimental é voltada para o desenvolvimento, implementação e avaliação comparativa de métodos de aprendizado de máquina para detecção de fissuras em nuvens de pontos LiDAR. A vertente bibliográfica baseia-se na revisão crítica de trabalhos em segmentação 3D, detecção de anomalias e aprendizado contrastivo aplicados à engenharia civil.

A hipótese central deste trabalho é que o treinamento global, realizado pela agregação de amostras provenientes de múltiplas nuvens de pontos, produz modelos mais robustos e generalizáveis do que abordagens treinadas individualmente por nuvem, independentemente do paradigma de supervisão empregado.

A Figura 6 sintetiza as seis etapas da metodologia, desde a aquisição das nuvens de pontos até a análise dos resultados.

Figura 6 – Visão geral da metodologia: seis etapas desde a aquisição LiDAR até os resultados do benchmark comparativo.



Fonte: Elaborado pelo autor.

5.1 Aquisição dos Dados

A coleta de dados pode ser realizada por meio de nuvens de pontos tridimensionais adquiridas com o uso de sensores LiDAR e *scanners* 3D (POUX; BILLEN, 2019). Esses dispositivos capturam a geometria de ambientes construídos, proporcionando informações espaciais detalhadas que são utilizadas na segmentação de superfícies verticais (paredes).

Segundo o autor Vieira *et al.* (2024) a nuvem de pontos utilizada neste trabalho foi digitalizada usando a tecnologia TLS com um dispositivo *Leica BLK 360*, na digitalização da Igreja, foram escaneadas 106 cenas internas e externas, com um total de 81 pontos internos e 25

externos, onde a primeira parte do processo durou cerca de 8 horas para ser concluída. O autor Vieira *et al.* (2024) menciona o uso do software *Leica Cyclone REGISTER 360* para realizar a ligação entre as nuvens de pontos parciais, a limpeza de dados brutos e ao final sendo exportada no formato *.E57*.

5.2 Preparação do Dataset

O dataset foi construído a partir das nuvens de pontos da Igreja Nossa Senhora do Rosário, segmentadas manualmente para isolar regiões de parede. Ao todo, o dataset compreende 64 nuvens divididas em três grupos, conforme a Tabela 2.

Tabela 2 – Composição do dataset de nuvens de pontos LiDAR

Grupo	**Quantidade**	**Uso**	**Rótulos**
Normal (sem fissuras)	22	Treino (todos os métodos)	Não necessário
Avaria (com fissuras)	32	Avaliação LOCO	Ponto a ponto
Teste	10	Inferência final	Não disponível
Total	**64**	–	–

Fonte: Elaborado pelo autor.

Em média, $\approx 7\%$ dos pontos de cada nuvem de avaria pertencem a regiões de fissura. Esse desbalanceamento elevado motivou o uso de métricas baseadas em *precision-recall* (F1 e AP) além do AUROC.

5.2.1 Segmentação e Recorte Manual

A partir da nuvem completa da Igreja Nossa Senhora do Rosário exportada em formato E57, as regiões de parede foram recortadas manualmente com o software *Autodesk ReCap* (AUTODESK, 2025), isolando superfícies verticais planas. Cada recorte resultante constitui uma nuvem de pontos individual no formato PLY, preservando os atributos originais do escâner: coordenadas XYZ, cores RGB e o *scalar field*. Os vetores normais não fazem parte da exportação padrão do Leica BLK 360, sendo estimados posteriormente pela biblioteca Open3D durante a extração de *features*.

Figura 7 – Nuvem de pontos da Igreja Nossa Senhora do Rosário (Aracati-CE)

Fonte: Autor

5.2.2 Processo de Rotulagem

Para cada nuvem de avaria, o rótulo por ponto foi criado identificando visualmente, no software CloudCompare (CLOUDCOMPARE, 2025), o intervalo $[x_1, x_2]$ do *scalar field* que corresponde às regiões de fissura (Figura 8). Pontos com *scalar field* nesse intervalo recebem rótulo 1 (fissura); os demais recebem rótulo 0 (normal). Esse processo implica uma dependência circular: qualquer método que use o *scalar field* terá correlação estrutural com o rótulo.

Esse processo implica uma dependência estrutural: qualquer método que utilize o *scalar field* diretamente como *feature* terá correlação intrínseca com o critério de rotulagem, o que pode inflar o desempenho medido. Essa característica não constitui, tecnicamente, vazamento de informação no sentido clássico (*label leakage*), uma vez que o *scalar field* reflete propriedades físicas reais da superfície capturadas pelo sensor LiDAR.

No entanto, ela implica que os resultados obtidos por métodos dependentes desse sinal podem não generalizar para outros sensores LiDAR com formas distintas de geração do *scalar field*. A análise dos dois regimes de discriminabilidade (Seção 6.3) e as limitações formais do trabalho (Seção 7) aprofundam essa discussão.

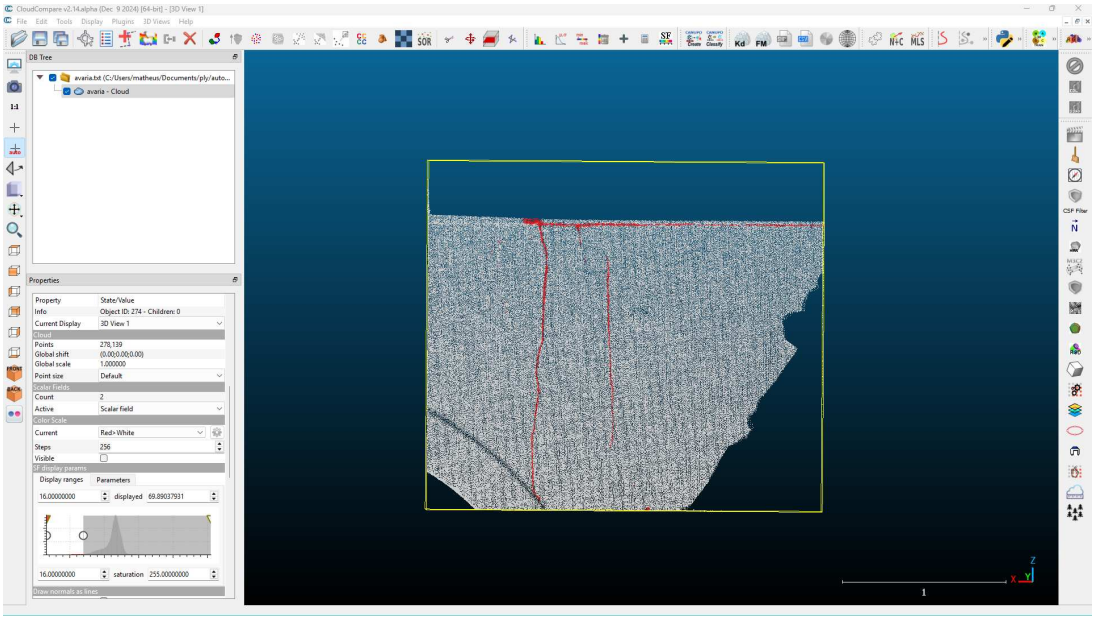


Figura 8 – Seleção do intervalo do *scalar field* correspondente a fissuras no CloudCompare. O histograma 1D permite identificar o limiar $[x_1, x_2]$ que separa as regiões de avaria.
Fonte: Autor

5.2.3 Análise dos Dois Regimes

A inspeção dos histogramas do *scalar field* por nuvem revelou dois regimes de separabilidade que têm impacto direto no desempenho de todos os métodos:

Regime gap (15 nuvens): Existe separação clara entre os valores do *scalar field* dos pontos de fissura e dos pontos normais. Qualquer método baseado no *scalar field* tende a atingir AUROC $\approx 1{,}0$ nessas nuvens.

Regime overlap (17 nuvens): Os valores do *scalar field* de fissura e normais se sobrepõem, tornando o sinal escalar não discriminativo. Nesses casos, somente métodos com acesso a *features* geométricas ou rótulos de treino conseguem alguma discriminabilidade.

A classificação de uma nuvem em um dos dois regimes é determinada pelo índice de separabilidade escalar Δ_μ, definido como a diferença entre as médias do *scalar field* das classes normal e fissura, normalizada pelo desvio padrão conjunto:

$$\Delta_\mu = \frac{|\mu_{\text{normal}} - \mu_{\text{fissura}}|}{\sqrt{(\sigma^2_{\text{normal}} + \sigma^2_{\text{fissura}})/2}} \tag{5.1}$$

Nuvens com $\Delta_\mu > 1{,}0$ são classificadas como regime *gap*; nuvens com $\Delta_\mu \leq 1{,}0$ são classificadas como regime *overlap*. No dataset, 14 nuvens pertencem ao regime *gap* e 18 ao regime *overlap*.

5.3 Pré-processamento

O pipeline de pré-processamento transforma as nuvens de pontos brutas no formato PLY em vetores de *features* prontos para os modelos. O processo é composto por quatro etapas sequenciais, descritas a seguir.

5.3.1 Vetor de Features

Cada ponto da nuvem é representado por um vetor de 18 dimensões, extraído a partir dos atributos do arquivo PLY e de características locais calculadas por vizinhança k-NN. A Tabela 3 descreve os atributos.

Tabela 3 – Vetor de features de 18 dimensões por ponto

Índices	**Atributo**	**Origem**
0–2	Coordenadas XYZ normalizadas	PLY (normalizado por nuvem)
3–5	Cores RGB normalizadas	PLY ($\div 255$)
6–8	Vetores normais (N_x, N_y, N_z)	PLY (estimado por Open3D)
9	*Scalar field* normalizado	PLY + normalização global
10	Curvatura local	Calculado por k-NN ($k = 20$)
11	Rugosidade de superfície	Calculado por k-NN
12–14	*Features* geométricas locais (PCA)	Autovalores da cov. local
15	Densidade local (pontos em raio r)	Calculado por k-NN
16	Z-score do *scalar field*	Calculado por nuvem
17	Gradiente local do *scalar field*	Calculado por k-NN

Fonte: Elaborado pelo autor.

O vetor de 18 dimensões reúne quatro categorias de atributos, cada uma cobrindo um aspecto distinto da superfície. As coordenadas XYZ e as cores RGB fornecem o contexto espacial e visual de cada ponto. Os vetores normais, estimados por PCA local, descrevem a orientação da superfície ao redor do ponto, sendo sensíveis a descontinuidades angulares nas bordas de fissura.

O núcleo discriminativo do vetor está nos atributos escalares (índices 9, 16 e 17) o *scalar field* normalizado globalmente, seu *z-score* por nuvem e seu gradiente local capturam, em granularidades diferentes, a intensidade de retorno do laser sobre superfícies degradadas.

Por fim, as medidas de vizinhança (curvatura, rugosidade, autovalores da covariância local e densidade) caracterizam a micro-geometria da superfície ao redor de cada ponto, tornando o vetor discriminativo mesmo nas nuvens de regime *overlap*, onde o sinal escalar absoluto não é informativo.

5.3.2 Normalização Global do Scalar Field

A normalização padrão (min-max por nuvem) destrói o sinal absoluto do *scalar field*: em nuvens de avaria, os pontos de fissura têm valor escalar baixo em escala global, mas podem ter qualquer valor em escala por-nuvem. Para preservar esse sinal, aplica-se uma normalização global com os percentis P_1 e P_{99} das nuvens normais de treino:

$$s_i^* = \frac{s_i - g_{\text{lo}}}{g_{\text{hi}} - g_{\text{lo}} + \varepsilon} \tag{5.2}$$

onde $g_{\text{lo}} = P_1$ e $g_{\text{hi}} = P_{99}$ são calculados sobre todas as nuvens normais. Com essa normalização, os pontos normais ficam majoritariamente em $[0, 1]$, enquanto pontos de fissura (com escalar baixo) tendem a receber valores próximos de zero ou negativos.

5.3.3 Extração do Vetor de Features

Cada PLY é processado pela biblioteca Open3D (ZHOU *et al.*, 2018) para produzir o vetor de *features* 18D descrito na Seção 5.3.1. As etapas são:

1. estimação de normais por PCA local ($k = 20$ vizinhos) os PLYs não incluem normais do escâner, portanto esse passo é sempre executado;
2. cálculo de curvatura local, rugosidade e densidade via vizinhança k-NN;
3. normalização global do *scalar field* com os percentis P_1 e P_{99} calculados sobre as nuvens normais de treino;
4. cálculo do *z-score* e do gradiente local do *scalar field*.

5.3.4 Armazenamento

Os vetores de *features* e rótulos são persistidos em arquivos `.npz` (NumPy) no *Google Drive* (GOOGLE, 2025), garantindo acesso centralizado nas etapas de treinamento executadas no Google Colaboratory. Cada arquivo contém a matriz de *features* $(N \times 18)$ e o vetor de rótulos $(N,)$, onde N é o número de pontos da nuvem.

5.4 Treinamento e Modelos

5.4.1 Processo de Treinamento

O treinamento de cada método segue a configuração de hiperparâmetros descrita nas respectivas seções deste capítulo. A maioria dos métodos supervisionados usa 200 épocas com *mini-batches* de 4096 pontos e otimizador AdamW (LOSHCHILOV; HUTTER, 2019) com escalonador cosseno; exceções são o NormFlow (500 épocas) e o ScalarMAE, que divide o treino em pré-treino não supervisionado (150 épocas) seguido de *fine-tuning* supervisionado (50 épocas). Em todos os casos, o treinamento é conduzido exclusivamente com as nuvens da partição de treino da iteração LOCO corrente dados concatenados em modo global sem acesso à nuvem de teste.

5.4.2 Estratégia de Treinamento Global

O principal diferencial metodológico deste trabalho não está nos algoritmos avaliados, mas na forma como eles são treinados. Em vez de ajustar um modelo separado para cada nuvem, todos os pontos das nuvens de treino da iteração LOCO corrente são concatenados em um único conjunto de dados, expondo o modelo a padrões de fissura provenientes de superfícies com geometrias, iluminações e estados de degradação distintos.

Na prática, para cada iteração LOCO, os vetores 18D de todas as 22 nuvens normais e das 31 nuvens de avaria do treino são agrupados em uma matriz única, gerando conjuntos de treino com milhões de pontos. A proporção de pontos de fissura sobre o total permanece em torno de 7%, tratada via *pos_weight* na BCEWithLogitsLoss (Seção 5.5) ou via Focal Loss nos demais métodos supervisionados. Esse modo de treinamento reduz o *overfitting* a características idiossincrásicas de nuvens individuais, aumentando a robustez na nuvem de teste.

A comparação com o modo por nuvem (treinamento individual) é apresentada nos resultados da Seção 6.2, onde o ganho médio de 36 pontos de AUROC confirma a hipótese central do trabalho.

5.5 GMMScalar — Método de Referência Global

O GMMScalar é o método com melhor desempenho neste benchmark. Trata-se de um perceptron multicamadas (MLP) treinado diretamente no vetor de *features* 18D de todas as

nuvens de treino concatenadas, com rótulos binários (fissura/normal). A arquitetura é simples e intencional: o objetivo é estabelecer até que ponto o sinal escalar e as *features* geométricas disponíveis são suficientes para a detecção global.

A arquitetura é:

$$\text{Input}\ (18\text{D}) \rightarrow \text{Linear}(64) \rightarrow \text{ReLU} \rightarrow \text{Linear}(32) \rightarrow \text{ReLU} \rightarrow \text{Linear}(1) \rightarrow \text{Sigmoid}$$

O treinamento usa BCEWithLogitsLoss com pos_weight $= n_{\text{neg}}/n_{\text{pos}}$ (limitado a 20) para lidar com o desbalanceamento ($\approx 7\%$ de pontos de fissura). Um StandardScaler é ajustado no conjunto de treino e aplicado na inferência. O otimizador é AdamW com *learning rate* 10^{-3} e escalonador cosseno por 200 épocas com *mini-batches* de 4096 pontos.

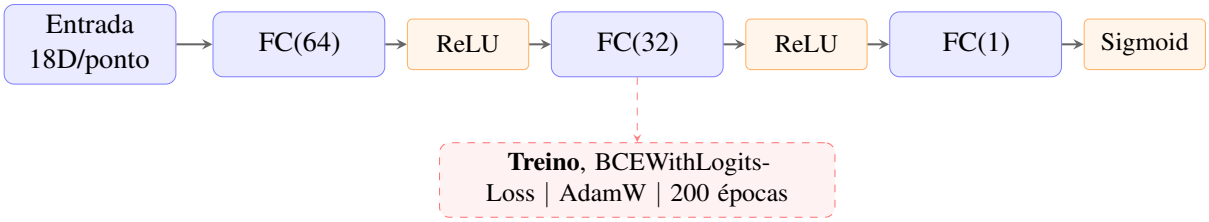


Figura 9 – Arquitetura do GMMScalar: MLP supervisionado com entrada 18D e saída binária.

5.6 ScalarMemBank

O ScalarMemBank é um detector de anomalias não supervisionado que adapta o paradigma PatchCore (ROTH *et al.*, 2022) para nuvens de pontos LiDAR, incorporando o sinal físico do *scalar field* como *feature* discriminativa.

5.6.1 Tokenização Multi-escala

A nuvem é dividida em $M = 64$ patches via *Farthest Point Sampling* / Amostragem do Ponto Mais Distante (FPS) (*Farthest Point Sampling*). Para cada centro $\mathbf{c}_i$ selecionado, dois conjuntos de vizinhança são definidos: local ($k_{\text{local}} = 32$ vizinhos) e global ($k_{\text{global}} = 96$ vizinhos). O token de cada patch é a concatenação das estatísticas dos dois níveis:

$$\mathbf{t}_i = \left[\bar{\mathbf{f}}_i^{\text{local}},\ \boldsymbol{\sigma}_i^{\text{local}},\ \bar{\mathbf{f}}_i^{\text{global}},\ \boldsymbol{\sigma}_i^{\text{global}},\ \mathbf{c}_i\right] \in \mathbb{R}^{75} \tag{5.3}$$

onde $\bar{\mathbf{f}}^{\text{local}}$ e $\boldsymbol{\sigma}^{\text{local}} \in \mathbb{R}^{18}$ são a média e o desvio padrão das features no nível local, e similarmente para o nível global. Os 3 últimos valores são as coordenadas do centro do patch.

5.6.2 *Encoder Contrastivo*

Um encoder MLP ($75 \to 256 \to 32$D) é treinado com a *NT-Xent Loss* (Seção 3.5) usando pares de tokens de patches vizinhos como exemplos positivos e tokens de patches distantes como exemplos negativos. A escolha do espaço latente de apenas 32 dimensões é intencional: forçar essa compressão obriga o encoder a aprender representações mais compactas das superfícies, o que torna a separação no espaço do memory bank mais efetiva. O encoder é treinado apenas nas nuvens normais.

5.6.3 *Memory Bank e Score de Anomalia*

Após o treino, os *embeddings* 32D de todos os patches de todas as 22 nuvens normais são armazenados no *memory bank* $\mathscr{M}$ (1408 protótipos). O *score* de anomalia de cada patch de consulta $\mathbf{q}$ é:

$$s_{\text{emb}}(\mathbf{q}) = \min_{\mathbf{m} \in \mathscr{M}} \left(1 - \frac{\mathbf{q} \cdot \mathbf{m}}{\|\mathbf{q}\| \|\mathbf{m}\|} \right) \tag{5.4}$$

O *score* final de cada ponto combina o score do memory bank com uma componente de distância de Mahalanobis no espaço do *scalar field* (colunas 9, 16, 17 do vetor de features), controlada pelo parâmetro $\alpha = 0{,}3$:

$$s_{\text{final}} = \alpha \cdot s_{\text{emb}} + (1 - \alpha) \cdot s_{\text{mah}} \tag{5.5}$$

5.6.4 *Suavização por Conectividade*

Após a pontuação, aplica-se uma etapa de suavização por difusão de vizinhança para preencher o interior das fissuras (que tendem a ser detectadas nas bordas):

$$s'_i = \beta \cdot \max_{j \in \mathscr{N}(i)} s_j + (1 - \beta) \cdot s_i \quad \text{com } \beta = 0{,}35,\ |\mathscr{N}(i)| = 15 \tag{5.6}$$

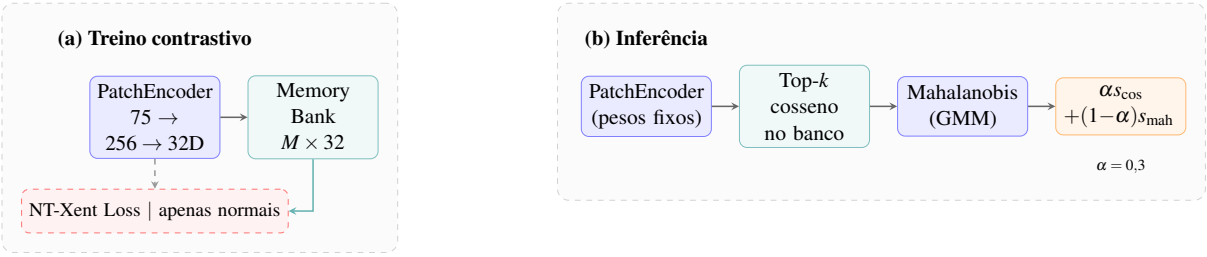


Figura 10 – Diagrama do ScalarMemBank: (a) treino contrastivo com NT-Xent e banco de memória; (b) inferência com score híbrido coseno–Mahalanobis ($\alpha = 0{,}3$).

5.7 PTv3-Binary

O PTv3-Binary é o algoritmo supervisionado mais forte. O *Point Transformer V3* (WU *et al.*, 2024) é pré-treinado no dataset S3DIS (segmentação de ambientes internos) e submetido a *fine-tuning* para classificação binária por ponto (fissura/normal) com Focal Loss (LIN *et al.*, 2017):

$$\mathscr{L}_{\text{focal}} = -\sum_i \left[(1-\hat{p}_i)^{\gamma} y_i \log \hat{p}_i + \hat{p}_i^{\gamma}(1-y_i)\log(1-\hat{p}_i)\right] \quad \gamma = 2 \tag{5.7}$$

O *fine-tuning* usa 200 épocas com otimizador AdamW (LOSHCHILOV; HUTTER, 2019). Por ser supervisionado, o PTv3-Binary acessa rótulos de avaria de outras nuvens durante o treino, o que representa uma vantagem estrutural sobre os métodos não supervisionados. Seu desempenho serve como teto de referência.

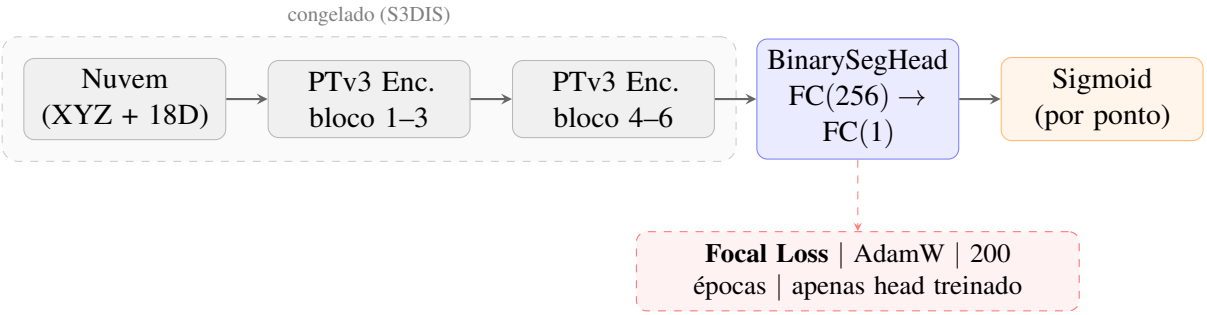


Figura 11 – Arquitetura do PTv3-Binary: *backbone* congelado pré-treinado no S3DIS mais *BinarySegHead* treinado com Focal Loss por 200 épocas.

5.8 ScalarGAT — Rede de Atenção em Grafos

O ScalarGAT aplica três camadas de Graph Attention Network (VELIČKOVIĆ *et al.*, 2018) sobre um grafo k-NN ($k = 16$) construído nas coordenadas XYZ de cada nuvem. A arquitetura é:

$$\text{Input (18D)} \rightarrow \text{GATConv(64D, 4 cabeças)} \rightarrow \text{GATConv(64D)} \rightarrow \text{GATConv(32D)} \rightarrow \text{Linear(1)} \rightarrow \text{Sigmoid}$$

O treinamento usa Focal Loss e AdamW por 200 épocas. Por usar os rótulos verdadeiros das demais nuvens de avaria durante o treino global, o ScalarGAT é classificado como supervisionado neste trabalho, assim como o GMMScalar e o PTv3 a diferença está na arquitetura, não no regime de supervisão.

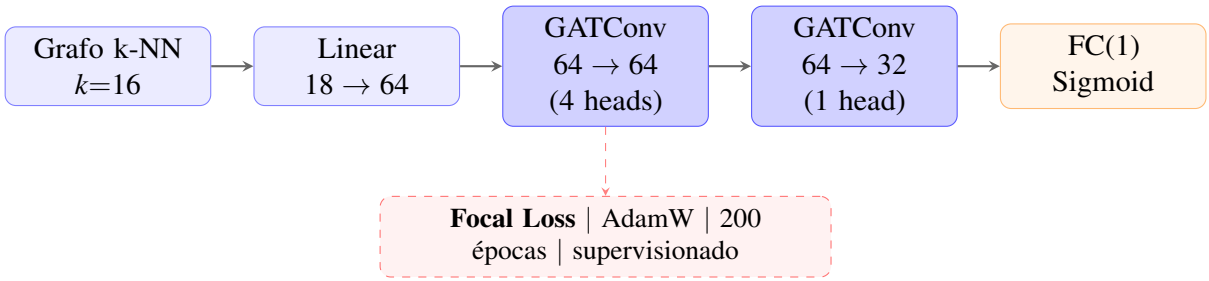


Figura 12 – Arquitetura do ScalarGAT: grafo k-NN seguido de duas camadas GATConv e saída binária por Sigmoid.

5.9 ScalarMAE — Autoencoder Mascarado

O ScalarMAE adapta o paradigma MAE (HE *et al.*, 2022; PANG *et al.*, 2022) para nuvens de pontos com mascaramento guiado pelo *scalar field*. A ideia central é aproveitar o próprio sinal físico do *scalar field* para direcionar o que o modelo precisa aprender: a probabilidade de mascarar um patch é inversamente proporcional ao seu valor escalar médio,

$$p(\text{mascarar patch } i) \propto \exp(-\bar{s}_i/\tau_s) \tag{5.8}$$

de forma que patches com escalar baixo que tendem a ser regiões de fissura são mascarados com maior frequência, forçando o encoder a reconstruí-los a partir do contexto das superfícies normais ao redor. O treinamento é dividido em duas fases bem definidas: na primeira, o modelo realiza 150 épocas de pré-treino não supervisionado, sem acesso a nenhum rótulo manual, aprendendo a distribuição das superfícies a partir da própria nuvem; na segunda fase, são realizadas 50 épocas de *fine-tuning* supervisionado com os rótulos disponíveis das demais nuvens de treino, onde o cabeçote de segmentação é ajustado com *Focal Loss* para distinguir os pontos de fissura dos normais. Essa separação em fases é o que classifica o método como semi-supervisionado: aprende geometria sem rótulos e refina com rótulos.

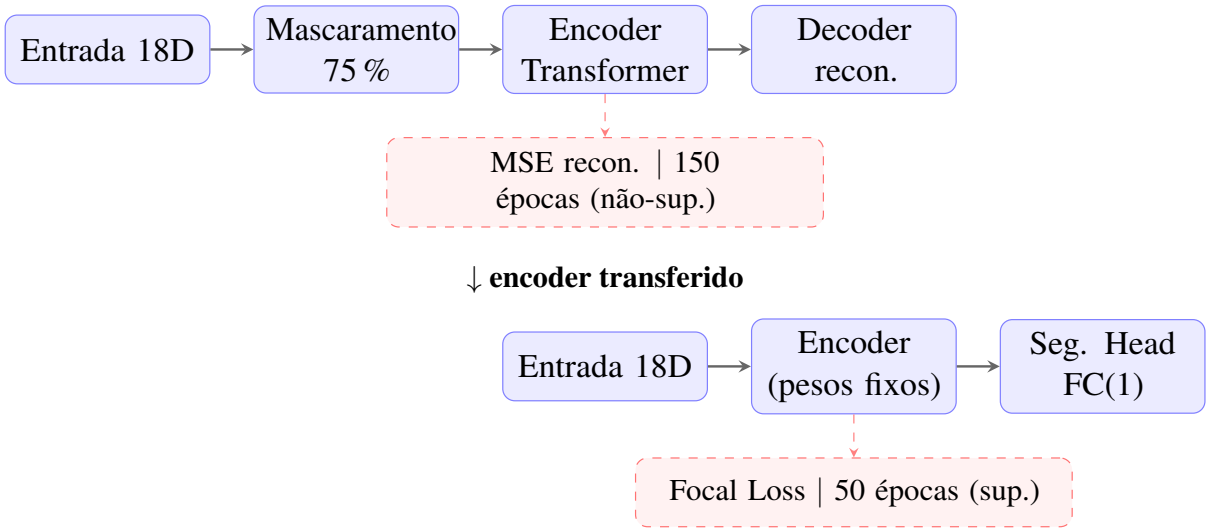


Figura 13 – Diagrama do ScalarMAE: (a) pré-treino MAE não supervisionado por 150 épocas; (b) *fine-tuning* supervisionado com Focal Loss por 50 épocas.

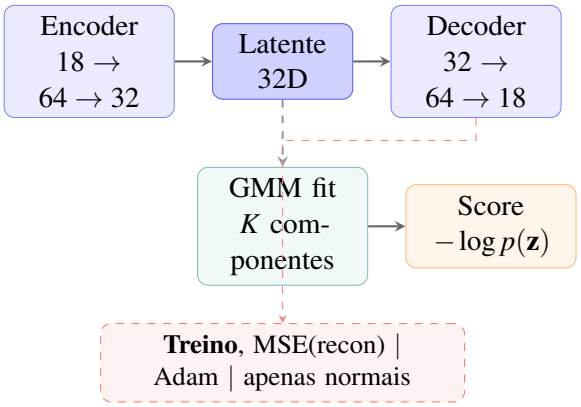
5.10 GMM-AE e GMM-VAE — Autoencoders com Mistura Gaussiana

Os métodos GMM-AE e GMM-VAE são inspirados no DAGMM (ZONG *et al.*, 2018) e operam em modo global: treinam um *encoder-classifier* em todas as nuvens de treino concatenadas, com rótulos binários (fissura/normal). Ao contrário da versão original por nuvem (que construía uma Mistura de Gaussianas $K = 2$ no espaço latente), a versão global integra a tarefa de classificação diretamente no objetivo de treinamento, via BCEWithLogitsLoss com StandardScaler e *mini-batches* de 4096 pontos.

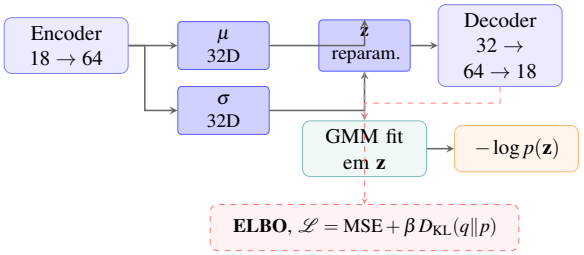
O GMM-VAE difere do AE simples ao impor uma *prior* gaussiana no espaço latente (dim=4) via *ELBO*:

$$\mathscr{L}_{\text{ELBO}} = \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})]}_{\text{reconstrução}} - D_{KL}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \tag{5.9}$$

No modo global adotado neste trabalho, os parâmetros de ambos os métodos são compartilhados entre todas as nuvens de treino, permitindo aprender padrões cross-cloud de fissura/normal que o treinamento por nuvem não captura.



(a) GMM-AE: Autoencoder + GMM no espaço latente.



(b) GMM-VAE: VAE com *prior* gaussiana (ELBO) + GMM.

Figura 14 – Arquiteturas dos métodos GMM-AE e GMM-VAE.

5.11 NormFlow — Fluxo Normalizador

O NormFlow é o método totalmente não supervisionado do benchmark: ao contrário dos demais, ele não acessa nenhum rótulo de avaria durante o treinamento nem os rótulos manuais de outras nuvens, nem os pseudo-rótulos do Otsu. O princípio é aprender a distribuição de probabilidade das superfícies *normais* e, na inferência, atribuir um *score* de anomalia a cada ponto com base em quão improvável aquele ponto é segundo o modelo treinado.

Para isso, utiliza-se a arquitetura *RealNVP* (DINH *et al.*, 2017) com 6 camadas de acoplamento afim. O RealNVP é um tipo de fluxo normalizador que aprende um mapeamento invertível $f : \mathbb{R}^d \to \mathbb{R}^d$ tal que a distribuição dos dados de treino é mapeada para uma gaussiana padrão. Dado um ponto $\mathbf{x}$ de uma nuvem de teste, o *score* de anomalia é a log-verossimilhança negativa calculada por mudança de variáveis:

$$s(\mathbf{x}) = -\log p(\mathbf{x}) = \frac{1}{2}\|f(\mathbf{x})\|^2 - \log\left|\det \frac{\partial f}{\partial \mathbf{x}}\right| \tag{5.10}$$

Valores altos de $s(\mathbf{x})$ indicam pontos improváveis sob a distribuição normal, ou seja, candidatos a fissura.

5.11.1 Features Utilizadas pelo NormFlow

Um aspecto importante da implementação é que o NormFlow *não utiliza o scalar field bruto* como feature de entrada. Essa decisão é motivada pelo regime de sobreposição: nas nuvens onde o *scalar field* de fissura e normal se sobrepõem, incluir esse atributo degradaria o modelo. Em vez disso, usam-se 16 dimensões compostas por dois grupos:

Base (8D): Colunas 10–17 do vetor de features curvatura local, densidade, variância, variação de superfície, luminância, saturação, *z-score* e gradiente do *scalar field*. O *z-score* e o gradiente capturam o contraste local do escalar sem depender do valor absoluto, evitando o problema do regime de sobreposição.

Contexto (8D): Média das mesmas 8 *features* nos $k = 15$ vizinhos mais próximos. Isso captura como cada ponto se diferencia da sua vizinhança imediata regiões de fissura tendem a ter curvatura e variância locais anômalas em relação à superfície ao redor.

O treinamento usa 200 épocas com AdamW e *mini-batches* de 4096 pontos apenas nas nuvens normais. Após o treino, aplica-se uma suavização espacial por vizinhança ($k = 15$) nos *scores* de saída, reforçando a coerência espacial das predições afinal, fissuras são regiões contíguas, e não pontos isolados no espaço.

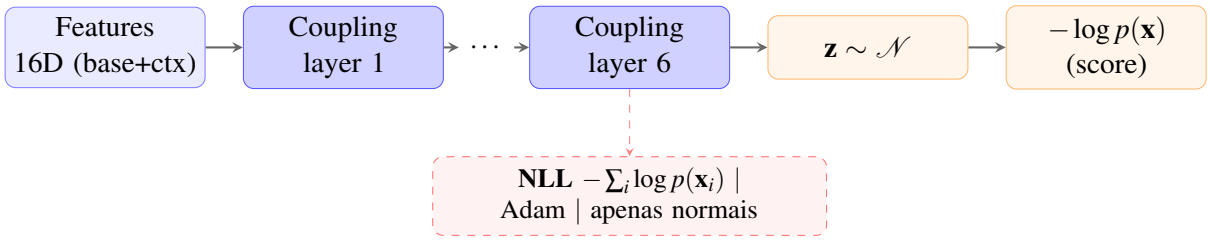


Figura 15 – Arquitetura do NormFlow: *RealNVP* com 6 camadas de acoplamento, treinado apenas em nuvens normais via NLL.

5.12 Resumo Comparativo dos Métodos

O benchmark foi construído em torno de uma questão prática: complexidade arquitetural adicional traz ganho real de desempenho nesse domínio? Para respondê-la, os oito métodos foram escolhidos de forma a cobrir os principais paradigmas de aprendizado disponíveis, desde supervisionado clássico até não supervisionado, e de MLP com poucas camadas até *Transformer* pré-treinado de última geração, conforme a Tabela 4.

Tabela 4 – Paradigmas representados pelos métodos avaliados

Método	**Paradigma representado**
GMMScalar	Supervisionado clássico (MLP)
PTv3	*Deep learning* de ponta (*Transformer*)
ScalarGAT	Modelagem relacional (grafos)
ScalarMAE	Auto-supervisão com *fine-tuning*
ScalarMemBank	*Memory bank* não supervisionado
NormFlow	Modelagem de densidade (fluxo normalizador)
GMM-AE	Compressão latente supervisionada
GMM-VAE	Modelagem probabilística (VAE)

Fonte: Elaborado pelo autor.

A Tabela 5 apresenta um resumo dos oito métodos avaliados, comparando tipo de supervisão, arquitetura e principal característica.

Tabela 5 – Resumo comparativo dos métodos avaliados para detecção de fissuras (modo global)

Método	**Supervisão**	**Arquitetura**	**Principal característica**
GMMScalar	Supervisionado	MLP 18→64→32→1	Melhor AUROC (0,976); simples e eficaz
GMM-AE	Supervisionado	Encoder-Classifier	Herança DAGMM, modo global
GMM-VAE	Supervisionado	VAE latente 4D	*Prior* gaussiana + ELBO
ScalarGAT	Supervisionado	GAT k-NN ($k = 16$)	Atenção em grafos de vizinhança
ScalarMAE	Semi-sup.	Transformer + EMA	Mascaramento guiado pelo escalar
NormFlow	Não-sup.	RealNVP 6 camadas	Densidade aprendida em nuvens normais
MemBank	Não-sup.	Memory bank + MLP	Tokens 75D + *NT-Xent loss*
PTv3	Supervisionado	Transformer serializado	*Backbone* congelado S3DIS

Fonte: Elaborado pelo autor.

5.13 Protocolo de Avaliação LOCO

O protocolo LOCO foi adotado por reproduzir um cenário operacional realista, no qual o modelo é aplicado a uma nova nuvem de pontos sem acesso prévio às suas amostras durante o treinamento. Essa escolha evita a contaminação entre treino e avaliação, que ocorreria se pontos da mesma nuvem estivessem presentes nos dois conjuntos, e ao mesmo tempo simula a aplicação real do sistema, em que o inspetor apresenta uma nuvem inédita ao modelo já treinado. O protocolo mede, portanto, diretamente a capacidade de generalização *cross-cloud*, que é o

requisito crítico para uso em inspeção estrutural.

Dado o tamanho reduzido do dataset (32 nuvens de avaria), o protocolo LOCO (*Leave-One-Cloud-Out*) é adotado para maximizar o uso dos dados:

1. Para cada nuvem de avaria $i \in \{1, \ldots, 32\}$:
   - Treino: 22 nuvens normais + 31 nuvens de avaria (exceto i)
   - Teste: nuvem i
2. Cada nuvem é avaliada exatamente uma vez, na posição de nuvem de teste.
3. As métricas finais são médias e desvios padrão das 32 rodadas.

Para os métodos supervisionados (GMMScalar, GMM-AE, GMM-VAE, ScalarGAT, PTv3), as nuvens de avaria do treino são utilizadas com rótulos binários (fissura/normal), com todos os dados concatenados em um único conjunto de treino global. Para os métodos não supervisionados (NormFlow, MemBank), apenas as nuvens normais e os pontos de rótulo 0 das nuvens de avaria são usados. O ScalarMAE realiza pré-treino não supervisionado seguido de *finetuning* com rótulos disponíveis.

5.13.1 Métricas de Avaliação

AUROC (*Area Under the ROC Curve*): Mede a capacidade de discriminação sem depender de *threshold*, sendo robusta a variações de prevalência entre nuvens. É adotada como métrica principal neste trabalho por ser independente do limiar de decisão, permitindo comparação justa entre métodos que produzem *scores* em escalas distintas. Um AUROC de 0,5 equivale a classificação aleatória; 1,0 é perfeito (FAWCETT, 2006). Para conjuntos altamente desbalanceados, a AP (*Average Precision*) é utilizada como métrica complementar por ser mais sensível ao desempenho na classe positiva (fissura).

F1-Score: Média harmônica entre precisão e revocação. O *threshold* é determinado pela estratégia de prevalência birregime: calcula-se o limiar de Otsu sobre os *scores* da nuvem de teste; se a prevalência resultante for $\leq 20\%$, o limiar de Otsu é adotado diretamente; caso contrário, multiplica-se por 0,22 para aproximar a prevalência estimada ao intervalo esperado de fissuras. Essa estratégia adapta o *threshold* à distribuição de cada nuvem sem exigir dados rotulados de validação. Adequado para datasets desbalanceados.

AP (*Average Precision*): Área sob a curva precisão-revocação, integrada sobre todos os *thresholds*. Mais informativa que o F1 para datasets com desbalanceamento severo.

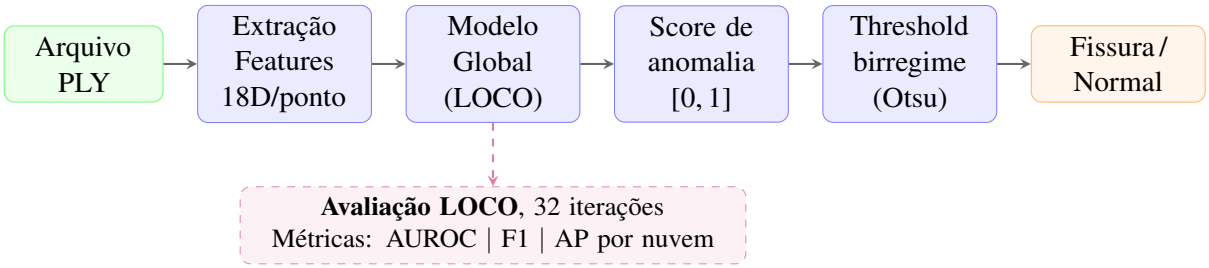


Figura 16 – Visão geral do *pipeline* metodológico: da nuvem PLY bruta ao rótulo por ponto, com avaliação LOCO em 32 iterações.

5.13.2 Registro dos Resultados

Ao final de cada iteração LOCO, são registrados o AUROC, F1 e AP da nuvem de teste corrente. Os resultados finais são médias e desvios padrão dessas métricas sobre as 32 iterações, reportados na Tabela 8 (Capítulo 6).

5.14 Inferência e Aplicação

Após o treinamento, o modelo é aplicado à nuvem de teste da iteração LOCO corrente para gerar um *score* de anomalia por ponto. Nenhuma informação sobre a nuvem de teste participa do treinamento. Os *scores* são então binarizados com o *threshold* de prevalência birregime (descrito na Seção 5.13) para produzir a segmentação final (fissura/normal). Do ponto de vista de aplicação prática, essa etapa corresponde ao uso em campo: o inspetor fornece uma nova nuvem de pontos digitalizada com o sensor LiDAR, e o modelo treinado produz um mapa probabilístico de fissuras por ponto, permitindo a identificação automática de regiões de avaria sem intervenção manual. A análise comparativa completa dos métodos, incluindo resultados quantitativos por nuvem e discussão dos dois regimes de discriminabilidade, é apresentada no Capítulo 6.

5.15 Ferramentas Utilizadas

A aplicação da metodologia sugerida utilizou uma variedade de ferramentas, que incluíram *softwares* especializados para o processamento de nuvens de pontos e ambientes de desenvolvimento focados em aprendizado profundo.

5.15.1 Software de Processamento de Nuvens de Pontos

- ***Autodesk ReCap*:** para recorte e segmentação manual das nuvens de pontos.
- ***CloudCompare*:** para seleção de valores de *scalar field* e análise geométrica.

5.15.2 Ambiente de Desenvolvimento

- **Linguagem *Python*** com as bibliotecas:
  - Open3D Zhou *et al.* (2018) para estimação de normais, cálculo de curvatura,rugosidade, densidade e extração do vetor de *features* 18D;
  - PyTorch 2.2.1 como *framework* base para *deep learning* (com suporte CUDA 11.8);
  - *PyTorch Geometric* para implementação dos modelos baseados em grafos (Scalar-GAT);
  - *Scikit-learn* para métricas de avaliação (AUROC, F1, AP) e pré-processamento (StandardScaler);
  - NumPy e pandas para manipulação e persistência dos dados.
- ***Jupyter Notebooks, Google Colaboratory, VSCode*:** para desenvolvimento e experimentação.
- **Google Drive:** para armazenamento e gestão dos *datasets*.

5.15.3 Hardware

- Plataformas com GPU (*Google Colaboratory*(GOOGLE, 2025)) para aceleração do treinamento dos modelos, como Tesla T4 ou L4.
- WSL (Windows Subsystem Linux, Subsistema do Windows para Linux) Ubuntu 24.04.4 LTS
- processador Ryzen 7 5700x
- 32 gigabytes de memória ram
- RTX 5060 TI 16 gigabytes de VRAM (video RAM)

5.15.4 Complexidade Computacional dos Métodos

A Tabela 6 apresenta uma comparação aproximada do custo computacional dos oito métodos avaliados, considerando o modo global de treinamento com o dataset completo.

Tabela 6 – Complexidade computacional dos métodos avaliados (modo global, 200 épocas)

Método	**Parâmetros treináveis**	**Inferência (s / nuvem)**
GMMScalar	3,5K	3,0
GMM-AE	1,5K	3,3
GMM-VAE	1,8K	3,7
NormFlow	19K	8,8
ScalarMemBank	57K	2,3
ScalarGAT	145K	5,7
ScalarMAE	988K	9,3
PTv3 (cabeça)	33K*	35,1

Nota: * O PTv3 utiliza um *backbone* congelado pré-treinado com $\approx$ 46M parâmetros; apenas a cabeça de segmentação binária (33K) é treinada. Os tempos de inferência foram medidos no ambiente descrito na Seção 5.15 (Google Colaboratory, GPU L4), médias sobre as 32 iterações LOCO. Os tempos de treinamento não foram registrados sistematicamente.
Fonte: Elaborado pelo autor.

A comparação entre parâmetros treináveis e tempo de inferência evidencia a assimetria entre custo e benefício: o GMMScalar, com apenas 3,5K parâmetros e 3s de inferência por nuvem, superou o PTv3, cuja cabeça treina apenas 33K parâmetros mas exige 35s por nuvem para passar pela inferência do *backbone* congelado de 46M parâmetros.

6 RESULTADOS

Os oito métodos avaliados sob protocolo LOCO produziram um resultado central: o treinamento global é o fator mais determinante para a detecção eficaz de fissuras, superando diferenças de complexidade arquitetural. Todos os experimentos seguem o protocolo LOCO (*Leave-One-Cloud-Out*), descrito na Seção 5.13. As métricas reportadas são AUROC, F1 (com *threshold* por prevalência birregime) e AP (*Average Precision*) (FAWCETT, 2006).

6.1 Configuração Experimental

Os experimentos foram conduzidos em uma estação de trabalho com GPU NVIDIA e 16 GB de memória RAM. O *framework* utilizado foi PyTorch 2.x com suporte a CUDA. A Tabela 7 apresenta os principais hiperparâmetros de cada método.

Tabela 7 – Hiperparâmetros dos métodos avaliados (modo global, 200 épocas, *mini-batch* 4096)

Método	**Hiperparâmetros principais**
GMMScalar	MLP (18→64→32→1), 200 épocas, AdamW lr=1e-3, BCEWithLogitsLoss, StandardScaler
GMM-AE	Encoder-Classifier (18→64→32→1), 200 épocas, AdamW, clip_grad, StandardScaler
GMM-VAE	VAEClassifier (18→latente 4D→1), 200 épocas, ELBO + BCE, StandardScaler
ScalarGAT	GAT k-NN ($k = 16$), 4 cabeças, 200 épocas, Focal Loss, AdamW lr=5e-4
ScalarMAE	Pré-treino MAE (150 ep.) + *finetuning* classificação (50 ep.), EMA
MemBank	$M = 64$ patches, $k_{\text{local}} = 32$, $k_{\text{global}} = 96$, encoder NT-Xent, $\alpha = 0{,}6$
NormFlow	RealNVP, 6 camadas, 200 épocas, features 16D (contextuais k=15)
PTv3	Backbone PTv3 congelado (S3DIS), BinarySegHead (512→1), 200 épocas, BCEWithLogitsLoss

Fonte: Elaborado pelo autor.

6.2 Resultados Quantitativos

A Tabela 8 apresenta o desempenho de todos os métodos avaliados, ordenados por AUROC decrescente. Os resultados são médias sobre as 32 nuvens de avaria do dataset, cada uma avaliada na posição de nuvem de teste no protocolo LOCO. A Figura 17 complementa a tabela com visualização comparativa das três métricas, e a Figura 18 exibe a distribuição AUROC

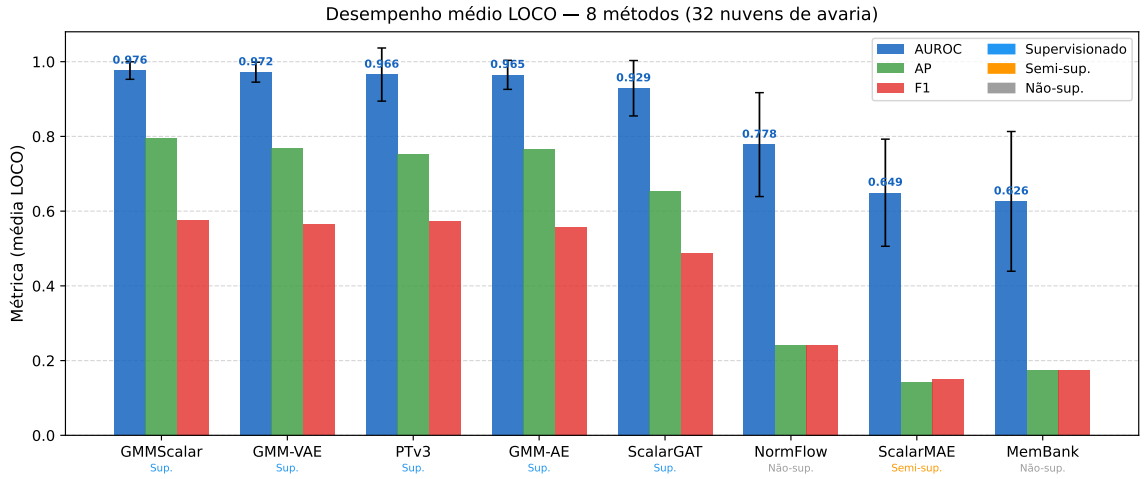
por nuvem, evidenciando a variabilidade intrínseca do dataset.

Tabela 8 – Comparação de desempenho dos métodos avaliados sob protocolo LOCO (modo global)

Método	Supervisão	AUROC	F1	AP
GMMScalar	**Supervisionado**	**0,976 ± 0,023**	**0,576 ± 0,183**	**0,796**
GMM-VAE	Supervisionado	0,972 ± 0,027	0,564 ± 0,183	0,770
PTv3	Supervisionado	0,966 ± 0,071	0,572 ± 0,213	0,752
GMM-AE	Supervisionado	0,965 ± 0,039	0,557 ± 0,185	0,766
ScalarGAT	Supervisionado	0,929 ± 0,074	0,488 ± 0,174	0,653
NormFlow	Não-sup.	0,778 ± 0,139	0,241 ± 0,152	0,241
ScalarMAE	Semi-sup.	0,649 ± 0,143	0,149 ± 0,111	0,143
MemBank	Não-sup.	0,626 ± 0,187	0,174 ± 0,126	0,175

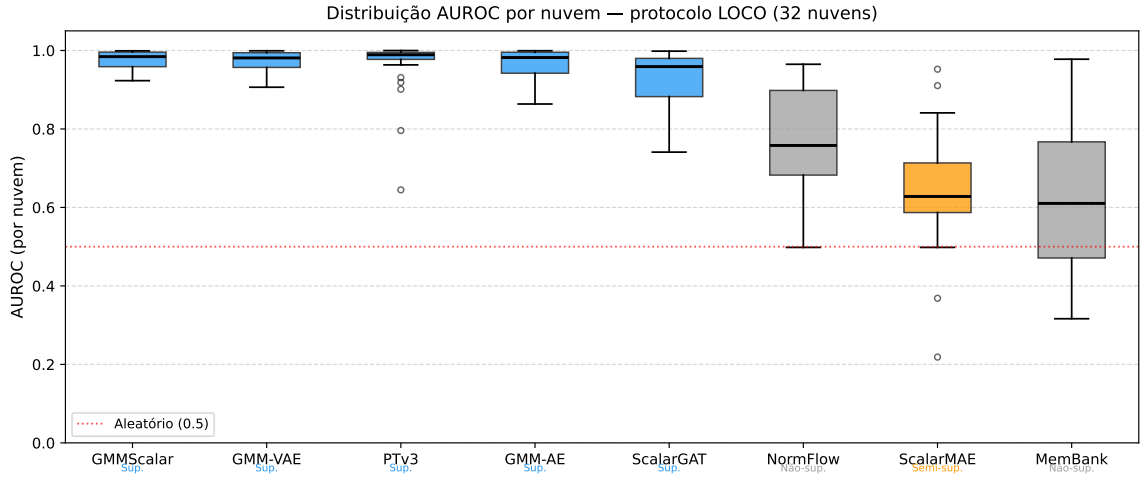
Fonte: Elaborado pelo autor.

Figura 17 – Comparação de desempenho (AUROC, AP, F1) dos oito métodos avaliados. As barras de erro no AUROC representam o desvio padrão sobre as 32 nuvens do protocolo LOCO.



Fonte: Elaborado pelo autor.

Figura 18 – Distribuição do AUROC por nuvem (protocolo LOCO, 32 nuvens). A variabilidade elevada reflete os dois regimes de discriminabilidade do *scalar field*.



Fonte: Elaborado pelo autor.

Os resultados revelam padrões consistentes. O achado mais expressivo é o impacto do treinamento global: ao serem migrados do modo por nuvem para o modo com todos os dados de treino concatenados, os métodos GMM-AE e GMM-VAE saltaram de AUROC $\approx 0{,}60$ para $\approx 0{,}96$, ganho de 36 pontos de AUROC. A arquitetura mais simples também superou as demais: o GMMScalar, um MLP de duas camadas treinado diretamente nas 18 *features* escalares, obteve o melhor AUROC médio ($0{,}976 \pm 0{,}023$), superando o PTv3 com *backbone* congelado (0,966) e o ScalarGAT com atenção em grafos (0,929).

Os métodos não supervisionados (NormFlow e MemBank) ficam significativamente abaixo do grupo supervisionado,evidenciando o valor dos rótulos de avaria no treinamento global. Os desvios padrão elevados do PTv3 ($\pm 0{,}071$) e do ScalarGAT ($\pm 0{,}074$) refletem sensibilidade a nuvens com características atípicas, enquanto o GMMScalar mantém variância consistentemente baixa ($\pm 0{,}023$) em todo o dataset.

O resultado mais expressivo do benchmark é que o GMMScalar, um MLP de duas camadas com $\approx 3{,}5$K parâmetros, superou o PTv3, uma arquitetura de *Transformer* pré-treinada em milhões de pontos do S3DIS com dezenas de milhões de parâmetros. A explicação está no próprio caráter do problema: as *features* geométricas e escalares do vetor 18D já representam a informação discriminativa de forma explícita e compacta, o que reduz o benefício da modelagem espacial implícita que o Transformer oferece.

A isso se soma o fato de que o *backbone* do PTv3 foi pré-treinado em segmentação semântica de ambientes internos (S3DIS), um domínio distante da detecção de fissuras em superfícies de parede, o que restringe a transferência de conhecimento para este contexto. Para o

dataset estudado, portanto, a qualidade e a discriminabilidade das *features* utilizadas exerceram influência maior do que a complexidade arquitetural do modelo.

Embora o PTv3 represente uma arquitetura significativamente mais sofisticada, seu desvio padrão elevado ($\pm$0,071) sugere sensibilidade heterogênea ao dataset. Esse comportamento é consistente com a hipótese de que o pré-treino no S3DIS não transfere de forma uniforme para todas as nuvens de fissura nas nuvens de regime *gap*, onde o sinal escalar já é altamente discriminativo, o PTv3 não acrescenta informação geométrica adicional.

Nas nuvens de regime *overlap*, sua geometria 3D profunda pode contribuir, como ilustrado pelo caso `avaria_36` (PTv3 = 0,987 vs. GMMScalar = 0,943, Figura 19). O resultado sugere que métodos baseados em *Transformer* 3D são mais beneficiados em cenários de regime *overlap*, enquanto métodos tabulares simples dominam no regime *gap*.

No modo global, todos os métodos supervisionados utilizam rótulos de avaria das demais nuvens durante o treino, o protocolo LOCO garante que nenhuma nuvem contamina seu próprio conjunto de avaliação. A comparação com métodos não supervisionados (NormFlow, MemBank) quantifica o custo da ausência de supervisão: ΔAUROC = 0,35 entre GMMScalar e MemBank.

6.3 Análise dos Dois Regimes de Discriminabilidade

A inspeção nuvem a nuvem revelou dois regimes distintos no dataset, determinados pela distribuição do *scalar field*:

Regime gap: Nuvens onde os pontos de fissura têm *scalar field* sistematicamente abaixo dos pontos normais, formando uma separação clara no histograma (14 nuvens). Nessas nuvens, o ScalarMemBank atinge AUROC $>$ 0,90, e o GMM-Scalar frequentemente atinge AUROC $\approx$ 1,0.

Regime overlap: Nuvens onde os pontos de fissura têm *scalar field* sobreposto ao dos pontos normais (18 nuvens). Nessas nuvens, o *scalar field* isolado não é discriminativo, e métodos que dependem exclusivamente desse sinal, como o ScalarMemBank, registram AUROC próximo a 0,5. O treinamento global com vetor de *features* completo (18D: cor, luminância, normais e geometria) permite ao GMMScalar manter AUROC acima de 0,90 na maioria dessas nuvens, evidenciando que o *scalar field* é um componente do sinal discriminativo, não o único. O PTv3-Binary complementa essa capacidade pela geometria 3D aprendida.

A Tabela 9 quantifica a diferença de desempenho entre os dois regimes, considerando

os métodos supervisionados.

Tabela 9 – AUROC médio por regime de discriminabilidade

Método	**Regime gap (14 nuvens)**	**Regime overlap (18 nuvens)**
GMMScalar	$0{,}993 \pm 0{,}007$	$0{,}963 \pm 0{,}023$
GMM-VAE	$0{,}992 \pm 0{,}008$	$0{,}957 \pm 0{,}026$
GMM-AE	$0{,}992 \pm 0{,}008$	$0{,}943 \pm 0{,}040$
PTv3	$0{,}980 \pm 0{,}029$	$0{,}955 \pm 0{,}091$
ScalarGAT	$0{,}981 \pm 0{,}018$	$0{,}888 \pm 0{,}076$
NormFlow	$0{,}914 \pm 0{,}041$	$0{,}672 \pm 0{,}083$

Fonte: Elaborado pelo autor.

A diferença entre regimes é especialmente pronunciada no NormFlow ($\Delta \approx 0{,}24$), confirmando que métodos não supervisionados dependentes do sinal escalar perdem discriminabilidade nas nuvens com sobreposição escalar. Nos métodos supervisionados, o treinamento global com *features* geométricas complementares atenua essa diferença: o GMMScalar cai apenas 3 pontos de AUROC entre os dois regimes, enquanto o PTv3 mantém desempenho estável (e até ligeiramente superior no overlap, pela sua capacidade de modelagem geométrica 3D).

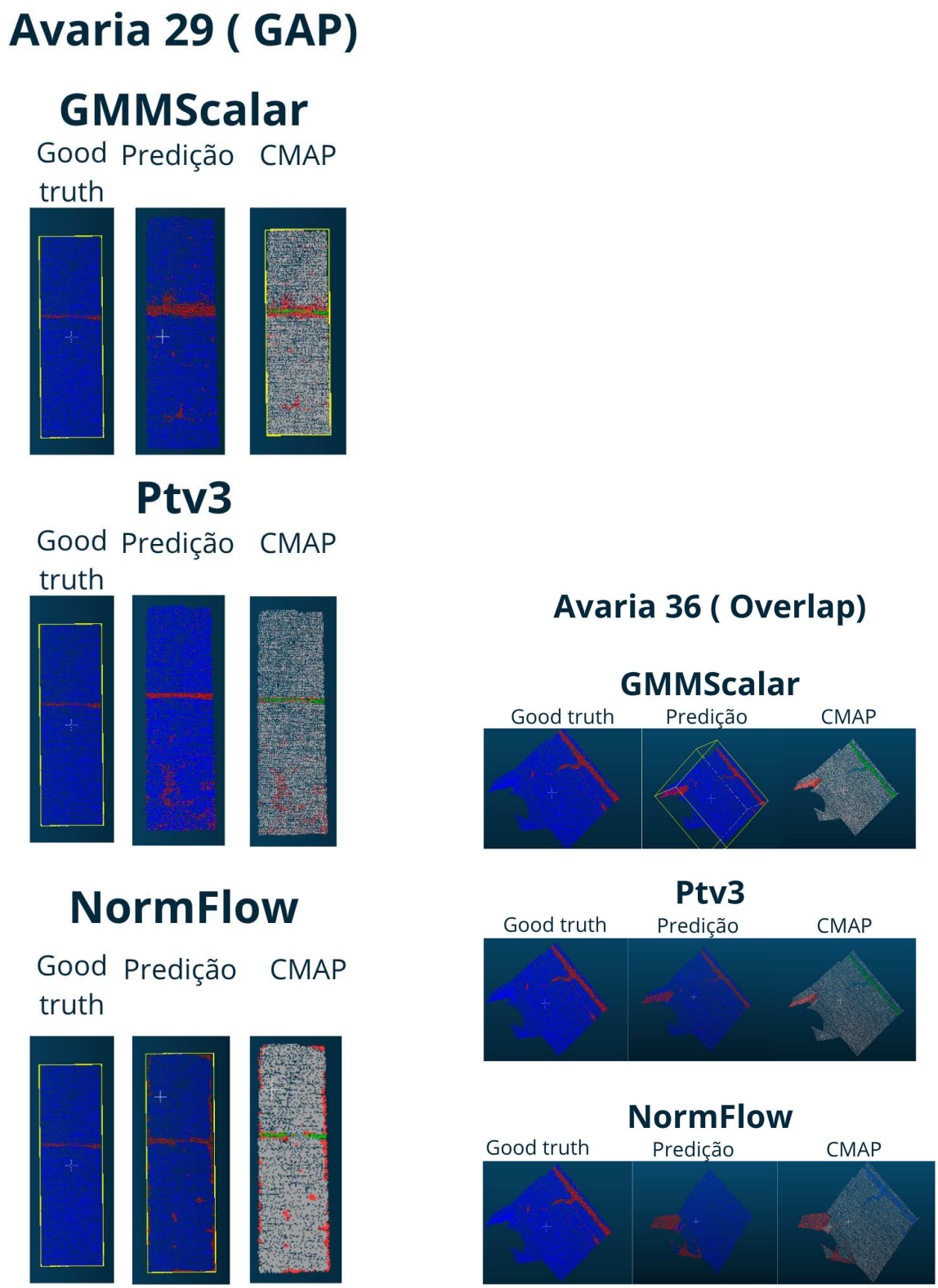
6.3.1 Custo da Ausência de Supervisão

Esse resultado não implica que os paradigmas não supervisionados sejam intrinsecamente inferiores, mas reflete uma assimetria de informação: enquanto os métodos supervisionados aprendem diretamente a fronteira decisória entre fissura e superfície íntegra a partir de rótulos de avaria de outras nuvens, os métodos não supervisionados precisam inferir essa fronteira apenas a partir da distribuição dos dados normais, sem nunca observar um exemplo positivo durante o treinamento. Em cenários sem rótulos disponíveis, o NormFlow (0,778) oferece desempenho razoável para triagem inicial de inspeção, especialmente em nuvens de regime *gap*, onde o sinal escalar já é naturalmente separável.

Essa distinção é fundamental para interpretar os desvios padrão elevados na Tabela 8. Não se trata de instabilidade do método, mas de heterogeneidade intrínseca do dataset: o método funciona bem em metade das nuvens e mal na outra metade, por razões físicas inerentes à aquisição LiDAR.

6.4 Análise Qualitativa

Os mosaicos qualitativos ajudam a entender o que os números escondem. Nas nuvens de regime *gap*, a fronteira entre fissura e superfície íntegra é nítida o suficiente para que todos os métodos supervisionados a delimitem corretamente, e a diferença de AUROC entre eles é pequena. Nas nuvens de regime *overlap*, essa fronteira se dissolve no histograma do *scalar field*, e é justamente aí que o treinamento global com o vetor 18D completo faz diferença: o GMMScalar mantém AUROC acima de 0,90 na maioria dessas nuvens porque aprendeu a usar cor, luminância e geometria como sinal complementar, enquanto métodos que dependem mais diretamente do escalar absoluto perdem discriminabilidade de forma abrupta. Os resultados qualitativos, portanto, não são apenas ilustrativos: confirmam e explicam as diferenças observadas na tabela quantitativa, incluindo o desvio padrão elevado do PTv3 e do ScalarGAT.



(a) Regime gap — avaria_29 (AUROC: GMM=0,999, PTv3=0,982, NormFlow=0,970)

(b) Regime overlap — avaria_36 (AUROC: GMM=0,943, PTv3=0,987, NormFlow=0,670)

Figura 19 – Visualização qualitativa nos dois regimes de discriminabilidade. Pontos em vermelho indicam fissura predita.

Fonte: Elaborado pelo autor.

As demais visualizações qualitativas, incluindo outras nuvens e configurações de predição, são apresentadas no Anexo A.

6.5 Análise Comparativa dos Métodos

Comparar métodos apenas por métricas médias oculta padrões importantes: métodos que convergem no ranking podem divergir radicalmente por nuvem, e métodos que parecem fracos globalmente podem ser os únicos viáveis em determinados cenários. Para cada abordagem, discutem-se a limitação central, as condições em que o método se mostra mais útil e a trajetória de evolução mais direta.

GMMScalar

A limitação estrutural do GMMScalar está na classificação ponto a ponto: cada predição depende exclusivamente do vetor 18D daquele ponto, sem modelar relações espaciais entre vizinhos além das já codificadas nas *features* locais. Fissuras são estruturas contínuas no espaço 3D, com padrões de propagação linear que o MLP não captura explicitamente.

Essa limitação não se manifesta nos resultados porque o vetor 18D já codifica o contexto local por construção: curvatura, densidade, gradiente do *scalar field* e luminância capturam, implicitamente, parte da vizinhança de cada ponto. No regime *gap*, o GMMScalar atinge AUROC $\approx 1{,}0$ com consistência; no regime *overlap*, onde o *scalar field* isolado não discrimina, mantém AUROC médio de 0,963, usando cor, luminância e normais como sinais complementares. É o único método que não colapsa em nenhum dos dois regimes, com o menor desvio padrão entre todos ($\pm 0{,}023$).

O potencial evolutivo imediato está na extensão espacial: incorporar um pós-processamento por grafos 3D, usando as predições do MLP como atributos de nó, permitiria capturar a continuidade das fissuras sem alterar o núcleo do modelo. A médio prazo, o modelo pode ser retreinado com nuvens de outras edificações sem reengenharia do *pipeline*, o que o torna o candidato mais direto a uma ferramenta de inspeção generalizável.

PTv3

O PTv3 opera diretamente sobre a geometria 3D, sem depender do vetor 18D pré-extraído, diferencial que nenhum outro método avaliado possui. Sua limitação principal é o

desalinhamento de domínio: o *backbone* foi pré-treinado no S3DIS (ARMENI *et al.*, 2016), composto por escritórios e corredores modernos, estruturalmente distinto das superfícies de alvenaria histórica do *dataset*. O reflexo disso é o desvio padrão mais alto entre os métodos supervisionados ($\pm$0,071), com desempenho instável entre nuvens.

Onde o PTv3 se sobressai é no caso mais difícil: nuvens do regime *overlap* com *scalar field* ambíguo. Na avaria_36, atingiu AUROC $= 0{,}997$ contra 0,949 do GMMScalar, o maior valor individual registrado no *benchmark*. A geometria 3D aprendida pelo *backbone* compensou, nesse caso, a ausência de sinal escalar discriminativo.

Um *fine-tuning* completo do *backbone* em dados de patrimônio histórico, em vez do congelamento adotado neste trabalho, provavelmente estenderia esse desempenho para a maioria das nuvens do regime *overlap*, e não apenas para casos isolados. Dentre os métodos avaliados, é o que apresenta maior margem de ganho com intervenção de domínio.

GMM-VAE

O GMM-VAE introduz um objetivo duplo: minimizar o erro de classificação e regularizar o espaço latente via divergência KL. A tensão entre esses dois objetivos é sua principal limitação: o termo KL pode suprimir *features* discriminativas de baixa prevalência, exatamente o caso das fissuras, que representam menos de 15% dos pontos em média. O espaço latente de quatro dimensões pode ser estreito demais para capturar a diversidade de padrões de fissura presentes nas 32 nuvens.

Ainda assim, o GMM-VAE é o segundo método em AUROC ($0{,}972 \pm 0{,}027$), próximo ao GMMScalar, com estabilidade comparável. A divergência KL, além de regularizar, oferece uma estimativa natural de incerteza por ponto: regiões onde o modelo hesita podem ser encaminhadas automaticamente para revisão humana, o que torna o GMM-VAE adequado para aplicações onde a confiança na predição é tão relevante quanto a predição em si. Técnicas de *KL annealing*, que introduzem o peso do termo KL de forma gradual durante o treino, podem reduzir essa supressão e aproximar o desempenho do GMM-VAE ao do GMMScalar.

GMM-AE

No GMM-AE, o encoder aprende a comprimir e reconstruir o vetor 18D, mas esse objetivo não está alinhado com a classificação: uma representação que generaliza bem a superfície não discrimina necessariamente fissuras de regiões normais. Apesar disso, o método

atingiu AUROC $= 0{,}965$, resultado que evidencia o peso do protocolo global sobre a arquitetura, dado que o GMM-AE com treinamento por nuvem alcançava apenas $\approx 0{,}60$.

Treinar o encoder simultaneamente para reconstruir e classificar, usando o erro de reconstrução como regularização implícita, é a extensão mais natural do modelo. Essa formulação também abre espaço para uso em cenários sem rótulos, onde o erro de reconstrução por ponto serve como *anomaly score*, abordagem consolidada em métodos como o PatchCore (ROTH *et al.*, 2022).

ScalarGAT

A limitação é de design: o grafo k-NN é construído no espaço das 18 *features* escalares, não nas coordenadas 3D. Os k vizinhos de cada ponto são os mais próximos no espaço de *features*, podendo estar fisicamente distantes na nuvem. O mecanismo de atenção opera sobre vizinhos de *features* e não sobre vizinhos geométricos reais, o que limita a capacidade de capturar a continuidade espacial das fissuras. O desvio padrão elevado ($\pm 0{,}074$) e o AUROC de 0,929 refletem essa inconsistência.

Em nuvens com textura de superfície variável, a atenção em *features* consegue identificar regiões atípicas mesmo sem referência geométrica explícita, o que representa seu cenário de uso mais favorável dentro do *dataset*. Construir o grafo pelas coordenadas 3D e usar as 18 *features* como atributos de nó corrigiria o problema central e permitiria que o mecanismo de atenção capturasse padrões de propagação das fissuras no espaço físico.

NormFlow

O NormFlow modela a distribuição esperada de superfícies normais e trata desvios como evidência de avaria, sem usar nenhum rótulo de fissura durante o treino. Essa escolha é também sua principal limitação: nas nuvens do regime *overlap*, onde as distribuições de fissura e superfície normal se sobrepõem no *scalar field*, o modelo não consegue separar as classes, caindo para AUROC $= 0{,}672$, próximo ao acaso.

No regime *gap*, onde o *scalar field* discrimina com clareza, o NormFlow atinge AUROC $= 0{,}914$ sem ter visto um único ponto rotulado como fissura. Esse resultado o torna o único método viável para inspeções exploratórias em estruturas sem dados rotulados disponíveis, funcionando como filtro de triagem de primeira passagem. Incorporar o vetor 18D completo, além das 16 *features* contextuais atuais, pode ampliar a discriminabilidade nas nuvens do regime

overlap.

ScalarMAE e MemBank

O ScalarMAE e o MemBank compartilham um problema central: o objetivo de treino não está alinhado com a tarefa final. No ScalarMAE, o pré-treino por reconstrução de *features* mascaradas não orienta o encoder a separar fissuras de superfícies normais, e 50 épocas de *fine-tuning* são insuficientes para reorientar as representações aprendidas, evidenciado pelo AUROC $= 0{,}649$, mais próximo do acaso do que dos demais métodos supervisionados. No MemBank, o banco de patches normais construído durante o treino não cobre a diversidade de superfícies em nuvens não vistas, e sem rótulos de avaria o método não distingue fissuras de irregularidades de superfície como textura de argamassa ou manchas de umidade, resultando em AUROC $= 0{,}626$.

O potencial de ambos está condicionado a mudanças de formulação: no ScalarMAE, substituir o MAE por aprendizado contrastivo alinhado à tarefa de classificação; no MemBank, aplicar *coreset greedy* (ROTH *et al.*, 2022) para compactar o banco de memória e aumentar a cobertura de superfícies normais com menor redundância.

6.6 Síntese dos Resultados

Os resultados deste benchmark levam a uma conclusão científica central:

> *Em problemas de detecção de fissuras baseados em nuvens de pontos LiDAR, a qualidade das features físicas e o protocolo de treinamento global podem ser mais determinantes para o desempenho do que a sofisticação arquitetural do modelo.*

O desempenho consistente dos métodos supervisionados todos com AUROC $> 0{,}92$ confirma que o *scalar field* possui elevado poder discriminativo para a separação entre regiões fissuradas e superfícies íntegras, ao mesmo tempo que concentra a dependência do benchmark em um único tipo de sensor e modalidade de aquisição. Essa conclusão tem implicações práticas diretas: para a maioria das nuvens deste dataset, um MLP simples treinado com *features* físicas bem construídas supera arquiteturas de ponta com custo computacional três ordens de magnitude maior. O investimento em engenharia de *features* e em estratégia de treinamento pode ser mais produtivo do que a busca por arquiteturas mais complexas, ao menos no domínio estudado.

7 CONCLUSÃO

7.1 Considerações Finais

Este trabalho propõe e avalia oito métodos para a detecção automática de fissuras em estruturas de concreto a partir de nuvens de pontos LiDAR, sob o protocolo LOCO com treinamento global. O GMMScalar, um perceptron multicamadas treinado nas 18 *features* escalares e geométricas de todas as nuvens de treino concatenadas, obteve o melhor desempenho: AUROC $= 0{,}976 \pm 0{,}023$ e AP $= 0{,}796$ sobre as 32 nuvens de avaria. Os achados sugerem que a eficácia na detecção de fissuras, ao menos neste conjunto de dados, está mais fortemente associada ao protocolo de treinamento global do que à complexidade da arquitetura métodos como GMM-AE e GMM-VAE, que atingiam AUROC $\approx 0{,}60$ no modo por nuvem, alcançaram AUROC $\approx 0{,}97$ após a migração para o modo global.

A análise diagnóstica confirmou dois regimes de discriminabilidade no *dataset*: nuvens com separação clara no *scalar field* (onde o GMMScalar atinge AUROC $\approx 1{,}0$) e nuvens com sobreposição escalar (onde o sinal escalar isolado é menos discriminativo, mas o treinamento global com múltiplas *features* mantém AUROC $> 0{,}90$ na maioria dos casos). Os métodos não supervisionados (NormFlow e MemBank), ao não aprenderem com rótulos de avaria de outras nuvens, ficam ΔAUROC $\approx 0{,}35$ abaixo do GMMScalar, sinalizando o valor dos rótulos no cenário de treino global.

7.2 Verificação dos Objetivos

Os objetivos específicos definidos no Capítulo 2 foram verificados conforme descrito a seguir:

- **Dataset**: Dataset de 64 nuvens de pontos LiDAR coletadas da Igreja Nossa Senhora do Rosário (Aracati-CE), com 22 nuvens normais, 32 de avaria e 10 de teste. Anotações por ponto derivadas do *scalar field*. ✓
- **Benchmark comparativo**: Oito métodos avaliados sob o protocolo LOCO com treinamento global, com métricas AUROC, F1 e AP reportadas para cada nuvem e em média. ✓
- **GMMScalar como método de referência**: MLP global treinado nas 18 *features* escalares e geométricas obteve AUROC $= 0{,}976$, melhor desempenho entre todos os métodos

avaliados. ✓

- **Protocolo global**: Demonstrado que o treinamento com dados de todas as nuvens de treino concatenadas supera consistentemente o modo por nuvem, com ganho médio de 36 pontos de AUROC nos modelos GMM. ✓
- **Análise de regimes**: Dois regimes de separabilidade identificados e caracterizados (gap e overlap), com diagnóstico das causas de variação do desempenho por nuvem. ✓

7.3 Principais Contribuições

As principais contribuições técnicas deste trabalho são:

1. O protocolo de treinamento global com avaliação LOCO, que demonstra que modelos treinados com dados concatenados de todas as nuvens superam consistentemente o paradigma de treinamento por nuvem — o principal achado metodológico deste trabalho;
2. O benchmark de oito métodos sob protocolo LOCO global, estabelecendo linhas de base para pesquisas futuras em detecção de fissuras 3D em nuvens de pontos LiDAR de patrimônio histórico;
3. O dataset de 64 nuvens com anotações por ponto, cobrindo superfícies normais e com avarias da Igreja Nossa Senhora do Rosário (Aracati-CE);
4. A análise dos dois regimes de discriminabilidade do *scalar field* (gap e overlap), que explica a variância elevada observada nos resultados e constitui referência para trabalhos futuros nesse domínio.

7.4 Limitações e Ameaças à Validade

7.4.1 Validade Interna

Circularidade nos rótulos: As anotações foram derivadas do próprio *scalar field*, que é também a principal *feature* dos modelos. Isso cria uma correlação estrutural que pode inflar as métricas de desempenho: qualquer método que utilize o *scalar field* tem vantagem estrutural sobre métodos que não o utilizam. A interpretação comparativa da tabela de resultados deve considerar essa assimetria.

Rotulagem sem especialista independente: As anotações foram realizadas por inspeção visual no CloudCompare sem validação por especialista em patologia de estruturas de concreto. A qualidade dos rótulos é uma fonte de variância não controlada nos resultados.

Dataset pequeno: O dataset contém 32 nuvens de avaria, que podem ser insuficientes para o treinamento estável de métodos de aprendizado profundo. Os desvios padrão elevados nas métricas (ex: F1 $\pm$0,247 no ScalarGAT) refletem essa situação.

7.4.2 Validade Externa

Única estrutura: Todos os dados foram coletados da Igreja Nossa Senhora do Rosário (Aracati-CE). A generalização para outros tipos de estrutura (pontes, viadutos, edificações industriais) não foi avaliada.

Único sensor: Os dados foram capturados exclusivamente com o scanner Leica BLK 360. O *scalar field* desse sensor tem características físicas específicas que podem não se reproduzir em outros dispositivos LiDAR, limitando a generalização dos resultados.

7.4.3 Validade de Construção

Definição operacional de fissura: A fissura é definida como o intervalo $[x_1, x_2]$ do *scalar field* selecionado visualmente no CloudCompare. Essa definição pode incluir anomalias com resposta escalar similar (manchas de umidade, variações de textura) ou excluir fissuras muito finas com escalar fora do intervalo selecionado.

Regime overlap: Nuvens com sobreposição escalar entre regiões de fissura e superfícies normais não são discrimináveis pelo *scalar field* isolado. Nenhum dos métodos avaliados resolve esse regime de forma satisfatória.

7.5 Trabalhos Futuros

Com base nas limitações identificadas, propõem-se as seguintes direções para pesquisas futuras:

7.5.1 Fusão com Features Geométricas e RGB

A integração do *scalar field* com *features* geométricas (curvatura local, rugosidade de superfície) e cor RGB pode resolver parcialmente o regime de sobreposição escalar. Métodos de fusão multi-modal em nuvens de pontos (GUO *et al.*, 2020) representam um caminho promissor.

7.5.2 Refinamento dos Métodos Não Supervisionados

Os métodos não supervisionados avaliados, NormFlow e ScalarMemBank, ficaram consistentemente abaixo dos supervisionados em todas as métricas, o que não implica que o paradigma seja inferior, mas que há espaço para aprimoramento. Para o ScalarMemBank, a aplicação de *coreset greedy* (GONZALEZ, 1985) , como proposto no PatchCore (ROTH *et al.*, 2022), pode compactar o banco de memória sem perda de desempenho. Para o NormFlow, a incorporação de *features* geométricas além das 16D atuais pode ampliar a capacidade discriminativa nas nuvens de regime *overlap*.

7.5.3 Expansão e Anotação Independente do Dataset

Futuros trabalhos devem coletar nuvens de pontos de diferentes tipos de estruturas (pontes, viadutos, edificações industriais) e adotar anotações independentes do *scalar field*, por exemplo, por inspeção visual humana, para eliminar a circularidade identificada (ZHU *et al.*, 2024).

7.5.4 Anotação com Referência a Classes de Fissura

A incorporação das categorias de fissura definidas pela ABNT NBR 6118 (micro, pequenas, médias e largas) como rótulos multiclasse permitiria avaliar a capacidade dos métodos em distinguir diferentes graus de severidade, indo além da detecção binária avaliada neste trabalho.

REFERÊNCIAS

ALZUBAIDI, L.; ZHANG, J.; HUMAIDI, A. J.; AL-DUJAILI, A.; DUAN, Y.; AL-SHAMMA, O.; SANTAMARÍA, J.; FADHEL, M. A.; AL-AMIDIE, M.; FARHAN, L. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. **Journal of Big Data**, SpringerOpen, v. 8, n. 1, p. 1–74, 2021.

ARMENI, I.; SENER, O.; ZAMIR, A. R.; JIANG, H.; BRILAKIS, I.; FISCHER, M.; SAVARESE, S. 3d semantic parsing of large-scale indoor spaces. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2016. p. 1534–1543.

AUTODESK. **Autodesk ReCap**. 2025. Acesso em: 01 ago. 2025. Disponível em: https://www.autodesk.com/products/recap/overview.

BABACAN, K.; CHEN, L.; SOHN, G. Semantic segmentation of indoor point clouds using convolutional neural network. **ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences**, IV-4/W4, p. 101–108, 11 2017.

CHEN, J.; CHO, Y. Crackembed: Point feature embedding for crack segmentation from disaster site point clouds with anomaly detection. **Advanced Engineering Informatics**, v. 52, p. 101550, 04 2022.

CHEN, T.; KORNBLITH, S.; NOROUZI, M.; HINTON, G. A simple framework for contrastive learning of visual representations. In: **Proceedings of the International Conference on Machine Learning (ICML)**. [*S. n.*], 2020. v. 119, p. 1597–1607. Disponível em: https://arxiv.org/abs/2002.05709.

CLOUDCOMPARE. **CloudCompare – 3D point cloud and mesh processing software**. 2025. Acesso em: 01 ago. 2025. Disponível em: https://www.cloudcompare.org/.

DINH, L.; SOHL-DICKSTEIN, J.; BENGIO, S. Density estimation using real NVP. In: **International Conference on Learning Representations (ICLR)**. [*S. n.*], 2017. Disponível em: https://openreview.net/forum?id=HkpbnH9lx.

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

FLAH, M.; NUNEZ, I.; CHAABENE, W. B.; NEHDI, M. L. Machine learning algorithms in civil structural health monitoring: a systematic review. **Archives of Computational Methods in Engineering**, Springer, v. 28, n. 4, p. 2621–2643, 2021.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 7. ed. São Paulo: Atlas, 2019.

GONZALEZ, T. F. Clustering to minimize the maximum intercluster distance. **Theoretical Computer Science**, Elsevier, v. 38, p. 293–306, 1985.

GOOGLE. **Google Colaboratory**. 2025. Acesso em: 01 ago. 2025. Disponível em: https://colab.research.google.com/.

GUO, Y.; WANG, H.; HU, Q.; LIU, H.; LIU, L.; BENNAMOUN, M. Deep learning for 3d point clouds: A survey. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 43, n. 12, p. 4338–4364, 2020.

HE, K.; CHEN, X.; XIE, S.; LI, Y.; DOLLÁR, P.; GIRSHICK, R. Masked autoencoders are scalable vision learners. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S. n.*], 2022. p. 16000–16009. Disponível em: https://arxiv.org/abs/2111.06377.

HORWITZ, E.; HOSHEN, Y. Back to the feature: Classical 3d features are (almost) all you need for 3d anomaly detection. In: **British Machine Vision Conference (BMVC)**. [*S. n.*], 2023. Disponível em: https://arxiv.org/abs/2203.05550.

HU, Q.; YANG, B.; XIE, L.; ROSA, S.; GUO, Y.; WANG, Z.; TRIGONI, N.; MARKHAM, A. **RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds**. 2020. Disponível em: https://arxiv.org/abs/1911.11236.

JING, L.; TIAN, Y. Self-supervised visual feature learning with deep neural networks: A survey. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 43, n. 11, p. 4037–4058, 2021.

JR, B. F. S.; HOSKERE, V.; NARAZAKI, Y. Advances in computer vision-based civil infrastructure inspection and monitoring. **Engineering**, Elsevier, v. 5, n. 2, p. 199–222, 2019.

LAKATOS, E. M.; MARCONI, M. d. A. **Fundamentos de metodologia científica**. 8. ed. São Paulo: Atlas, 2017.

LAXMAN, K.; TABASSUM, N.; AI, L.; COLE, C.; ZIEHL, P. Automated crack detection and crack depth prediction for reinforced concrete structures using deep learning. **Construction and Building Materials**, Elsevier, v. 370, p. 130709, 2023.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [*S. n.*], 2017. p. 2980–2988. Disponível em: https://arxiv.org/abs/1708.02002.

LIU, S.; ZHANG, M.; KADAM, P.; KUO, C.-C. J. **3D Point Cloud Analysis Traditional, Deep Learning, and Explainable Machine Learning Methods**. [*S. l.*]: Editora Springer, 2021. v. 1º edição.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: **International Conference on Learning Representations (ICLR)**. [*S. n.*], 2019. Disponível em: https://arxiv.org/abs/1711.05101.

MIRZAEI, K.; ARASHPOUR, M.; ASADI, E.; MASOUMI, H.; BAI, Y.; BEHNOOD, A. 3d point cloud data processing with machine learning for construction and infrastructure applications: A comprehensive review. **Advanced Engineering Informatics**, v. 51, p. 101501, 2022. ISSN 1474-0346. Disponível em: https://www.sciencedirect.com/science/article/pii/S1474034621002500.

PANG, G.; SHEN, C.; CAO, L.; HENGEL, A. V. D. Deep learning for anomaly detection: A review. **ACM computing surveys**, ACM New York, NY, USA, v. 54, n. 2, p. 1–38, 2021.

PANG, Y.; WANG, W.; TAY, F. E. H.; LIU, W.; TIAN, Y.; YUAN, L. Masked autoencoders for point cloud self-supervised learning. In: **European Conference on Computer Vision (ECCV)**. [*S. n.*], 2022. p. 604–621. Disponível em: https://arxiv.org/abs/2203.06604.

POUX, F.; BILLEN, R. Voxel-based 3d point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. **ISPRS International Journal of Geo-Information**, MDPI, v. 8, n. 5, p. 213, 2019.

QI, C. R.; SU, H.; MO, K.; GUIBAS, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)**. [*S. n.*], 2017. p. 652–660. Disponível em: https://arxiv.org/abs/1612.00593.

QI, C. R.; YI, L.; SU, H.; GUIBAS, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [*S. n.*], 2017. v. 30. Disponível em: https://arxiv.org/abs/1706.02413.

ROTH, K.; PEMULA, L.; ZEPEDA, J.; SCHÖLKOPF, B.; BROX, T.; GEHLER, P. Towards total recall in industrial anomaly detection. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S. n.*], 2022. p. 14318–14328. Disponível em: https://arxiv.org/abs/2106.08265.

SONY, S.; LAVENTURE, S.; SADHU, D. A literature review of next-generation smart sensing technology in structural health monitoring. **Structural Control and Health Monitoring**, Wiley Online Library, v. 26, n. 3, p. e2321, 2019.

THOMAS, H.; QI, C. R.; DESCHAUD, J.; MARCOTEGUI, B.; GOULETTE, F.; GUIBAS, L. J. Kpconv: Flexible and deformable convolution for point clouds. **CoRR**, abs/1904.08889, 2019. Disponível em: http://arxiv.org/abs/1904.08889.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. **CoRR**, abs/1706.03762, 2017. Disponível em: http://arxiv.org/abs/1706.03762.

VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIÒ, P.; BENGIO, Y. Graph attention networks. In: **International Conference on Learning Representations (ICLR)**. [*S. n.*], 2018. Disponível em: https://arxiv.org/abs/1710.10903.

VIEIRA, M. M.; GONçALVES, J. E.; SILVA, D. M. d. O.; MESQUITA, E. F. T.; LIMA, J. M. Semi-automatic scan-to-bim procedure applied to architectural ornaments of nossa senhora do rosário church, aracati-ce. **Journal of Building Pathology and Rehabilitation**, 2024. Disponível em: https://link.springer.com/article/10.1007/s41024-024-00436-0.

WANG, Y.; SUN, Y.; LIU, Z.; SARMA, S. E.; BRONSTEIN, M. M.; SOLOMON, J. M. Dynamic graph CNN for learning on point clouds. **ACM Transactions on Graphics**, v. 38, n. 5, 2019. Disponível em: https://arxiv.org/abs/1801.07829.

WU, X.; JIANG, L.; WANG, P.-S.; LIU, Z.; LIU, X.; QIAO, Y.; OUYANG, W.; HE, T.; ZHAO, H. Point transformer v3: Simpler, faster, stronger. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S. n.*], 2024. Disponível em: https://arxiv.org/abs/2312.10035.

ZHAO, H.; JIANG, L.; JIA, J.; TORR, P.; KOLTUN, V. **Point Transformer**. 2021. Disponível em: https://arxiv.org/abs/2012.09164.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Open3d: A modern library for 3d data processing. **arXiv preprint arXiv:1801.09847**, 2018. Disponível em: https://arxiv.org/abs/1801.09847.

ZHU, Q.; FAN, L.; WENG, N. Advancements in point cloud data augmentation for deep learning: A survey. **Pattern Recognition**, v. 153, p. 110532, 2024. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320324002838.

ZONG, B.; SONG, Q.; MIN, M. R.; CHENG, W.; LUMEZANU, C.; CHO, D.; CHEN, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: **International Conference on Learning Representations (ICLR)**. [*S. n.*], 2018. Disponível em: https://openreview.net/forum?id=BJJLHbb0-.

GLOSSÁRIO

Anomaly detection	Detecção de padrões, pontos ou regiões que se desviam do comportamento considerado normal nos dados.
As-built	Representação digital do estado real de uma edificação ou estrutura após sua construção ou levantamento.
Backbone	Rede principal usada para extrair características dos dados antes de uma camada ou módulo específico de decisão.
Ball query	Estratégia de agrupamento que seleciona vizinhos dentro de um raio definido em torno de um ponto.
Batch	Lote de amostras processado em conjunto durante uma etapa de treinamento.
Benchmark	Conjunto de referência usado para comparar métodos, modelos ou resultados experimentais.
Coreset	Subconjunto representativo de dados usado para reduzir custo computacional mantendo diversidade informativa.
Dataset	Conjunto organizado de dados utilizado para treinamento, validação ou teste de modelos.
Deep learning	Área de aprendizado de máquina baseada em redes neurais profundas capazes de aprender representações hierárquicas.
Embedding	Representação vetorial compacta que codifica características relevantes de uma amostra, ponto ou região.
Encoder	Módulo neural responsável por transformar dados de entrada em uma representação latente ou de características.
Feature	Característica ou atributo numérico usado como entrada ou representação intermediária em um modelo.
Fine-tuning	Ajuste adicional de um modelo pré-treinado para adaptá-lo a uma tarefa ou conjunto de dados específico.
Focal Loss	Função de perda que reduz o peso de exemplos fáceis e enfatiza exemplos difíceis, útil em problemas desbalanceados.
Framework	Conjunto de ferramentas e bibliotecas que fornece estrutura para desenvolvimento e execução de modelos.

Gap	Separação clara entre distribuições, pontuações ou regimes de discriminabilidade nos dados.
Grid	Estrutura regular de células usada para organizar dados espaciais, comum em imagens e representações discretizadas.
Kernel	Função, filtro ou operação usada para calcular relações locais entre pontos ou características.
Learning rate	Taxa que controla o tamanho das atualizações dos parâmetros durante o treinamento de um modelo.
Loss	Função de perda usada para medir o erro do modelo e orientar o processo de otimização.
Memory bank	Banco de representações armazenadas usado como referência para comparação, recuperação ou detecção de anomalias.
Mini-batch	Pequeno lote de amostras usado em uma iteração de treinamento.
Overlap	Sobreposição entre distribuições ou pontuações, dificultando a separação entre classes ou regimes.
Patch	Recorte ou subconjunto local de dados usado para representar uma região da nuvem de pontos.
Pipeline	Sequência organizada de etapas de processamento, treinamento, inferência e avaliação.
Pooling	Operação que agrega informações de múltiplos elementos, reduzindo dimensionalidade ou sintetizando características.
Prior	Distribuição ou conhecimento assumido previamente pelo modelo sobre variáveis latentes ou parâmetros.
Query	Consulta usada para selecionar pontos, vizinhos ou chaves em operações de busca e atenção.
Scalar field	Campo escalar associado aos pontos da nuvem, representando uma medida numérica por ponto.
Scanner	Equipamento de varredura usado para capturar geometria tridimensional de objetos ou ambientes.
Score	Pontuação atribuída a uma amostra, ponto ou região, geralmente usada para decisão ou ranqueamento.

Self-attention	Mecanismo de atenção que calcula relações entre elementos de uma mesma sequência ou conjunto.
Skip connection	Conexão direta entre camadas não adjacentes, usada para preservar informação e facilitar o treinamento.
Softmax	Função que transforma valores numéricos em probabilidades normalizadas.
Threshold	Limiar usado para converter pontuações contínuas em decisões, como normal ou anômalo.
Token	Unidade de representação processada por modelos, especialmente em arquiteturas baseadas em atenção.
Transformer	Arquitetura neural baseada em mecanismos de atenção, aplicada a sequências, imagens e nuvens de pontos.
Voxel	Unidade volumétrica regular usada para discretizar espaços tridimensionais.
Z-score	Normalização que expressa um valor em termos de desvios padrão em relação à média.

Algoritmo 1: Protocolo de avaliação LOCO (*Leave-One-Cloud-Out*)

Data: Nuvens normais $\mathscr{T}_N = \{c_1, \ldots, c_{22}\}$, nuvens de avaria $\mathscr{T}_A = \{a_1, \ldots, a_{32}\}$, método $\mathscr{M}$

Result: AUROC, F1 e AP: médias e desvios padrão sobre 32 rodadas

resultados $\leftarrow$ []

for $i = 1$ *até* 32 **do**

$\mathscr{T}_{\text{treino}} \leftarrow \mathscr{T}_N \cup (\mathscr{T}_A \setminus \{a_i\})$

`// Métodos não supervisionados usam apenas as 22 nuvens normais`

Treinar $\mathscr{M}$ em $\mathscr{T}_{\text{treino}}$

$\mathbf{s} \leftarrow \mathscr{M}$.inferir$(a_i)$ `// score de anomalia por ponto`

Binarizar com *threshold* P95 dos scores de $\mathscr{T}_N$

$\text{auroc}_i \leftarrow \text{AUROC}(\mathbf{s}, \text{rótulos}(a_i))$

$\text{f1}_i \leftarrow \text{F1}(\mathbf{s}, \text{rótulos}(a_i))$

$\text{ap}_i \leftarrow \text{AP}(\mathbf{s}, \text{rótulos}(a_i))$

resultados.append$((\text{auroc}_i, \text{f1}_i, \text{ap}_i))$

end

return média(resultados), std(resultados)

Algoritmo 2: GMMScalar – MLP supervisionado global com vetor de *features* 18D

Data: Conjunto de treino LOCO $\mathscr{T}_{-i}$ (nuvens normais + avaria com GT, excluindo a_i), épocas $E = 200$, mini-lote $B = 4096$

Result: MLP f_θ treinado e normalizador *StandardScaler*

`// Montagem do conjunto de treino global`

$X \leftarrow []$, $y \leftarrow []$

for *cada nuvem* $c \in \mathscr{T}_{-i}$ **do**

  if *c é nuvem de avaria (has_crack)* **then**

    Adicionar todos os pontos de c com rótulos GT ($y \in \{0,1\}$)

  else

    Adicionar todos os pontos de c com rótulo $y = 0$

  end

end

$X \leftarrow$ StandardScaler.fit_transform(X)

$w_{\text{pos}} \leftarrow |y{=}0| \,/\, |y{=}1|$ `// peso para classe minoritária`

`// Arquitetura: Linear(18,64) + BN + Dropout(0,2) + Linear(64,32) + BN + Linear(32,1)`

Inicializar MLP f_θ, AdamW($\eta = 10^{-3}$, wd$= 10^{-4}$), CosineAnnealingLR

for *época* $e = 1$ *até* E **do**

  Amostrar mini-lote B aleatório de X

  $\hat{y} \leftarrow f_\theta(X_B)$

  $\mathscr{L} \leftarrow$ BCEWithLogitsLoss$(\hat{y},\ y_B,\ w_{\text{pos}})$

  Backpropagation + `AdamW.step()` + `Scheduler.step()`

end

return f_θ, StandardScaler

`// Inferência na nuvem de teste` a_i

$X_{\text{test}} \leftarrow$ StandardScaler.transform$(a_i$.features$)$

s $\leftarrow$ Sigmoid$(f_\theta(X_{\text{test}}))$ `// score por ponto` $\in [0,1]$

return s

Algoritmo 3: GMM-AE – Encoder-Classifier supervisionado global

Data: Conjunto de treino LOCO $\mathscr{T}_{-i}$, épocas $E = 200$, mini-lote $B = 4096$, dimensão latente $d = 8$

Result: EncoderClassifier f_θ treinado e normalizador *StandardScaler*

`// Montagem idêntica ao GMMScalar (Algoritmo 2)`

Construir X, y e w_{pos} a partir de $\mathscr{T}_{-i}$

$X \leftarrow \text{StandardScaler.fit\_transform}(X)$

`// Arquitetura do EncoderClassifier`

`// Encoder: Linear(18,32)+BN+ReLU` $\rightarrow$ `Linear(32,16)+ReLU` $\rightarrow$ `Linear(16,`d`)+ReLU`

`// Head: Linear(`d`,16)+ReLU` $\rightarrow$ `Linear(16,1)`

Inicializar f_θ, AdamW($\eta = 10^{-3}$, wd$= 10^{-4}$), CosineAnnealingLR

for *época* $e = 1$ *até* E **do**

- Amostrar mini-lote B aleatório de X
- $\hat{y} \leftarrow f_\theta(X_B)$ `// passa por encoder + head`
- $\mathscr{L} \leftarrow \text{BCEWithLogitsLoss}(\hat{y},\ y_B,\ w_{\text{pos}})$
- Backpropagation
- `clip_grad_norm(`θ`, 1,0)`
- `AdamW.step()` + `Scheduler.step()`

end

return f_θ, StandardScaler

`// Inferência na nuvem de teste` a_i

$X_{\text{test}} \leftarrow \text{StandardScaler.transform}(a_i.\text{features})$

$\mathbf{s} \leftarrow \text{Sigmoid}(f_\theta(X_{\text{test}}))$

return s

Algoritmo 4: GMM-VAE – VAE semi-supervisionado com *head* de classificação global

Data: Conjunto de treino LOCO $\mathscr{T}_{-i}$, épocas $E = 200$, mini-lote $B = 4096$, $d_z = 8$, $\lambda_{\text{cls}} = 1{,}0$, $\beta_{\text{KL}} = 0{,}001$

Result: VAEClassifier f_θ treinado e normalizador *StandardScaler*

Construir X, y e w_{pos} a partir de $\mathscr{T}_{-i}$

$X \leftarrow \text{StandardScaler.fit\_transform}(X)$

`// Arquitetura do VAEClassifier`

`// Encoder: Linear(18,32)+BN+ReLU` $\rightarrow$ μ: `Linear(32,`d_z`)`, $\log\sigma^2$: `Linear(32,`d_z`)`

`// Decoder: Linear(`d_z`,32)+ReLU` $\rightarrow$ `Linear(32,18)`

`// Head (aplicado em` μ`): Linear(`d_z`,16)+ReLU` $\rightarrow$ `Linear(16,1)`

Inicializar f_θ, AdamW($\eta = 10^{-3}$), CosineAnnealingLR

for *época* $e = 1$ *até* E **do**

- Amostrar mini-lote B de X
- $h \leftarrow \text{Encoder}(X_B)$
- $\mu, \log\sigma^2 \leftarrow$ projeções lineares de h
- $z \leftarrow \mu + \sigma \odot \varepsilon, \quad \varepsilon \sim \mathscr{N}(0, I)$ `// reparametrização`
- $\hat{x} \leftarrow \text{Decoder}(z)$
- $\hat{y} \leftarrow \text{Head}(\mu)$
- $\mathscr{L}_{\text{rec}} \leftarrow \text{MSE}(\hat{x},\ X_B)$
- $\mathscr{L}_{\text{KL}} \leftarrow -\frac{1}{2}\sum_j (1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$
- $\mathscr{L}_{\text{cls}} \leftarrow \text{BCEWithLogitsLoss}(\hat{y},\ y_B,\ w_{\text{pos}})$
- $\mathscr{L} \leftarrow \mathscr{L}_{\text{rec}} + \beta_{\text{KL}}\mathscr{L}_{\text{KL}} + \lambda_{\text{cls}}\mathscr{L}_{\text{cls}}$
- Backpropagation + `clip_grad_norm(`θ`, 1,0)` + `step()`

end

return f_θ, StandardScaler

`// Inferência: score via Head(`μ`) sem reparametrização`

$X_{\text{test}} \leftarrow \text{StandardScaler.transform}(a_i.\text{features})$

$\mu \leftarrow \text{Encoder}(X_{\text{test}}).\mu$

$\mathbf{s} \leftarrow \text{Sigmoid}(\text{Head}(\mu))$

return s

Algoritmo 5: ScalarGAT — Treinamento da rede de atenção em grafos

Data: Nuvem com *features* 18D e rótulos por ponto, $k = 16$, épocas $E = 30$, $\gamma = 2$

Result: ScalarGAT treinado

Construir grafo k-NN ($k = 16$) nas coordenadas XYZ

Inicializar AdamW com $\mathrm{lr} = 5 \times 10^{-4}$

$\mathrm{F1}_{\mathrm{melhor}} \leftarrow 0$, paciência $\leftarrow 0$

for *cada época* $e = 1$ *até* E **do**

  for *cada lote* B **do**

    $\mathbf{h}^{(1)} \leftarrow \mathrm{GATConv}(\mathrm{features}, \mathrm{grafo},\ 64\mathrm{D},\ 4\ \mathrm{cabeças})$

    $\mathbf{h}^{(2)} \leftarrow \mathrm{GATConv}(\mathbf{h}^{(1)}, \mathrm{grafo},\ 64\mathrm{D})$

    $\mathbf{h}^{(3)} \leftarrow \mathrm{GATConv}(\mathbf{h}^{(2)}, \mathrm{grafo},\ 32\mathrm{D})$

    $\hat{p}_i \leftarrow \mathrm{Sigmoid}(\mathrm{Linear}(\mathbf{h}_i^{(3)}, 1))$

    Calcular Focal Loss $\mathscr{L}_{\mathrm{focal}}$ entre $\hat{p}$ e rótulos

    Backward + `AdamW.step()`

  end

  $\mathrm{F1}_{\mathrm{atual}} \leftarrow$ avaliar F1 na validação

  if $F1_{atual} > F1_{melhor}$ **then**

    $\mathrm{F1}_{\mathrm{melhor}} \leftarrow \mathrm{F1}_{\mathrm{atual}}$

    paciência $\leftarrow 0$

    Salvar modelo

  else

    paciência $\leftarrow$ paciência $+ 1$

    if *paciência* ≥ 10 **then**

      Parar treinamento

    end

  end

end

Algoritmo 6: ScalarMAE em duas partes: pré-treino não supervisionado e *fine-tuning* supervisionado

Data: Nuvens normais $\mathscr{T}_N$, nuvens de treino $\mathscr{T}_{\text{sup}}$ com rótulos binários, $M = 32$ tokens, taxa de mascaramento $r = 0{,}75$, $\tau_s = 0{,}5$, épocas $E_1 = 100$ e $E_2 = 50$

Result: ScalarMAE com encoder pré-treinado e cabeçote de segmentação treinado

Parte 1: Pré-treino não supervisionado com mascaramento guiado pelo *scalar field*

for *cada época* $e = 1$ *até* E_1 **do**

  for *cada nuvem* $c_i \in \mathscr{T}_N$ **do**

    Dividir c_i em M tokens via FPS

    for *cada token* k **do**

      $p_k \leftarrow \exp(-\bar{s}_k/\tau_s)$ `// tokens com escalar baixo são mascarados com maior probabilidade`

    end

    Normalizar: $p_k \leftarrow p_k / \sum_j p_j$

    Mascarar $\lfloor r \cdot M \rfloor$ tokens com probabilidade $\{p_k\}$

    $\mathbf{e}_{\text{vis}} \leftarrow f_\theta(\text{tokens visíveis})$

    $\hat{\mathbf{t}}_{\text{mask}} \leftarrow \text{Decoder}(\mathbf{e}_{\text{vis}})$ `// reconstrução dos tokens mascarados`

    $\mathscr{L} \leftarrow \|\hat{\mathbf{t}}_{\text{mask}} - \mathbf{t}_{\text{mask}}\|^2$

    Backward + `AdamW.step()`

  end

end

Parte 2: *Fine-tuning* supervisionado com os rótulos disponíveis

Adicionar cabeçote $\text{Linear}(d_{\text{emb}}, 1) + \text{Sigmoid}$ ao encoder pré-treinado f_θ

Inicializar AdamW com $\text{lr} = 10^{-4}$ e escalonador cosseno

for *cada época* $e = 1$ *até* E_2 **do**

  for *cada mini-batch de 4096 pontos em* $\mathscr{T}_{sup}$ **do**

    $\hat{p}_i \leftarrow f_\theta(\mathbf{x}_i)$ `// score por ponto`

    $\mathscr{L} \leftarrow \text{FocalLoss}(\hat{p}_i,\, y_i)$ `//` $y_i \in \{0,1\}$ `rótulo verdadeiro`

    Backward + `AdamW.step()`

  end

end

Algoritmo 7: ScalarMemBank em duas partes: treinamento contrastivo com *memory bank* e inferência por pontuação de anomalia

Data: Nuvens normais $\mathscr{T} = \{c_1, \ldots, c_{22}\}$, nuvem de consulta q, épocas $E = 100$, $M = 64$ patches, $k_{\text{local}} = 32$, $k_{\text{global}} = 96$, $\alpha = 0{,}3$, $\beta = 0{,}35$ e $|\mathscr{N}| = 15$

Result: Encoder f_θ treinado, *memory bank* $\mathscr{M}$ e *score* de anomalia por ponto s'_i

Parte 1: Treinamento contrastivo e construção do *memory bank*

Calcular $g_{\text{lo}} = P_1$ e $g_{\text{hi}} = P_{99}$ do *scalar field* sobre $\mathscr{T}$

for *cada época* $e = 1$ *até* E **do**

  for *cada nuvem* $c_i \in \mathscr{T}$ **do**

    Extrair vetor de *features* 18D por ponto

    Normalizar *scalar field*: $s^* \leftarrow (s - g_{\text{lo}})/(g_{\text{hi}} - g_{\text{lo}} + \varepsilon)$

    Amostrar M centros $\{\mathbf{c}_k\}$ via FPS

    for *cada centro* $\mathbf{c}_k$ **do**

      $\bar{\mathbf{f}}^{\text{local}}, \sigma^{\text{local}} \leftarrow$ média e desvio dos k_{local} vizinhos (36D)

      $\bar{\mathbf{f}}^{\text{global}}, \sigma^{\text{global}} \leftarrow$ média e desvio dos k_{global} vizinhos (36D)

      $\mathbf{t}_k \leftarrow [\bar{\mathbf{f}}^{\text{local}},\ \sigma^{\text{local}},\ \bar{\mathbf{f}}^{\text{global}},\ \sigma^{\text{global}},\ \mathbf{c}_k] \in \mathbb{R}^{75}$

    end

    Forward: $\mathbf{e}_k \leftarrow f_\theta(\mathbf{t}_k) \in \mathbb{R}^{32}$ para todo k

    Calcular *NT-Xent Loss* com pares positivos (patches vizinhos) e negativos (patches distantes)

    Backward + `AdamW.step()`

  end

end

$\mathscr{M} \leftarrow \emptyset$

for *cada nuvem* $c_i \in \mathscr{T}$ **do**

  Extrair todos os *embeddings* de patches de c_i

  $\mathscr{M} \leftarrow \mathscr{M} \cup \{\mathbf{e}_k\}$ `// 1408 protótipos ao final`

end

Parte 2: Inferência e pontuação de anomalia

Extrair *features* 18D da nuvem de consulta q e normalizar *scalar field* com $g_{\text{lo}}, g_{\text{hi}}$

Construir tokens 75D para todos os pontos

Computar *embeddings*: $\mathbf{e}_i \leftarrow f_\theta(\mathbf{t}_i)$

for *cada ponto* i **do**

  $s_{\text{emb}}(i) \leftarrow \min_{\mathbf{m} \in \mathscr{M}} \left(1 - \dfrac{\mathbf{e}_i \cdot \mathbf{m}}{\|\mathbf{e}_i\| \|\mathbf{m}\|}\right)$

  $s_{\text{mah}}(i) \leftarrow$ distância de Mahalanobis de $(s_i^*, z_i, \nabla s_i)$ à distribuição normal

  $s_i \leftarrow \alpha \cdot s_{\text{emb}}(i) + (1 - \alpha) \cdot s_{\text{mah}}(i)$

end

for *cada ponto* i **do**

  $s'_i \leftarrow \beta \cdot \max_{j \in \mathscr{N}(i)} s_j + (1 - \beta) \cdot s_i$ `// suavização por difusão de vizinhança`

end

Algoritmo 8: NormFlow: RealNVP não supervisionado em pontos normais

Data: $\mathscr{T}_{-i}$ (pontos normais), épocas $E = 200$, vizinhos $k = 15$, mini-lote $B = 4096$

Result: RealNVP f_θ, normalização μ, σ

`// Features contextuais 16D`

for *cada nuvem* $c \in \mathscr{T}_{-i}$ **do**

  if *c é nuvem de avaria* **then**

    Selecionar pontos GT $= 0$

  else

    Usar todos os pontos de c

  end

  $\mathbf{b} \leftarrow$ colunas 10–17 de c.features `// 8D`

  $\bar{\mathbf{b}} \leftarrow$ média k-NN de $\mathbf{b}$ no espaço XYZ `// 8D`

  Acumular $[\mathbf{b} \| \bar{\mathbf{b}}]$ `// 16D`

end

Concatenar em $D \in \mathbb{R}^{N \times 16}$

$\mu \leftarrow$ média(D), $\sigma \leftarrow$ std(D)

$D \leftarrow (D - \mu)/(\sigma + \varepsilon)$

`// RealNVP: 6 acoplamentos, MLP oculto 128D`

Inicializar f_θ, AdamW($\eta = 10^{-3}$), CosineAnnealingLR

$\mathscr{L}_{\text{melhor}} \leftarrow \infty$

for *época* $e = 1$ *até* E **do**

  Amostrar mini-lote B de D

  $\mathscr{L} \leftarrow -\frac{1}{B} \sum \log p_{f_\theta}(\mathbf{x}_b)$ `// NLL`

  Backpropagation + `clip_grad_norm` + `step()`

  if $\mathscr{L} < \mathscr{L}_{melhor}$ **then**

    $\mathscr{L}_{\text{melhor}} \leftarrow \mathscr{L}$

    Salvar estado θ_{melhor}

  end

end

Restaurar θ_{melhor}

return f_θ, μ, σ

`// Inferência na nuvem de teste` a_i

Extrair features 16D de a_i como acima

$X_{\text{norm}} \leftarrow (X - \mu)/(\sigma + \varepsilon)$

$\text{NLL}_i \leftarrow -\log p_{f_\theta}(X_{\text{norm},i})$

$\widetilde{\text{NLL}} \leftarrow$ média k-NN da NLL `// suavização`

$\mathbf{s} \leftarrow \text{clip}((\widetilde{\text{NLL}} - P_2)/(P_{98} - P_2), 0, 1)$

return s

Algoritmo 9: Procedimento de treinamento do modelo

Data: Conjunto de treinamento com arquivos de entrada

Result: Modelo treinado

for *cada* `epoch` *de 1 até* `max_epochs` **do**

  for *cada arquivo no conjunto de treinamento* **do**

    1. Carregar `features (X)` e `labels (y)`
    2. **Forward pass**: $\hat{y} \leftarrow$ `modelo(X)`
    3. Calcular perda: $\mathscr{L} \leftarrow$ `CrossEntropyLoss(`$\hat{y}, y$`)`
    4. **Backward pass**: `L.backward()`
    5. Clipar gradientes: `clip_grad_norm(parâmetros, 1.0)`
    6. Atualizar pesos: `optimizer.step()`
    7. Resetar gradientes: `optimizer.zero_grad()`

  end

  Atualizar taxa de aprendizado: `scheduler.step()`

  Avaliar métricas de validação

  Aplicar `early stopping` se necessário

end

Algoritmo 10: PTv3-Binary — *Fine-tuning* supervisionado com Focal Loss

Data: Pesos PTv3 pré-treinados em S3DIS, nuvens de treino $\mathscr{T}_{\text{treino}}$ com rótulos por ponto, épocas $E = 50$, $\gamma = 2$

Result: PTv3-Binary com cabeçote de classificação binária treinado

Substituir cabeçote de segmentação original por Linear$(d_{\text{emb}}, 2)$

Inicializar AdamW com lr $= 10^{-4}$ e *weight decay* $= 0{,}05$

$\text{F1}_{\text{melhor}} \leftarrow 0$, paciência $\leftarrow 0$

for *cada época* $e = 1$ *até* E **do**

  for *cada lote* B *de* $\mathscr{T}_{treino}$ **do**

    $\hat{p}_i \leftarrow \text{PTv3}(B)$ `// probabilidade de fissura por ponto`

    $\mathscr{L}_{\text{focal}} \leftarrow -\sum_i \left[(1-\hat{p}_i)^{\gamma} y_i \log \hat{p}_i + \hat{p}_i^{\gamma}(1-y_i)\log(1-\hat{p}_i)\right]$

    Backward + `AdamW.step()`

  end

  $\text{F1}_{\text{atual}} \leftarrow$ avaliar F1 no conjunto de validação

  if $F1_{atual} > F1_{melhor}$ **then**

    $\text{F1}_{\text{melhor}} \leftarrow \text{F1}_{\text{atual}}$

    paciência $\leftarrow 0$

    Salvar modelo

  else

    paciência $\leftarrow$ paciência $+ 1$

    if *paciência* ≥ 10 **then**

      Parar treinamento

    end

  end

end

Algoritmo 11: Critério de Early Stopping

Data: $F1_{atual}$, $F1_{melhor}$, $paciencia$

Result: Modelo treinado ou parada antecipada

if $F1_{atual} > F1_{melhor}$ **then**

    $F1_{melhor} \leftarrow F1_{atual}$

    $paciencia \leftarrow 0$

    Salvar modelo

else

    $paciencia \leftarrow paciencia + 1$

    if $paciencia \geq 30$ **then**

        Parar treinamento

    end

end

Algoritmo 12: Critério de Early Stopping

Data: $F1_{atual}$, $F1_{melhor}$, $paciencia$

Result: Modelo treinado ou parada antecipada

if $F1_{atual} > F1_{melhor}$ **then**

    $F1_{melhor} \leftarrow F1_{atual}$

    $paciencia \leftarrow 0$

    Salvar modelo

else

    $paciencia \leftarrow paciencia + 1$

    if $paciencia \geq 30$ **then**

        Parar treinamento

    end

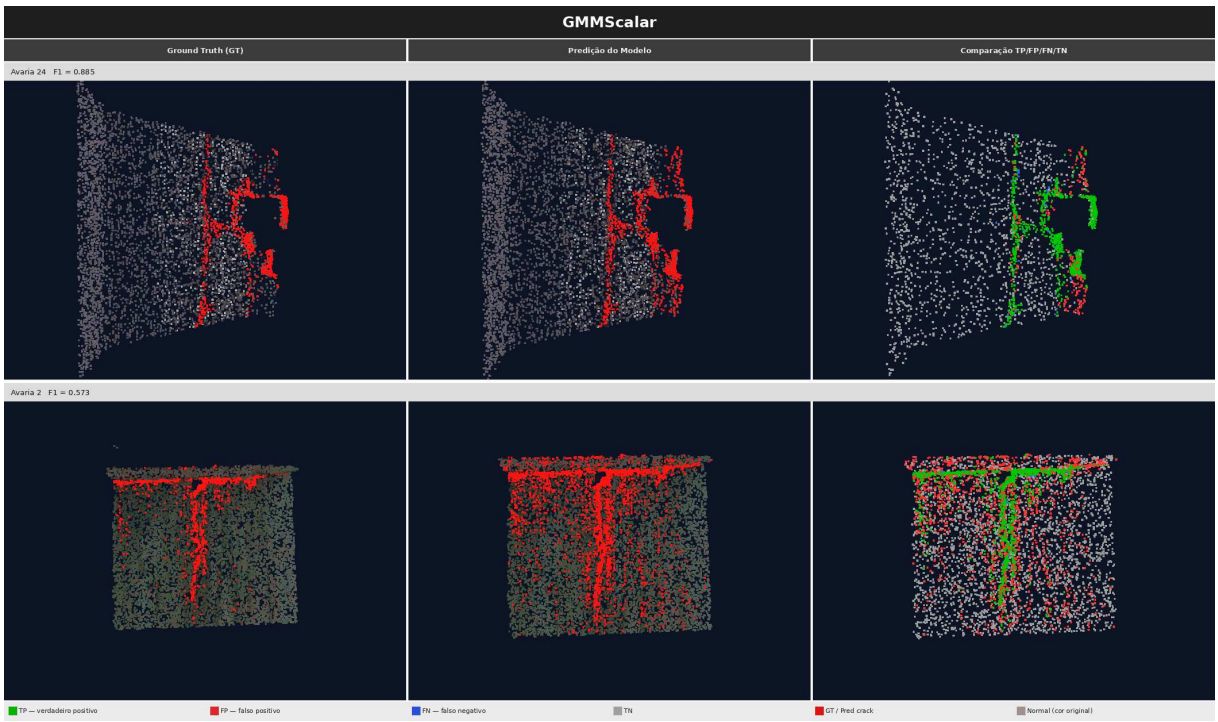
end

ANEXO A – MOSAICOS QUALITATIVOS DOS MÉTODOS AVALIADOS

Este apêndice apresenta os mosaicos qualitativos de todos os oito métodos avaliados sob protocolo LOCO. Cada mosaico exibe duas nuvens representativas do respectivo método (melhor caso e caso mediano, selecionados por F1), organizadas em três colunas: *ground truth* (GT), predição do modelo e comparação ponto a ponto (TP=verde, FP=vermelho, FN=azul, TN=cinza). Os pontos normais são exibidos com a cor original do escâner LiDAR, diferenciando-se visualmente do azul dos falsos negativos. Os métodos são apresentados separados por paradigma de supervisão.

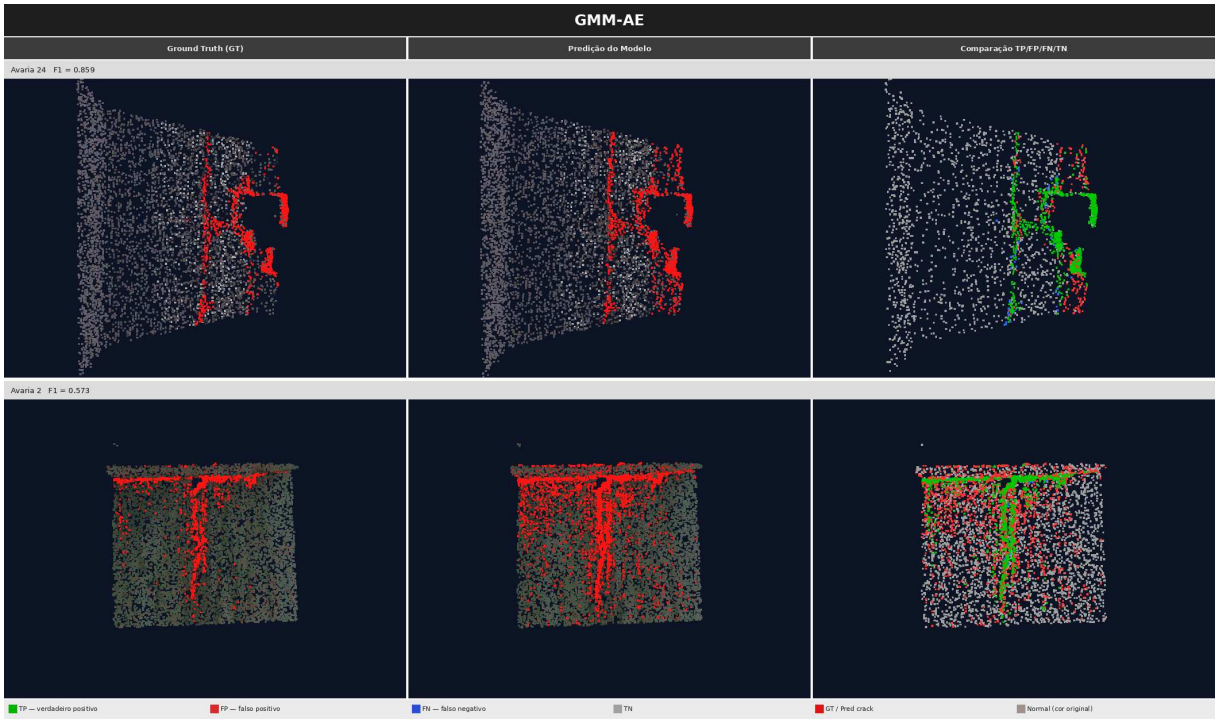
A.1 Métodos Supervisionados

Figura 20 – Mosaico qualitativo — GMMScalar (supervisionado, AUROC médio 0,976). MLP de duas camadas treinado nas 18 *features* escalares de forma global; obteve o melhor desempenho médio entre todos os métodos avaliados.



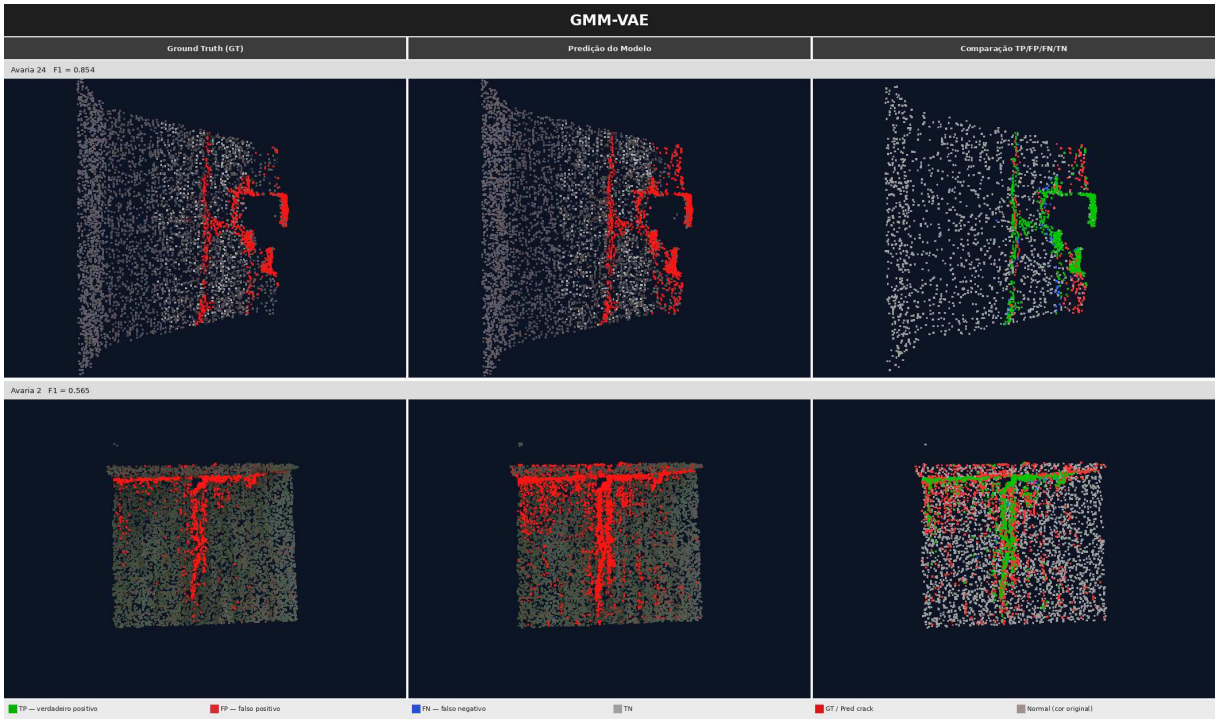
Fonte: Elaborado pelo autor.

Figura 21 – Mosaico qualitativo — GMM-AE (supervisionado, AUROC médio 0,965). Codificador que aprende representação latente das *features* 18D; a classificação binária opera no espaço comprimido.



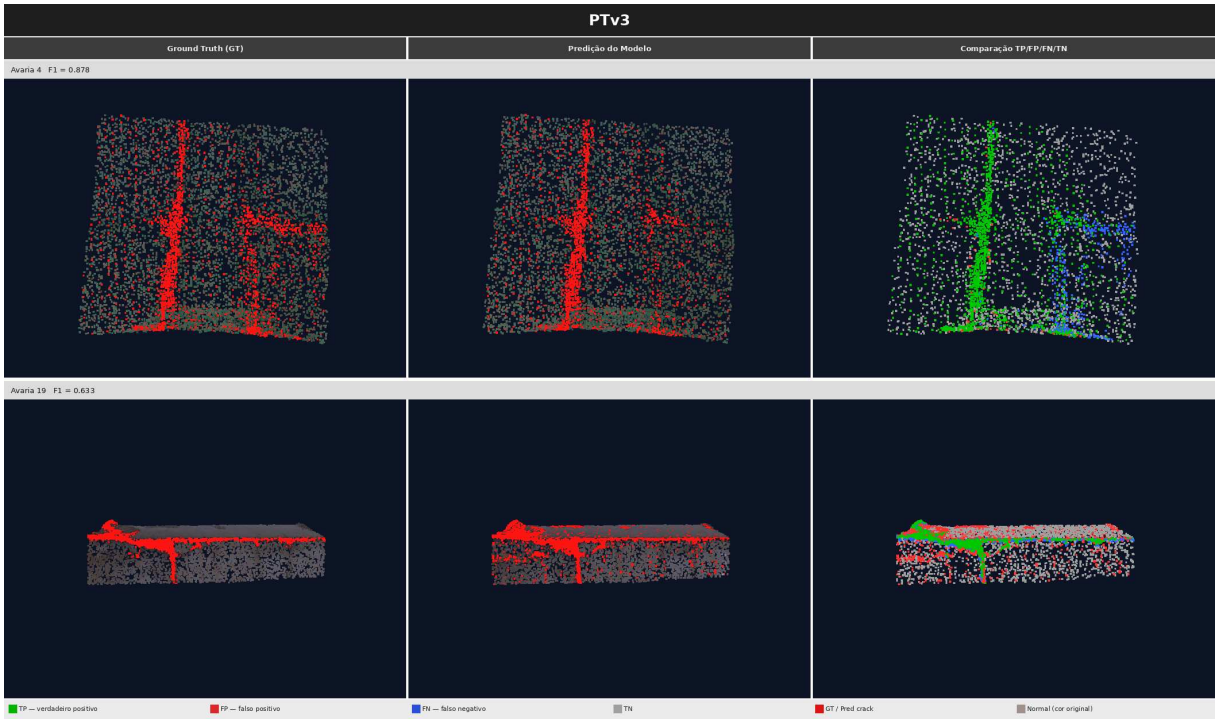
Fonte: Elaborado pelo autor.

Figura 22 – Mosaico qualitativo — GMM-VAE (supervisionado, AUROC médio 0,972). A regularização variacional do espaço latente produz predições de distribuição ligeiramente mais conservadora de falsos positivos em relação ao GMM-AE.



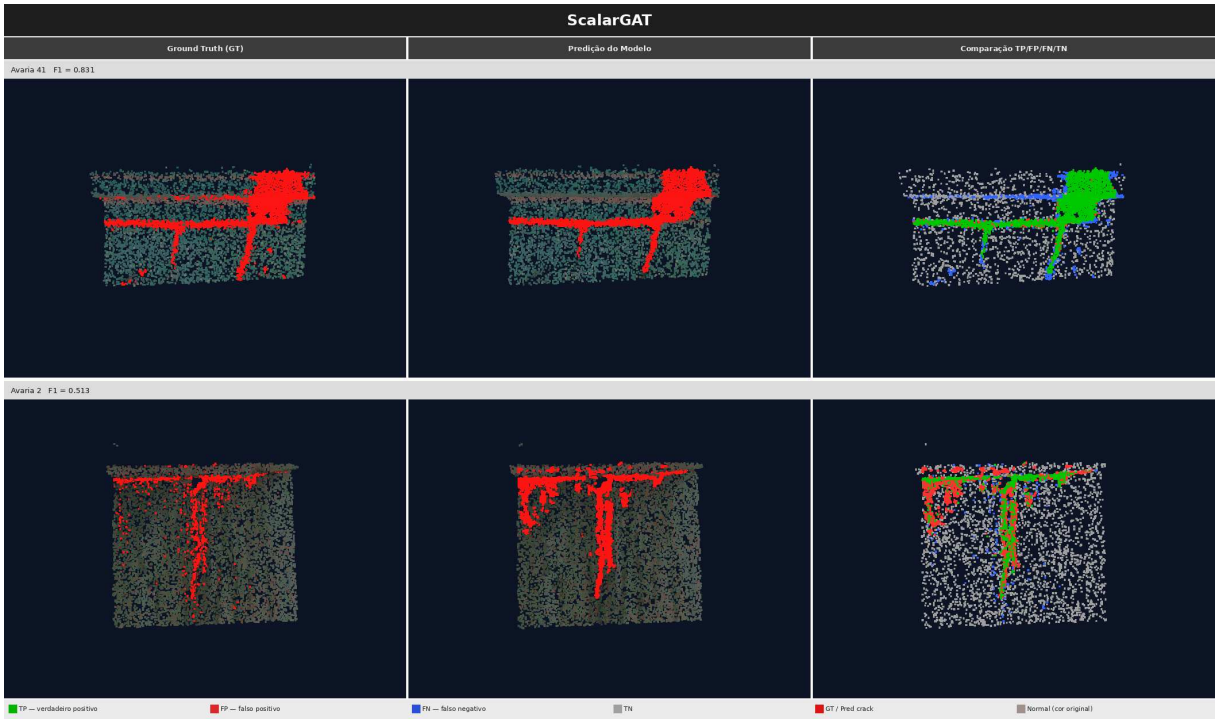
Fonte: Elaborado pelo autor.

Figura 23 – Mosaico qualitativo — PTv3 (supervisionado, AUROC médio 0,966). *Backbone* congelado do Point Transformer v3 com cabeça de segmentação binária; extrai representações geométricas 3D profundas, o que permite segmentar fissuras mesmo em nuvens do regime *overlap*.



Fonte: Elaborado pelo autor.

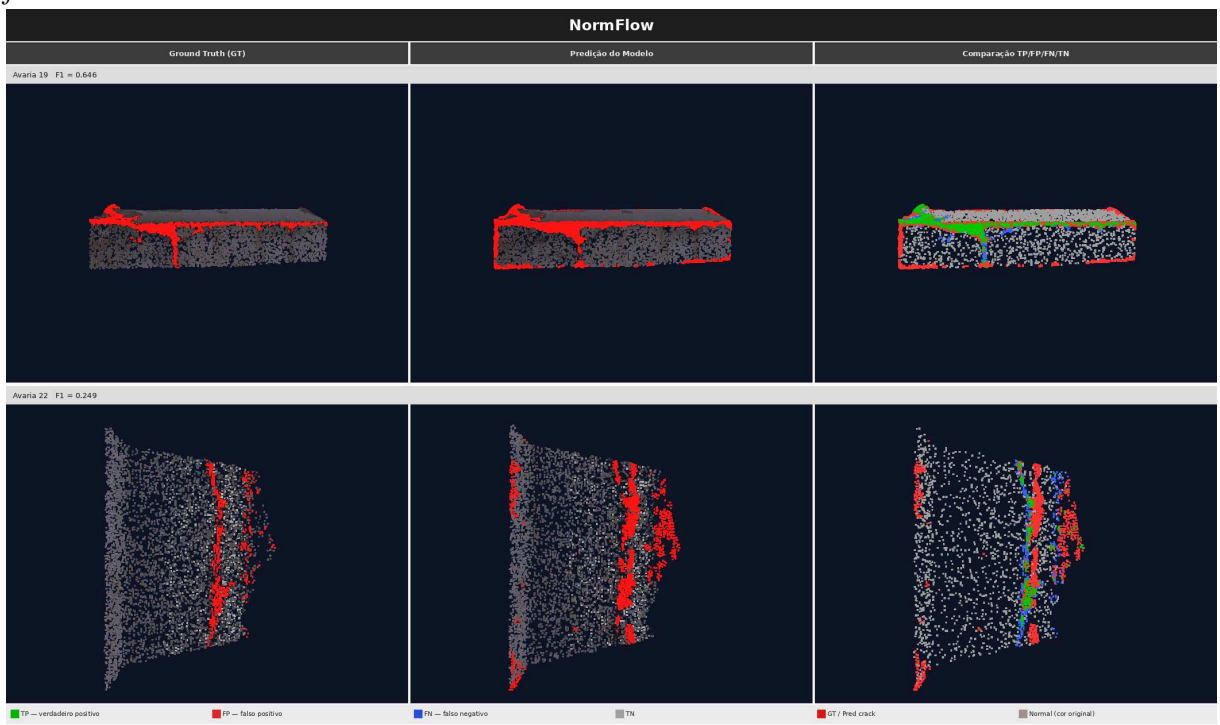
Figura 24 – Mosaico qualitativo — ScalarGAT (supervisionado, AUROC médio 0,929). Rede de atenção em grafos *k*-NN que propaga informação de vizinhança; apresenta maior taxa de falsos positivos e desvio padrão mais elevado (±0,074) em relação aos três melhores métodos.



Fonte: Elaborado pelo autor.

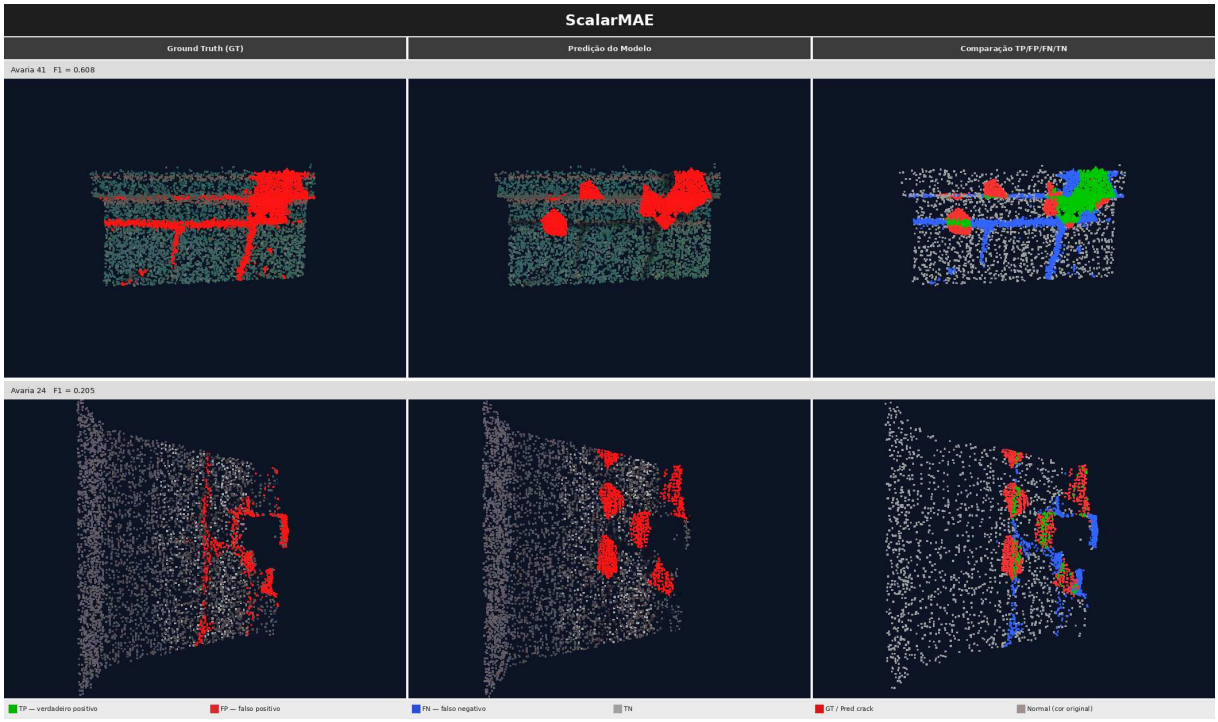
A.2 Métodos Não Supervisionados e Semi-supervisionado

Figura 25 – Mosaico qualitativo — NormFlow (não supervisionado, AUROC médio 0,778). Fluxo normalizante que modela a densidade das *features* normais e sinaliza anomalias por baixa verossimilhança; sem rótulos de avaria, o modelo não aprende o que é fissura quando o *scalar field* falha.



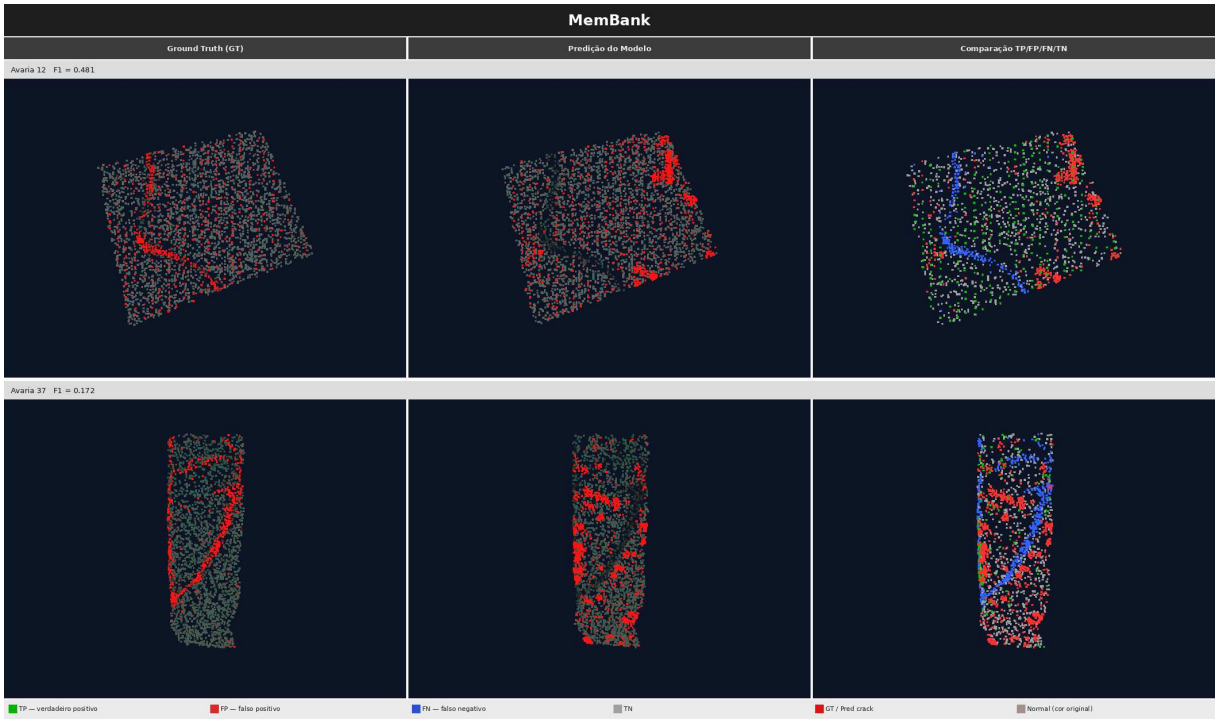
Fonte: Elaborado pelo autor.

Figura 26 – Mosaico qualitativo — ScalarMAE (semi-supervisionado, AUROC médio 0,649). Pré-treino por reconstrução mascarada seguido de *finetuning* supervisionado com poucos rótulos; não generaliza suficientemente entre os dois regimes de discriminabilidade do dataset.



Fonte: Elaborado pelo autor.

Figura 27 – Mosaico qualitativo — MemBank (não supervisionado, AUROC médio 0,626). Banco de memória de *patches* normais usado como referência para detecção de anomalias; a ausência de supervisão impede distinguir fissuras de variações texturais naturais da superfície escaneada.



Fonte: Elaborado pelo autor.