THIAGO TEIXEIRA PETRONE

Estimativa de tempo de processo para peças usinadas de baixa complexidade baseada em redes neurais gráficas

Versão Corrigida

São Paulo
2024

THIAGO TEIXEIRA PETRONE

Estimativa de tempo de processo para peças usinadas de baixa complexidade baseada em redes neurais gráficas

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2024

THIAGO TEIXEIRA PETRONE

Estimativa de tempo de processo para peças usinadas de baixa complexidade baseada em redes neurais gráficas

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Sistemas de Potência

Orientador:
José Aquiles Baesso Grimoni

São Paulo
2024

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Petrone, Thiago Teixeira
Estimativa de tempo de processo para peças usinadas de baixa complexidade baseada em redes neurais gráficas / T. T. Petrone -- versão corr. -- São Paulo, 2024.
144 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.Redes neurais gráficas 2.Usinagem 3.Tempo de processo 4.Modelo CAD I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

Em primeiro lugar à Blanca, que me ensinou quase tudo que eu sei sobre amor e dedicação. Aos meus pais, que desde sempre me ensinaram que *não existe almoço grátis*. À minha irmã que meu amor supera qualquer distância, física ou temporal. À família Bezerra que me acolheu como seu próprio filho e que sem a orientação e perspectiva de mundo eu jamais teria me tornado uma pessoa melhor. À Ju que me apoiou grande parte do processo, até que nossos caminhos se afastaram. À Pá que foi incansável no seu suporte incondicional. À minha avó que foi amor e dedicação durante a fase final do projeto. À Bárbara que me ensinou muito sobre resiliência. Ao meu orientador pela sua flexibilidade, paciência, confiança e apoio às minhas decisões.

RESUMO

PETRONE, Thiago Teixeira. Estimativa de Tempo de Processo para Peças Usinadas de Baixa Complexidade Baseada em Redes Neurais Gráficas. 2024. 144 f. Dissertação (Mestrado em Ciências) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2024.

A estimativa de tempo de processo é uma componente fundamental na determinação do custo total de produção de peças usinadas e desempenha um papel essencial nas estratégias da indústria de manufatura. Por um lado, as empresas enfrentam a escolha de realizar a produção internamente ou terceirizar a usinagem para fornecedores especializados, e essa escolha muitas vezes é baseada nas estimativas de tempo de processo. Por outro lado, os fornecedores de usinagem também dependem dessas estimativas para fazer orçamentos e calcular os custos de venda de seus produtos. Para ambas as pontas da cadeia, o principal desafio enfrentado é a necessidade de realizar estimativas de tempo de processo de forma ágil e com aceitável precisão, considerando as demandas de um mercado cada vez mais competitivo. Abordagens tradicionais frequentemente recorrem a análises manuais ou modelos que podem não representar a realidade da produção, resultando em estimativas imprecisas ou com longos tempos de processamento. No entanto, com o aumento da disponibilidade de informações detalhadas sobre peças usinadas e o contínuo avanço na capacidade dos algoritmos de aprendizado profundo, surge uma oportunidade para a implementação de abordagens mais sofisticadas. Nesta pesquisa, é abordado o problema da estimativa de tempo de processo para peças usinadas de baixa complexidade, com foco na agilidade e precisão necessárias para decisões estratégicas na indústria de manufatura. Nossos experimentos ressaltam o potencial das Redes Neurais Gráficas em extrair abstrações relevantes e processar de forma eficiente modelos 3D. Além disso, este estudo também fornece uma revisão dos métodos de precificação tradicionais e modernos, contextualizando as abordagens baseadas em aprendizado profundo dentro do panorama da usinagem.

Palavras-Chave – Redes neurais gráficas. Usinagem. Tempo de processo. Modelo CAD

ABSTRACT

PETRONE, Thiago Teixeira. Process Time Estimation for Low Complexity Machined Parts Based on Graph Neural Networks. 2024. 144 f. Dissertação (Mestrado em Ciências) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2024.

The estimation of process time is a fundamental component in determining the total production cost of machined parts and plays an essential role in manufacturing industry strategies. On one hand, companies face the choice of conducting production in-house or outsourcing machining to specialized suppliers, often basing this decision on process time estimates. On the other hand, machining suppliers also rely on these estimates to create budgets and calculate the selling costs of their products. For both ends of the spectrum, the main challenge is the need for swift and reasonably accurate process time estimations, considering the demands of an increasingly competitive market. Traditional approaches often resort to manual analyses or models that may not accurately represent production reality, resulting in imprecise estimates or long processing times. However, with the growing availability of detailed information about machined parts and the continuous advancement of deep learning algorithms, an opportunity arises to implement more sophisticated approaches. This research addresses the problem of process time estimation for low-complexity machined parts, with a focus on the agility and precision required for strategic decisions in the manufacturing industry. Our experiments highlight the potential of Graph Neural Networks to extract relevant abstractions efficiently from 3D models. Furthermore, this study provides a review of both traditional and modern pricing methods, contextualizing deep learning-based approaches within the machining landscape.

Keywords – Graph Neural Networks. Machining. Machining Time. CAD Model

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A estimativa do tempo de usinagem desempenha um papel de extrema importância tanto no desenvolvimento de produtos quanto no planejamento de produção. Na etapa de concepção, possibilita a otimização de projetos, a redução dos ciclos de fabricação e, consequentemente, a diminuição dos custos envolvidos. Já para os engenheiros de produção, a capacidade de realizar previsões do tempo de usinagem de forma eficiente contribui para o planejamento eficaz de todo o processo fabril, desde a estimativa de custos à programação de fabricação em tempo real. Ambos os aspectos se tornam ainda mais relevantes quando envolvem terceirização de produtos, como destacado por [1].

De fato, a estimativa do tempo de usinagem desempenha uma posição fundamental em várias metodologias para formação do custo total de produtos fabricados [2]. Na década de 1990, a engenharia simultânea ou concorrente ganhou notável atenção, com trabalhos como [3] contribuindo significativamente para o seu desenvolvimento. Durante esse período, muitos estudos exploraram técnicas de estimativa de custos baseadas em características de usinagem e processos. Essas técnicas visavam fornecer informações precoces sobre o ciclo de vida do produto, principalmente utilizando informações geométricas e topológicas extraídas de modelos CAD [4–6].

Nesse contexto, o reconhecimento automático de características de usinagem tem sido um desafio de pesquisa por mais de duas décadas [7]. Abordagens baseadas em grafos foram amplamente adotadas devido à sua semelhança estrutural intrínseca com sólidos no formato B-Rep, que é o padrão mais utilizado na indústria para a representação de peças. Essa tendência remonta ao trabalho pioneiro de [8], que introduziu o conceito de Grafo de Adjacência de Atributos (AAG). Posteriormente, várias metodologias exploraram o uso dessa estrutura para representar modelos CAD 3D e suas características, abrangendo tanto faces quanto arestas [9, 10].

À medida que a tecnologia avançou, os métodos de aprendizado profundo se tornaram cada vez mais relevantes na tarefa de reconhecimento de características de usinagem, que essencialmente é uma tarefa de segmentação semântica. Em particular, a utilização

de redes neurais convolucionais 3D abriu numerosas oportunidades nesse domínio. Autores propuseram uma ampla variedade de modelos de referência, cada um empregando diferentes representações de dados de entrada, como *voxels* [11], *nuvens de pontos* [12], *malhas* [13] e *representação de bordas* [14].

Com o avanço das redes neurais convolucionais em grafos, inicialmente propostas por [15] e posteriormente otimizadas por [16], essa tecnologia emergiu como um campo promissor no processamento de modelos CAD. O trabalho de [17] foi o primeiro a propor a adoção dessa tecnologia para realizar tarefas de reconhecimento de características de usinagem em modelos CAD 3D. O algoritmo proposto operava em um grafo de adjacência porém possuía a limitação de efetuar o reconhecimento em modelos que contivessem apenas faces planas. Este trabalho se baseou nos desenvolvimentos de [18] em relação à Rede Neural Convolucional de Grafo Dinâmico (DGCNN).

Nesse contexto, este estudo apresenta um modelo de aprendizado profundo convolucional, baseado em grafos (ConvGNN) para a estimativa do tempo de usinagem de peças prismáticas de baixa complexidade.

1.1 Justificativa

A estimativa de tempo de fabricação de um produto usinado é um fator relevante para a tomada de decisão em ambas as pontas da cadeia produtiva. Seja por parte da equipe de desenvolvimento de produto que otimizaria a etapa de criação sabendo o impacto das decisões no custo de fabricação, ou então por parte dos fabricantes, que necessitam dessa informação para estimar o custo de seus produtos. Além disso o tempo de fabricação é crucial para decisões estratégicas de terceirização ou produção interna (*make-or-buy)* para ambas as partes da cadeia [19].

O tempo de processo é um elemento essencial na determinação do preço final de um produto usinado. A escolha dessa grandeza é justificada tanto pela sua relevância quanto pelo fato de que, como destacado por [20] em seu estudo, ela pode servir como base para o cálculo do custo de produção de uma peça usinada, independentemente de fatores externos ou da localização geográfica.

A falta de uma ferramenta prática e acessível cria um problema sistêmico de ineficiência na cadeia de suprimentos, conhecido como **pesquisa de orçamentos**. Seja para fins de compra ou tomada de decisões, a maneira convencional de obter informações relativamente precisas sobre o tempo de processo, adotada pela indústria, envolve a con-

sulta a especialistas que dedicam a maior parte de seu tempo produtivo à análise de requisitos para emitir uma cotação.

Durante o processo de estimativa, é possível identificar dois extremos que englobam a grande maioria dos casos: a estimativa baseada na intuição e no conhecimento prévio, e a cotação parametrizada, frequentemente realizada por meio de um software de simulação CAD/CAM. O primeiro método se baseia na experiência empírica de um profissional para gerar uma estimativa subjetiva e muitas vezes imprecisa. Embora seja amplamente utilizado, esse método apresenta uma grande variação entre diferentes interpretações e está sujeito a influências não técnicas. Por outro lado, o segundo método frequentemente envolve um tempo de processamento e custos operacionais elevados, frequentemente inviável especialmente quando aplicado à fabricações de baixa complexidade e menor valor agregado.

1.2 Objetivos

Visando alcançar uma melhor relação entre esforço e acurácia, superando a lacuna entre a estimativa baseada na experiência de profissionais e o cálculo preciso de tempo de máquina por meio de ferramentas de simulação, este trabalho se propõe a investigar e implementar tecnologias de Aprendizado Profundo (*Deep Learning)* para a estimativa de tempo de processo. O objetivo é estabelecer uma base que sirva como referência para uma elaboração detalhada mas ao mesmo tempo ágil do preço final das peças, a partir de arquivos 3D CAD. Nesse trabalho são propostas as seguintes contribuições:

- Apresentar os principais conceitos que impactam o tempo de máquina de uma peça usinada prismática.
- Apresentar um estudo sobre a tecnologia de redes neurais profundas afim de embasar a escolha arquitetural.
- Apresentar uma revisão das principais aplicações de algoritmos de aprendizado profundo no âmbito da usinagem.
- Criar um arcabouço para geração de modelos sintéticos e tempo de processo que possa servir para treinar algoritmos de aprendizado de máquina
- Propor um algoritmo para a estimativa de tempo de processo, utilizando redes neurais gráficas, que possa evidenciar o potencial dessa representação no âmbito de processamento de modelos CAD

1.3 Metodologia

Após a conclusão dos estudos teóricos, almeja-se obter uma compreensão clara da definição da arquitetura e das variáveis independentes do modelo. Com base nessa definição, será conduzido um experimento prático composto por três etapas.

Na etapa inicial deste estudo, começamos gerando 130 modelos sintéticos de peças usinadas prismáticas. Esses modelos são posteriormente processados por meio de um sistema especialista, que calcula o tempo de usinagem estimado com auxílio da interação humana. Ao mesmo tempo, extraímos dados relacionados ao volume total de material removido com base no volume envolvente (*bounding box*). Para garantir a comparabilidade, tanto o tempo de usinagem quanto o volume removido do componente são normalizados.

Na segunda fase, o modelo CAD é transformado em um Grafo de Adjacência Atribuído (AAG) para servir como entrada em uma Rede Neural Convolucional Gráfica (ConvGNN). A estratégia de convolução adotada é a convolução espectral, implementada com o uso de polinômios de Chebyshev. O objetivo desta etapa é criar uma representação abstrata e significativa do modelo, a ser empregada em uma tarefa de regressão.

Na terceira e última fase, os vetores de características obtidos a partir da rede convolucional gráfica são fundidos com informações globais da peça. Essa combinação é então utilizada como entrada para múltiplas camadas de perceptrons (MLP). Nessa etapa, conduzimos uma etapa final de regressão para determinar o tempo de usinagem.

1.4 Estrutura do texto

O capítulo 2 oferece uma revisão abrangente dos conceitos fundamentais relacionados à usinagem de peças prismáticas. Especificamente, aborda tópicos envolvendo a cinemática dos processos de fresamento e furação, os quais desempenharão um papel essencial na construção teórica do cálculo analítico do tempo de processo. Além disso, este capítulo se propõe a introduzir o conceito de "características usináveis"no contexto da interpretação de modelos CAD para a fabricação. Ele também explora as metodologias convencionais empregadas na determinação do custo de peças usinadas, evidenciando que a maioria dessas metodologias requer o cálculo do tempo de máquina como componente fundamental.

Em seguida, o capítulo 3 apresenta uma revisão dos conceitos fundamentais dos algoritmos de aprendizado de máquina, com ênfase nas redes neurais convolucionais. Ele também destaca a evolução recente no campo, especialmente no processamento de dados

tridimensionais, e introduz o conceito de convolução em redes neurais gráficas.

O capítulo 4 é dedicado exclusivamente à exploração e discussão dos conceitos relacionados às redes neurais gráficas.

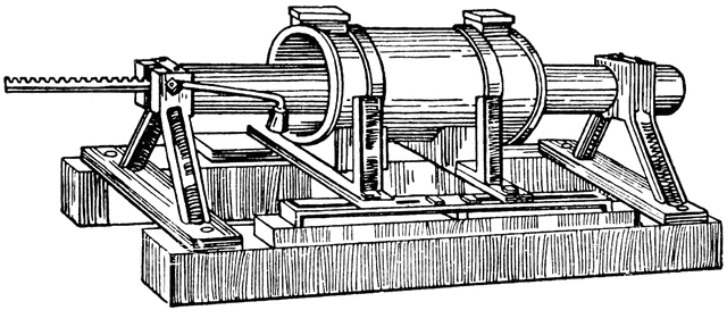
O capítulo 5, o último da revisão bibliográfica, tem como objetivo apresentar os recentes avanços nas aplicações de aprendizado profundo, concentrando-se principalmente no contexto da usinagem. Aborda temas essenciais, como o reconhecimento automático de características usináveis e a estimativa de custos de peças usinadas.

Após a apresentação teórica, o capítulo 6 introduz o modelo proposto, os dados, simulações e discussões dos resultados. Já o capítulo 8 aborda as conclusões e sugere possíveis direções para pesquisas futuras.

2 FUNDAMENTOS DA USINAGEM

A história do corte de metal remonta à segunda metade do século XVIII. Antes desse período, as ferramentas de máquina não existiam. Em 1776, James Watt enfrentou dificuldades ao tentar criar a primeira máquina a vapor funcional devido ao problema de fazer um cilindro de metal que não vazasse vapor. Foi John Wilkinson quem finalmente resolveu o problema ao inventar a máquina de perfuração horizontal, hoje conhecida como mandriladora, tornando possível para James Watt desenvolver uma máquina a vapor bem-sucedida. Essa máquina de perfuração foi a primeira ferramenta de máquina eficaz. O corte de metal, como o conhecemos hoje, começou com a introdução dessa inovação. Hoje, as ferramentas de máquina formam a base de nossa indústria e são usadas direta ou indiretamente na fabricação de todos os produtos da civilização moderna [21].

Figura 1: Mandriladora de Wilkinson



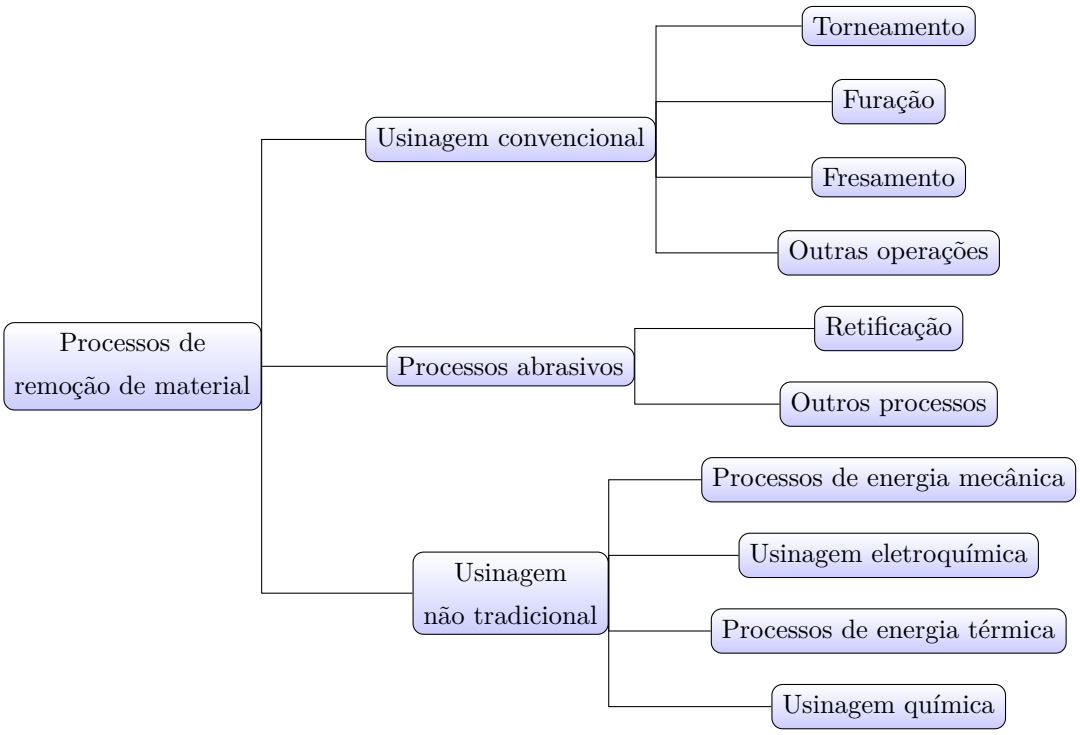
Fonte: extraído de [22]

A usinagem é categorizada como um processo de fabricação de remoção de material, o que significa dizer que suas operações retiram o excesso do material bruto inicial para que a forma resultante seja a geometria desejada. As peças usinadas podem ser divididas em duas categorias predominantes: aquelas que giram, com formatos cilíndricos ou de discos, sendo produzidas por meio de operações onde uma ferramenta de corte retira material de uma peça em rotação; e as peças **prismáticas** que possuem formas mais retangulares ou em formato de placa e são fabricadas por meio de movimentos lineares da peça, acom-

panhados de movimentos rotativos ou lineares da ferramenta de corte. Neste contexto, a usinagem representa o método utilizado para dar forma a peças metálicas, essas operações de corte são mais comumente aplicadas a metais sólidos, utilizando ferramentas de corte mais duras e resistentes do que o metal de trabalho [23].

Existem diversas operações de usinagem, cada uma com a capacidade de criar diferentes geometrias e texturas de superfície. Neste contexto, destacamos três das mais comuns: torneamento, perfuração e fresamento. No torneamento, uma única aresta de corte é utilizada para remover material de uma peça giratória e criar uma forma cilíndrica. A perfuração, por sua vez, cria furos redondos usando uma ferramenta rotativa com duas bordas de corte. Já o fresamento envolve uma ferramenta com múltiplas bordas de corte, que é alimentada lentamente através do material de trabalho para gerar superfícies planas. Além dessas, existem outras operações de usinagem convencionais, como modelagem, plainagem, brochamento e serramento. Operações abrasivas, como a retificação, também são comuns e usadas para melhorar a qualidade da superfície das peças após as operações de usinagem convencionais [23]. A figura 2 relaciona os processos de remoção de material definidos em suas classes.

Figura 2: Classificação dos processos de remoção de material



Fonte: elaborado pelo autor

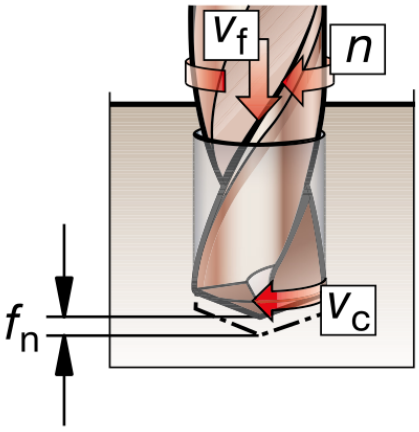
Os fundamentos da usinagem são baseados em princípios de cinemática, que é o es-

tudo do movimento sem considerar as forças que causam esse movimento. Na usinagem, a cinemática é crucial para entender como o movimento da ferramenta e da peça de trabalho interagem para produzir uma forma desejada. As próximas seções se dedicam a apresentar as especificidades dos processos de fresamento e furação cujos conceitos cinemáticos impactam diretamente no tempo de usinagem que é objeto central do trabalho.

2.1 Furação

A furação (*drilling)* é uma operação de usinagem que cria cavidades redondas em peças de trabalho. Isso é feito com uma ferramenta cilíndrica rotativa chamada broca, que possui duas bordas de corte. A broca gira e penetra na peça de trabalho estacionária, formando um furo com o mesmo diâmetro que o seu. Esta operação é frequentemente realizada em uma furadeira, mas também pode ser executada em outras máquinas. Diferencia-se da operação de mandrilamento (*boring)*, que é usada para aumentar o diâmetro de furos existentes. A velocidade de rotação e a taxa da avanço da ferramenta, apresentados na figura 3, são as principais componentes da determinação do tempo do processo de furação [23].

Figura 3: Principais componentes cinemáticas da furação



Fonte: extraído de [24]

A velocidade de corte na operação de furação é definida pela velocidade na superfície do diâmetro da broca, sendo ajustada para determinar a velocidade rotacional da broca, ou seja, o número de rotações por minuto (RPM). Matematicamente, a velocidade do eixo N é calculada como:

$$N = \frac{v}{\pi D} \tag{2.1}$$

onde v representa a velocidade de corte em mm/min, e D é o diâmetro da broca em mm. É importante ressaltar que, embora a velocidade de corte seja especificada na superfície externa da broca, a maior parte do corte real ocorre a velocidades menores, mais próximas ao eixo de rotação. Por outro lado, a taxa de avanço na perfuração é especificada em mm por revolução (mm/rev). Recomenda-se que o avanço seja proporcional ao diâmetro da broca, com avanços maiores para brocas de maior diâmetro. A relação entre a taxa de avanço f_r e o avanço f é dada por:

$$f_r = nf \tag{2.2}$$

A taxa de avanço determina quão rápido a broca penetra no material a cada revolução. A profundidade de cada corte é metade do avanço, pois as brocas normalmente possuem dois gumes de corte ($n = 2$). Os furos realizados pelo processo descrito podem ser classificados como furos passantes, onde a broca atravessa completamente a peça, e furos cegos, onde não. O tempo de usinagem para um furo passante é calculado levando em consideração a espessura do material t e uma distância de aproximação A, que compensa a angulação da ponta da broca, conforme a fórmula:

$$T_m = \frac{t + A}{f_r} \tag{2.3}$$

Na operação de furação, a Taxa de Remoção de Material (TRM) é uma medida do volume de material que é removido por unidade de tempo durante o processo de perfuração. A TRM é importante pois influencia a eficiência do processo de usinagem, o desgaste da ferramenta e a qualidade do furo. A Taxa de Remoção de Material (TRM) é calculada pela seguinte fórmula:

$$TRM = \frac{\pi D^2}{4} \times f \times N \tag{2.4}$$

Onde:
D é o diâmetro da broca
f é o avanço por revolução (em mm/rev ou in/rev)
N é a velocidade de rotação da broca (em RPM)

Essa fórmula considera a área da seção transversal do furo, que é calculada como $\frac{\pi D^2}{4}$,

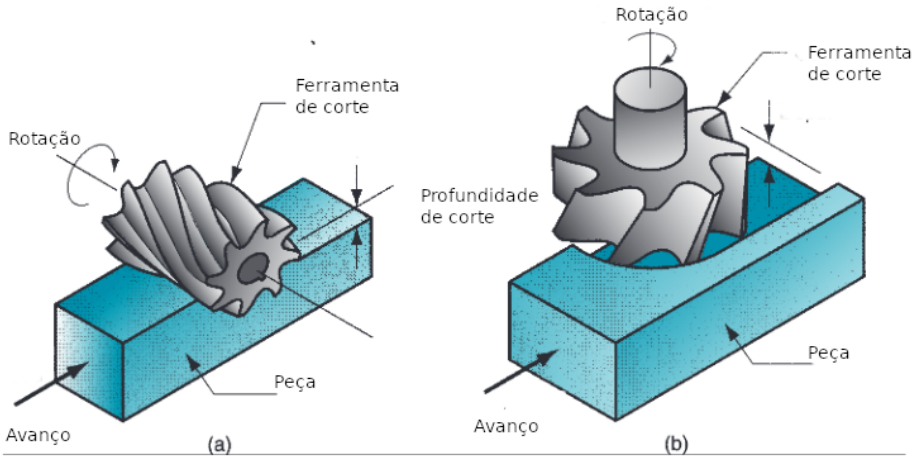
e multiplica pelo avanço total por minuto, que é o produto do avanço por revolução pelo número de rotações por minuto.

2.2 Fresamento

O fresamento é uma operação de usinagem em que uma peça de trabalho é movida em relação a uma ferramenta cilíndrica rotativa com várias bordas de corte. Isso difere da perfuração, onde a ferramenta é alimentada paralelamente ao seu eixo de rotação. O fresamento cria principalmente superfícies planas, mas outras geometrias podem ser obtidas controlando o trajeto da ferramenta. É uma das operações de usinagem mais versáteis e amplamente utilizada. No entanto, o fresamento é um processo de corte interrompido, expondo as ferramentas a forças de impacto e choque térmico, o que requer cuidados no design das ferramentas [23].

Existem dois tipos principais de fresamento, o fresamento periférico e o fresamento frontal, como pode ser visto na figura 4. O fresamento periférico, também chamado de fresamento plano, ocorre quando o eixo da ferramenta é paralelo à superfície que está sendo usinada. Este método utiliza as arestas de corte na periferia externa da fresa, sendo mais adequado para operações de acabamento devido à alta rotação e baixa velocidade de avanço, resultando em uma superfície mais suave. Por outro lado, o fresamento frontal acontece quando o eixo da fresa é perpendicular à superfície sendo usinada, envolvendo as arestas de corte na extremidade e na periferia externa da fresa. Geralmente utilizado em operações de desbaste, este método possui uma rotação menor, mas uma velocidade de avanço maior devido à maior área de contato entre a ferramenta e a peça. É empregado quando a rugosidade da superfície não é tão crítica ou como um desbaste inicial antes da operação de acabamento.

Figura 4: Principais tipos de fresamento: (a) periférico e (b) frontal

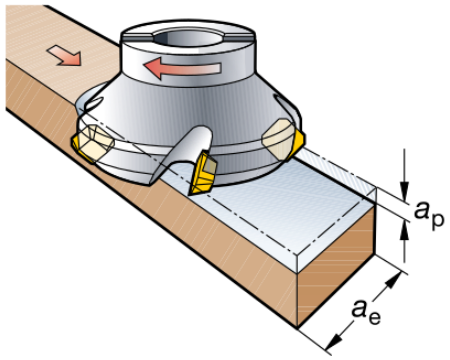


Fonte: extraído de [23]

2.2.1 Parâmetros de corte

A profundidade de corte (a_p) e a largura de corte (a_e) são parâmetros definidos pela rotação do *spindle* e pela geometria da ferramenta, como mostra a figura 5. A a_p é a profundidade na qual a ferramenta penetra no material da peça. Aumentar a a_p eleva o volume de material removido, mas também pode aumentar as forças de corte e impactar a qualidade da superfície. A a_e é a extensão da interação entre a ferramenta e a peça ao longo da superfície de corte. Uma a_e maior implica na remoção de mais material por revolução da ferramenta, afetando a taxa de remoção de material e as forças de corte.

Figura 5: Parâmetros de corte de fresamento



Fonte: extraído de [23]

2.2.2 Avanço no fresamento

O avanço, representado como f_n, é a medida do deslocamento relativo entre a ferramenta de corte e a peça durante uma única rotação. Sua importância reside na capacidade de influenciar o acabamento superficial resultante do processo de fresamento. Em situações em que a ferramenta possui várias arestas de corte, o valor do avanço pode ser determinado usando a seguinte fórmula:

$$f_n = f_z \cdot z_C \tag{2.5}$$

Onde z_C é o número de dentes ou arestas de corte, e f_z é o avanço por dente, que corresponde ao percurso de cada dente, medido na direção do avanço da ferramenta. Quando são conhecidos os valores da velocidade de rotação e do avanço, é possível calcular o valor do avanço por minuto (V_f), que corresponde ao deslocamento da ferramenta em relação à peça, medido em unidades de distância por unidade de tempo. Também é comumente referido como avanço da mesa, avanço da máquina ou velocidade de avanço. A equação para o cálculo do V_f é a seguinte:

$$V_f = f_z \cdot z_C \cdot n \tag{2.6}$$

Onde:
f_z é o avanço por dente
z_C é o número de dentes ou arestas de corte
n é a velocidade de rotação

O avanço por dente (f_z) representa a distância linear percorrida pela ferramenta enquanto um dente específico está realizando o processo de corte. Esse valor é necessário para a definição do avanço da mesa durante a operação de fresagem. O cálculo do f_z pode ser realizado utilizando a seguinte expressão:

$$f_z = \frac{V_f}{n \cdot z_C} \tag{2.7}$$

Onde:
f_z é o avanço por dente.
V_f é o avanço por minuto.
n é a velocidade de rotação.
z_C é o número de dentes ou arestas de corte.

2.2.3 Equações para Estimativa do Tempo de Ciclo

O tempo de ciclo refere-se ao tempo necessário para completar uma operação de usinagem, desde o início até a conclusão. No contexto do fresamento, o tempo de ciclo é frequentemente determinado com base na taxa de remoção de volume e nas dimensões da peça de trabalho [23].

2.2.4 Taxa de Remoção de Volume (Q)

A taxa de remoção de volume é uma medida da quantidade de material removido por unidade de tempo e é comumente expressa em cm^3/min. A fórmula básica para calcular Q é:

$$Q = a_p \cdot a_e \cdot v_f \tag{2.8}$$

Onde:
Q é a taxa de remoção de material
a_p é a profundidade de corte
a_e é a largura de corte
v_f é a velocidade de avanço

2.2.5 Tempo de Ciclo (Tc) para Fresamento

$$Tc = \frac{\text{Volume}}{Q} \tag{2.9}$$

Estas equações fornecem uma estimativa do tempo necessário para completar a operação de usinagem com base nas dimensões da peça de trabalho e nos parâmetros de corte selecionados [25]. Além das equações básicas, várias outras considerações podem influenciar o tempo de ciclo real. Isso inclui tempos de troca de ferramenta, tempos de reposicionamento do material (tombos), pausas para verificação de qualidade, tempos de aceleração e desaceleração da máquina, entre outros. Portanto, é importante considerar esses fatores adicionais para obter uma estimativa precisa [23].

2.3 Definição de Características

No contexto da engenharia de manufatura e do design assistidos por computador, a noção de *característica usinável* é fundamental para o raciocínio geométrico e semântico necessários na elaboração da estratégia de manufatura. Contudo, a definição deste termo é notavelmente diversa, refletindo a variedade de suas aplicações e a ausência de um consenso universal sobre sua interpretação. Esta diversidade é um reflexo direto das necessidades e interesses específicos dos pesquisadores e profissionais da área. Na prática, as características deveriam ser empregadas como conceitos de engenharia para facilitar a compreensão, o manuseio e a comunicação entre equipes de diferentes áreas. Porém, em muitos casos, a definição de uma característica é adaptada para atender aos requisitos de um projeto ou pesquisa específicos, evidenciando a flexibilidade e a adaptabilidade do conceito em diferentes contextos de engenharia e *design* [26].

Segundo [7] as características em um modelo CAD podem ser definidas de duas formas distintas. A definição implícita de característica, frequentemente referida como *característica de forma*, proporciona uma compreensão conceitual de padrões em uma peça ou um produto. Esta abordagem é abrangente e inclusiva, mas demasiada subjetiva para ser diretamente implementada por algoritmos. Sob uma perspectiva mais ampla, as *características de forma* são descritas como ”regiões de interesse”na superfície de uma peça, sendo úteis em várias aplicações. Estas características podem ser definidas como uma única face ou um conjunto de faces conectadas que apresentam combinações de características de diversas naturezas e possuem um significado semântico **coeso**. Por outro lado, a definição explícita de característica se concentra nas particularidades de um domínio específico e devem conter um conjunto de informações a fim de criar uma decomposição em sub classes. Esse conjunto de informações é definido com base nos padrões descritivos de topologia e geometria, como ortogonal/não ortogonal, poliédrico/cilíndrico, interior/exterior, convexo/côncavo, superfície/volume e assim por diante. Exemplos incluem características de usinagem, forjamento e fundição. O objetivo principal ao definir explicitamente uma característica é representar de forma única uma região de interesse para um domínio específico e que seja possível gerar uma representação em um formato computacional. Nos sistemas de reconhecimento de características de usinagem existentes, a definição explícita normalmente inclui uma representação de forma (como plana ou cilíndrica) e uma representação paramétrica (como ângulo ou dimensões) que é utilizada como dado de entrada para o posterior processamento.

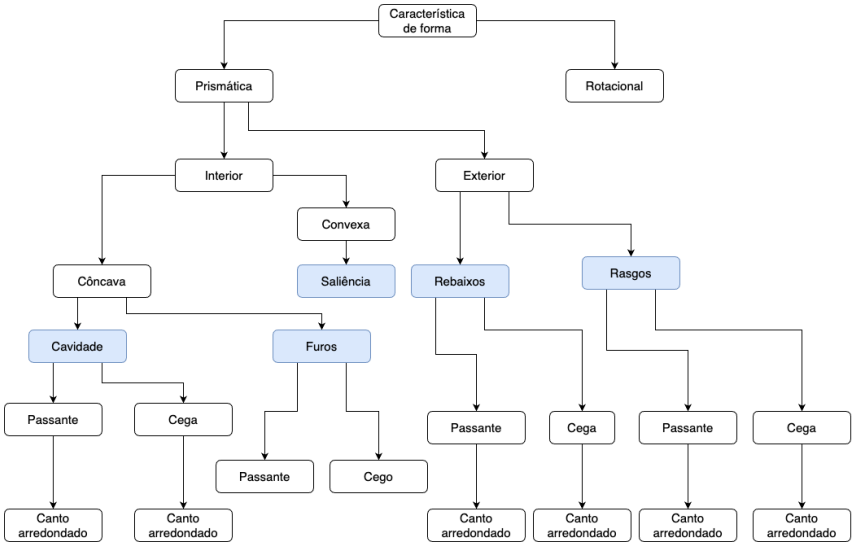
2.3.1 Classificação de características de manufatura

Características de usinagem são definidas como elementos distintos em uma peça de trabalho que necessitam de operações específica para sua fabricação, alguns exemplos são: furos, ranhuras e rebaixos. Esses elementos são tipicamente encontrados em modelos CAD dos projetos de peças. No reconhecimento de características, é vital considerar tanto a geometria quanto a topologia para garantir que todas as operações de usinagem necessárias sejam identificadas e adequadamente planejadas. Informações geométricas referem-se às dimensões físicas e à forma das características, como protuberâncias ou depressões. Características topológicas, entretanto, descrevem a conectividade entre elementos geométricos, como a relação entre faces, arestas e vértices. Um exemplo clássico de uma característica geométrica é o furo cilíndrico, que pode ser descrito por seu diâmetro e profundidade. A relação entre o furo e outras características na peça, como ranhuras ou cavidades adjacentes, determinam sua característica topológica [27].

Segundo [28], uma característica de manufatura é definida como um componente físico de uma peça que pode ser mapeado para uma forma genérica e que tem importância em termos de engenharia. Baseado nesse conceito, foi apresentado uma abordagem sistemática para extrair características de fabricação de sistemas CAD. Foi explorada uma taxonomia lógica para identificar características a partir de informações geométricas, facilitando assim a integração entre *design* e manufatura.

A classificação proposta divide as características em prismáticas e rotacionais, cada qual com suas subcategorias, baseadas em características geométricas específicas. As características prismáticas são divididas em interiores ('*Interior Form Feature*') e exteriores ('*Exterior Form Feature*'). As características de forma interior podem ser côncavas ou convexas, incluindo cavidades e furos, enquanto as características de forma exterior englobam degraus e ranhuras. Essas categorias são então diferenciadas pela presença de cantos arredondados e se a característica é passante (*through*) ou cega (*blind*). A figura 6 ilustra a hierarquia de classificação das características proposta. Essa estrutura hierárquica detalhada visa um planejamento de processo mais sistemático e organizado, que potencialmente pode contribuir para a fabricação eficiente e precisa de componentes [28].

Figura 6: Hierarquia das características de forma



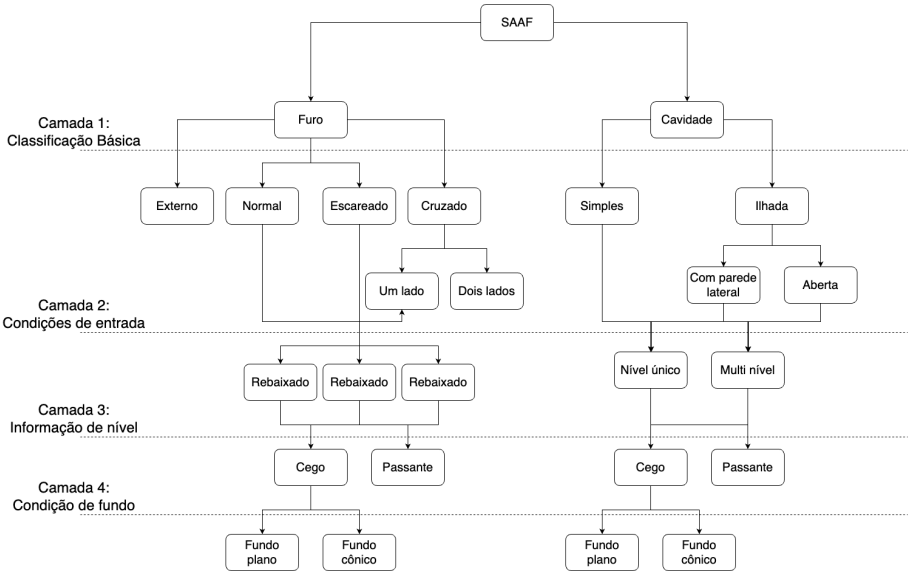
Fonte: extraído de [28]

Embora essa classificação tenha proporcionado uma fundação robusta para o entendimento das características usináveis, a abordagem encontrada em [29] expande este entendimento ao incorporar a acessibilidade das ferramentas como um critério chave. Na taxonomia proposta, as características são classificadas em função da facilidade de acesso das ferramentas. Segundo essa metodologia, no primeiro nível, as características são divididas em:

2.3.1.1 Características Acessíveis por um Único Eixo (*SAAF*)

Essas são características que permitem acesso de ferramentas paralelo a um único eixo coordenado, como furos inclinados e cavidades. Essas características são detalhadas em uma estrutura de classificação em camadas, considerando condições de entrada, níveis de informação como furos com rebaixo ou cavidades de vários níveis, e condições de fundo. A Figura 8 mostra a classificação proposta para esse tipo de característica.

Figura 7: Estrutura de classificação das características acessíveis por um único eixo

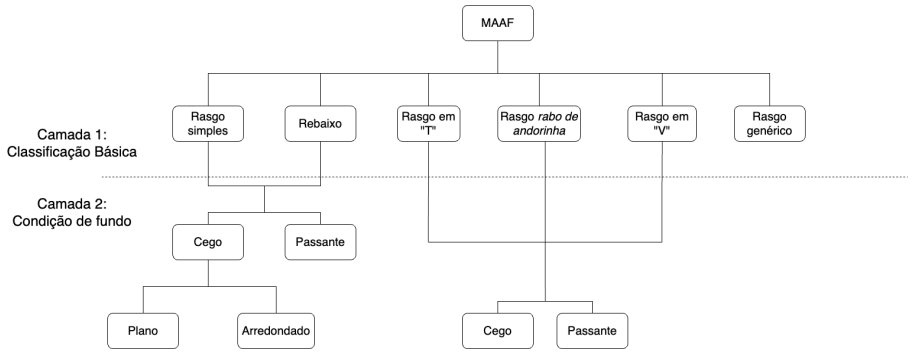


Fonte: extraído de [29]

2.3.1.2 Características acessíveis por múltiplos eixos (*MAAF*)

Características que proporcionam pelo menos duas possíveis direções de acesso de ferramentas paralelas a dois eixos coordenados diferentes, como ranhuras simples.

Figura 8: Estrutura de classificação das características acessíveis por múltiplos eixos

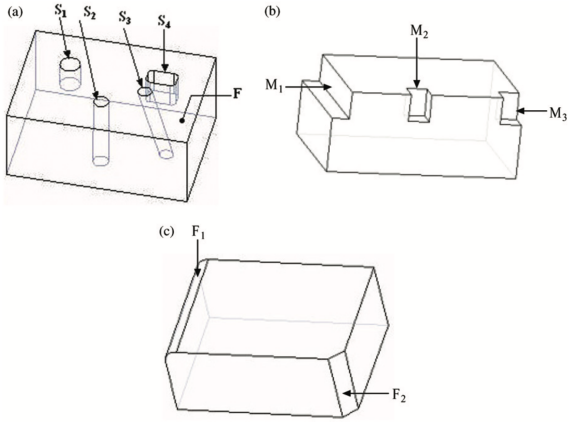


Fonte: extraído de [29]

2.3.1.3 Características de acabamento (*FF*)

Incluem chanfros, arredondamentos e superfícies que devem ser acabadas para se atingir o formato final desejado do produto a partir do material bruto.

Figura 9: (a) Características Acessíveis por um Único Eixo. (b) Características Acessíveis por Múltiplos Eixos. (c) Características de Acabamento.



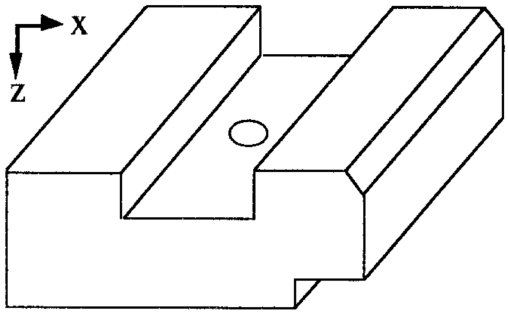
Fonte: extraído de [29]

Esta classificação, que possui representantes ilustrados na Figura 2, oferece um entendimento mais profundo e uma abordagem sistemática para o reconhecimento de características usináveis, com um foco ainda maior para aplicações de automatização de processos fabris, como planejamento e configuração automáticos e possivelmente geração de trajetórias de ferramentas de máquinas de controle numérico.

2.3.2 Características Interagentes

Características de manufatura são definidas como interagentes quando duas ou mais características simples tem seus limites cruzados, formando um volume conectado, criando assim um padrão geométrico mais complexo. Por exemplo, um furo que atravessa uma ranhura cria uma interseção entre duas características distintas como ilustrado na Figura 10. Características interagentes ou compostas, representam um desafio adicional no campo do reconhecimento de características. Uma das principais dificuldades no reconhecimento de características interagentes é a necessidade de interpretar corretamente a relação entre as características individuais e como elas afetam o processo de manufatura [30].

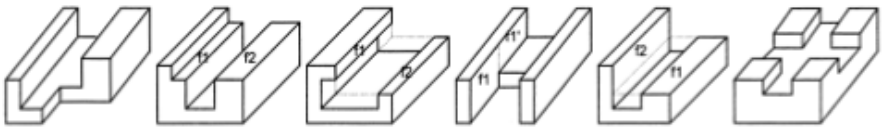
Figura 10: Furo atravessando uma ranhura



Fonte: extraído de [29]

Segundo [10] as interações entre características são classificadas em seis categorias, baseadas em três tipos de variações topológicas: fusão de faces, perda de arestas côncavas e divisão de faces. Faces fundidas são aquelas compartilhadas por mais de uma característica. Para abranger completamente todas as combinações das três condições acima, seriam necessários oito tipos de interações de características. No entanto, duas combinações são impossíveis, pois tanto a face dividida quanto a aresta côncava perdida não podem ser simultaneamente verdadeiras, o que reduz os tipos possíveis para seis, conforme mostrado na Figura 11

Figura 11: (a) Tipo I (sem separação ou perda) — (b) Tipo II (divisão da face) — (c) Tipo III (perda de aresta) — (d) Tipo IV (junção de faces) — (e) Tipo V (perda de aresta & junção de faces) — (f) Tipo VI (junção de faces de fundo & divisão de paredes)



Fonte: extraído de [10]

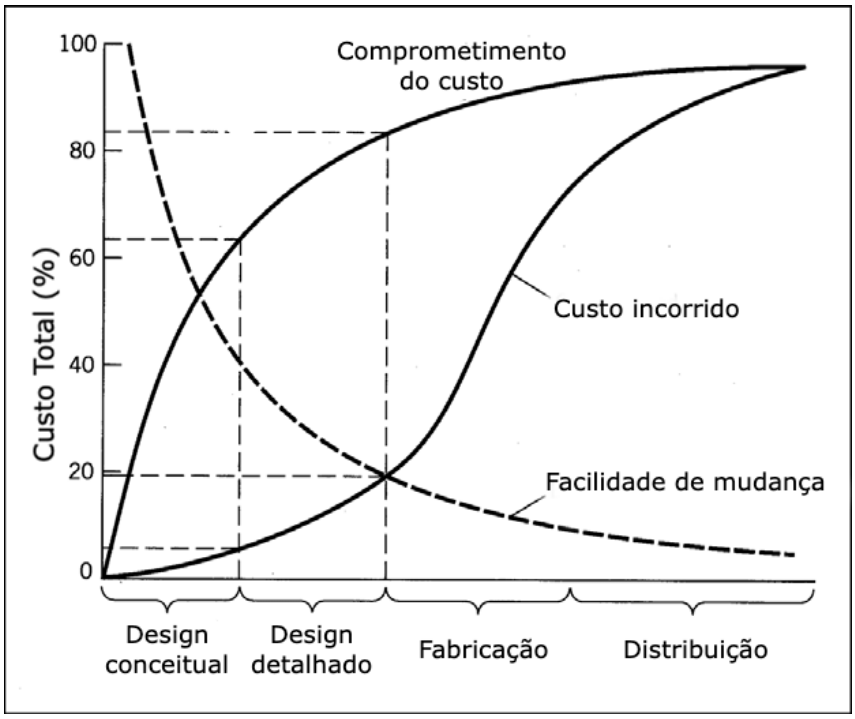
Como veremos ao longo dos próximos capítulos, as abordagens modernas para reconhecimento de características envolvem técnicas avançadas que podem analisar e interpretar dados de modelos CAD para identificar e classificar corretamente essas interações complexas. É possível constatar que grande parte dos avanços em pesquisa na área de reconhecimento automático de características de usinagem visam tornar os algoritmos suficientemente inteligentes para enfrentar esse tipo de desafio.

2.4 Estimativa de custos para peças usinadas

A estimativa de custos de fabricação de peças usinadas representa um desafio para ambos os lados da cadeia produtiva. Tanto para empresas que desenvolvem produtos quanto para aquelas que prestam serviços de usinagem. a estimativa de custos eficiente é um elemento que influencia as decisões estratégicas e operacionais, moldando o sucesso no mercado de manufatura.

Para empresas que desenvolvem produtos, a precisão na estimativa de custos é essencial para a viabilidade econômica, competitividade no mercado e sustentabilidade financeira, uma vez que a etapa de manufatura possui grande impacto no custo total do produto e problemas inesperados de manufaturabilidade podem comprometer todo o projeto. A habilidade de prever os custos associados à usinagem, ainda na fase de concepção, permite o desenvolvimento de produtos que não apenas atendam aos requisitos técnicos e de qualidade, mas também sejam economicamente viáveis. O gráfico da figura 12 relaciona, entre as diversas etapas do processo de produção, a parcela do comprometimento com o custo e o custo de fato que cada uma dessas etapas representa. Embora o custo da etapa de design da peça represente apenas cerca de 5% do custo total do desenvolvimento do produto, ele determina de 60% a 70% do custo total das partes mecânicas [31].

Figura 12: Comprometimento do custo total por etapa de produção



Fonte: extraído de [32]

Por outro lado, prestadores de serviços de usinagem enfrentam o desafio de elaborar orçamentos com relativa precisão, em um ambiente altamente competitivo. Muitas vezes, investindo recursos significativos na elaboração de propostas, mas se deparando com uma baixa taxa de conversão em contratos efetivos. Isso não apenas impacta a eficiência operacional, mas também coloca pressões financeiras consideráveis sobre essas empresas [33].

O propósito desta seção é fornecer uma visão abrangente sobre as técnicas **clássicas** de estimativa de custos para peças usinadas mas com um enfoque na importância do tempo de processo. Como poderá ser constatado, grande parte dos estudos de modelamento de custo trazem propostas de cálculo de tempo de processo.

2.5 Metodologias de Estimativa de Custos

A classificação entre metodologias de estimativa de custo na indústria mais aceita atualmente é a sugerida por [2]. Estas metodologias são divididas em abordagens **qualitativas** e **quantitativas**. As análises qualitativas são divididas nos grupos das técnicas intuitivas e das técnicas analógicas (no sentido de analogia). As análises quantitativas também possuem duas subdivisões denominadas técnicas paramétricas e técnicas analíticas. Esses subgrupos ainda podem ser novamente subdivididos em metodologias específicas que podem ser utilizadas em diferentes aplicações e momentos do ciclo de vida do produto. A tabela 1 apresenta um resumo da classificação proposta pelo autor.

Os métodos analíticos e paramétricos focam nos custos específicos de cada elemento do processo de manufatura, enquanto os métodos intuitivos utilizam a similaridade com produtos anteriores como base para a estimativa. Já os métodos de regressão aplicam modelos estatísticos para prever custos com base em variáveis históricas, e o método ABC distribui os custos indiretos com base nas atividades realizadas. Nas seções subsequentes, será discutido como essas metodologias são aplicadas na prática. É importante ressaltar que a taxonomia proposta por [2] abrange processos de fabricação em um sentido mais amplo. Sendo assim, nem todas as metodologias possuem uma aplicação prática e direta para o processo fabril específico de usinagem.

Tabela 1: Estrutura de classificação das metodologias de estimativas de custo

Técnicas de Estimação de Custo de Produto	Vantagens Chave	Limitações
Técnicas Intuitivas de Estimação		
Sistemas Baseados em Casos	Abordagem de design inovador	Dependência de casos passados
Sistemas Baseados em Regras	Pode fornecer resultados otimizados	Consumo de tempo
Sistemas de lógica difusa	Lida com incertezas, estimativas confiáveis	Estimar custos complexos é tedioso
Sistemas de Especialistas	Mais rápido, mais consistente e mais resultados precisos	Programação complexa necessária
Técnicas Analíticas de Estimação de Custo		
Análise de Regressão Modelo	Método mais simples	Limitado para resolver questões de linearidade
Modelo de rede neural Back Propagation	Lida com problemas incertos e não-lineares	Completamente dependente de dados, maior custo estabelecimento
Técnicas Paramétricas de Estimação de Custo	Utiliza de forma efetiva os fatores de custo	
Técnicas Analíticas de Estimação de Custo		
Modelos de custo baseados em operações	Tipos diferentes de planos de processo podem ser avaliados para obter resultados otimizados	Consumo de tempo, requer informações detalhadas
Modelos de custo de decomposição	Método mais fácil	Informação detalhada necessária
Modelos de tolerância de custo	Custo efetivo, tolerâncias de design podem ser identificadas	Requer informação de design detalhado
Modelos de custo baseados em características	Características com maiores custos podem ser identificadas	Limitada eficácia para pequenas e complexas estruturas
Modelos de custo baseados em atividades	Método fácil e eficaz usando custos de atividades unitárias	Requer tempo de espera nos estágios iniciais do design

Fonte: elaborado pelo autor

2.5.1 Metodologias Qualitativas

As metodologias qualitativas representam um conjunto de técnicas utilizadas na indústria de manufatura e algumas delas podem ser utilizadas no contexto de peças usinadas. Essas abordagens são caracterizadas pela utilização de dados e experiências anteriores, conhecimento especializado e comparações analógicas (referindo-se à comparação e análise por similaridade) para formular estimativas de custos. Essas metodologias são particularmente úteis nas fases iniciais do processo de design, onde a informação detalhada pode não estar completamente disponível.

Dentro das metodologias qualitativas, encontram-se os métodos intuitivos, que se baseiam na experiência e no conhecimento acumulado dos especialistas no campo. Esses

métodos dependem da prática e julgamento humanos em vez de fórmulas matemáticas ou modelos estatísticos, o que os torna flexíveis, mas pode resultar em menor precisão e repetibilidade. Os métodos intuitivos incluem a metodologia baseada em casos (*Case-Based Reasoning)*, que utiliza dados de projetos anteriores para informar estimativas de novos projetos, e os sistemas de apoio à decisão (*Decision Support Systems (DSS)*), que podem integrar regras determinísticas e/ou de lógica difusa para fornecer estimativas de custo.

Outra categoria importante dentro das abordagens qualitativas é a de métodos por analogia. Estes métodos empregam técnicas como análise de regressão e redes neurais artificiais para estabelecer relações entre custos passados e variáveis fortemente correlatas, permitindo prever custos futuros em cenários complexos.

2.5.1.1 Case-Based Reasoning

A abordagem baseada em casos, é uma metodologia que se apoia na experiência prática e no conhecimento acumulado ao longo do tempo. Nesse método, um banco de dados contendo casos anteriores de projetos é mantido. Cada caso no banco de dados inclui informações detalhadas sobre características do produto, materiais usados, processos empregados, tempos de processo e *setup*, custos diretos e indiretos, entre outros detalhes relevantes. Quando um novo projeto é apresentado para estimativa de custo, o sistema baseado em casos permite a busca de casos anteriores semelhantes com base nesses critérios. O próximo passo envolve a adaptação das informações dos casos anteriores ao contexto do novo projeto. Isso pode incluir a consideração de diferenças entre os casos anteriores e o novo projeto e a aplicação de ajustes apropriados. A estimativa de custo é gerada com base nessas informações adaptadas [2].

Em [34] o autor descreve a criação de um sistema que utiliza soluções de experiências passadas (casos) para a estimativa de custo de pistões de motor a diesel de trens, envolvendo a escolha e adaptação de uma das soluções (caso fonte) para o novo problema (caso alvo). O sistema implementa técnicas de indexação de casos e medidas quantitativas de similaridade para a recuperação do caso fonte, além de procedimentos de adaptação da solução para o caso alvo. Por exemplo, segundo o artigo, os tempos de usinagem são proporcionais ao volume de material removido.

2.5.1.2 Decision Support Systems (DSS):

Os sistemas de suporte à decisão (DSS) utilizam regras determinísticas ou heurísticas para auxiliar os tomadores de decisão na estimativa de custo. Uma aplicação comum é permitir que profissionais sem total conhecimento em formação de preço realizem essa tarefa. Além disso, a "Lógica Fuzzy"pode ser usada para lidar com incerteza e ambiguidade em dados de estimativa, oferecendo uma representação flexível das variáveis de custo. Os "Sistemas Especialistas"também são parte desses DSS, emulando especialistas humanos para melhorar a precisão e eficiência das estimativas de custo, reduzindo o viés humano [4, 35].

O sistema proposto por [36] utiliza restrições de design e fabricação, representadas como regras para garantir a consistência e viabilidade do design e processo de fabricação. A otimização de processos é alcançada através da análise de características da peça, seleção de processos, máquinas e ferramentas disponíveis. A **estimativa de tempo** e custo do processo é uma parte fundamental do sistema, utilizando fórmulas padrão para calcular o tempo de processo com base no volume das características usináveis e taxa de remoção de material. A análise abrange o custo total de usinagem, incluindo material, ferramentas, despesas gerais e custo de mão-de-obra. Essa abordagem iterativa e baseada em restrições permite um *loop* de iterações até que um produto economicamente viável seja projetado.

A abordagem de lógica fuzzy é aplicada para a estimativa de custos em [37], com o objetivo de superar a incerteza no modelo de estimativa de custo proposto. O artigo traz uma aplicação da lógica fuzzy para a fabricação de furos que utiliza variáveis linguísticas para modelar os principais fatores que afetam o custo. As variáveis de entrada são o diâmetro do furo, a profundidade do furo e o acabamento de superfície, enquanto a variável de saída é o **tempo de usinagem**. O modelo é desenvolvido em cima de conceitos abstratos como "raso", "largo"e "profundo".

O sistema apresentado em [38] compreende uma solução baseada em conhecimento especializado para a otimização da seleção de ferramentas de corte e condições operacionais em usinagem. Utilizando um motor de inferência desenvolvido, o sistema analisa e otimiza a escolha de ferramentas e condições de corte, impactando significativamente na formação do custo de fabricação. O sistema segue diretrizes rigorosas para minimizar o número de ferramentas necessárias, escolhendo aquelas que minimizam o **tempo de processo** ou que reduzem os custos operacionais. A seleção das condições de corte pelo sistema proposto começa com o cálculo das condições ideais para operações de acabamento, com a profundidade de corte de acabamento sendo subtraída da profundidade máxima de corte

para determinar a remoção de material nas operações de desbaste.

2.5.1.3 Métodos Analógicos

Os métodos de estimativa de custo por analogia representam uma abordagem qualitativa para prever os custos de produção com base em **critérios de semelhança** e **processamento de dados históricos**. Esses métodos se fundamentam em modelos estatísticos que utilizam as semelhanças identificadas para estabelecer correlações entre os fatores conhecidos de produtos fabricados anteriormente e os parâmetros do produto atual. Ao combinar dados históricos de técnicas analíticas, os métodos por analogia oferecem uma ferramenta relevante para a estimativa de custos. Essa metodologia é o foco central deste trabalho.

Ainda que exista uma quantidade razoável de estudos que empregam modelos estatísticos para a estimativa de custo de produção, a maioria dos estudos se concentra em processos de fabricação mais uniformizados, como a fundição, exemplificados em trabalhos como [39], ou em produtos específicos e padronizados, tal como abordado por [40,41], que se dedicam exclusivamente a produtos aeronáuticos.

2.5.2 Metodologias Quantitativas

Ao contrário das metodologias qualitativas, que se baseiam em experiências e analogias, as metodologias quantitativas empregam modelos matemáticos e paramétricos para calcular o custo de fabricação de um produto. Estes métodos fornecem estimativas detalhadas considerando vários fatores, como materiais, mão de obra, tempo de máquina e outros recursos diretos. A abordagem quantitativa é amplamente utilizada na estimativa de custos de produtos manufaturados, especialmente em sua vertente analítica, e é aplicável à estimativa de custos de peças usinadas. No entanto, para usar esses métodos com eficácia, são necessárias informações detalhadas sobre o projeto e o processo de manufatura, o que pode aumentar a complexidade do sistema e/ou o tempo de processamento [33]. As metodologias quantitativas podem ser divididas em duas categorias principais:

1. **Metodologias Paramétricas:** Essas metodologias empregam parâmetros específicos do produto ou processos para calcular os custos, utilizando relações estatísticas ou matemáticas entre esses parâmetros e os custos totais.

2. **Metodologias Analíticas:** Essas metodologias englobam técnicas mais elaboradas, como o Custo Baseado em Atividades (*Activity-Based Costing - ABC*), que aloca os custos indiretos com base nas atividades realizadas, além de abordagens que se baseiam em características específicas do produto, como métodos que utilizam características construtivas (*features*) e tolerâncias.

Essas metodologias são mais úteis em estágios avançados de desenvolvimento, onde há muitos detalhes disponíveis. Elas são usadas principalmente para entender os custos de fabricação, melhorar a gestão de recursos e estabelecer preços mais precisos para os produtos.

2.5.2.1 Activity-Based Costing (ABC)

O método ABC (Custeio Baseado em Atividades) é uma abordagem avançada de contabilidade de custos que visa atribuir custos de forma organizada e eficaz. Ele identifica atividades na empresa, determina os fatores que influenciam os custos (*drivers*), aloca custos às atividades com base no consumo de recursos e, em seguida, aloca esses custos aos produtos ou serviços com base no uso dos drivers de custo. Isso ajuda a compreender quais produtos ou serviços demandam mais recursos e permite calcular os custos totais de forma precisa, para aquela organização específica, incluindo custos diretos e indiretos [42, 43].

2.5.2.2 Feature-Based Estimation

O método de estimativa baseado em características é amplamente utilizado na determinação dos custos de usinagem de produtos. Ele estabelece uma conexão direta entre as características geométricas e topológicas da peça e seus custos de produção, envolvendo a identificação de dimensões críticas, geometrias específicas e tolerâncias apertadas. Após essa identificação, selecionam-se os processos de usinagem apropriados, juntamente com materiais e ferramentas adequadas. Os tempos de usinagem são calculados considerando a complexidade das características da peça, e a estimativa de custo abrange tanto custos diretos quanto indiretos relacionados aos processos de usinagem, como energia e manutenção de máquinas [2]. Em [5], é descrito um sistema integrado para estimativa de custos de fabricação com base em características de projeto. O sistema inclui um módulo CAD com base em características, uma biblioteca de processos de fabricação e custos, e um módulo de análise. Esse sistema calcula custos estimados e exibe os resultados para facilitar ajustes no projeto. O artigo também aborda a estimativa de tempo de usinagem em operações de fresamento e perfuração com base em parâmetros geométricos e de

acabamento superficial, fornecendo equações para essa finalidade. Em um estudo posterior, [4] busca modelar o custo de fabricação na fase de concepção do projeto. Além de adotar a metodologia de [5], introduz dois métodos de representação de conhecimento: um baseado em *frames* para organizar informações detalhadas sobre *designs* e objetos, e outro em regras de produção que estabelecem relações lógicas entre elementos e condições por meio de condicionais "se-então". Essas regras são interconectadas e aplicam encadeamento lógico, o que é útil para determinar sequências de operações. O estudo também utiliza lógica difusa para lidar com regras subjetivas, como a modelagem de variáveis linguísticas. Ambos os sistemas propostos em [5] e [4] fazem o reconhecimento de características usináveis a partir da integração com a ferramenta de modelamento, o que limita o uso dessas abordagens para projetos que tenham sido criados sem essa preocupação inicial.

2.5.2.3 Tolerance-Based Methods

A Estimativa Baseada em Tolerâncias é uma abordagem de estimativa de custos de usinagem que se concentra nas tolerâncias dimensionais das peças a serem produzidas. Começa com a análise das tolerâncias no desenho da peça e, com base nelas, identificam-se os processos de usinagem necessários para atender às tolerâncias. A escolha de materiais e ferramentas adequadas é fundamental, e os tempos de usinagem são calculados com base nas tolerâncias. A estimativa de custo abrange materiais, mão de obra direta, tempo de usinagem e custos indiretos. Essa metodologia é especialmente aplicável quando a precisão dimensional é crítica para a qualidade da peça, pois garante que os processos de usinagem sejam selecionados e executados para atender às tolerâncias, evitando retrabalho e garantindo a qualidade final [44].

2.5.2.4 Breakdown Approach

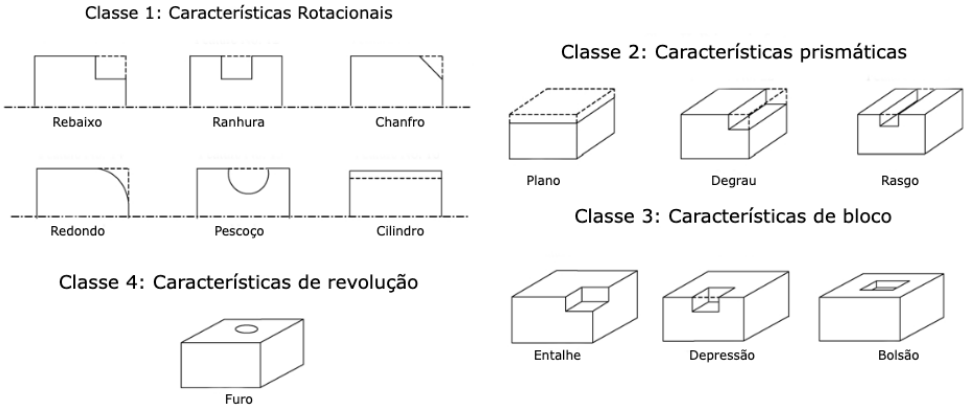
A Abordagem de Decomposição é uma metodologia de estimativa de custos na usinagem que se concentra na análise minuciosa de cada componente do processo, dividindo-o em etapas individuais. Ela identifica materiais, operações, ferramentas e tempo necessário para cada etapa, considerando custos diretos e indiretos. Essa abordagem é valiosa para entender como os recursos são alocados em cada componente do processo, otimizando a eficiência e identificando oportunidades de redução de custos [45, 46].

2.5.2.5 Operation-Based Approach

A Estimativa Baseada em Operações se concentra nas operações de fabricação necessárias para produzir uma peça usinada. Ela identifica e analisa todas as operações de usinagem, seleciona os processos, máquinas e ferramentas adequados, calcula os tempos de usinagem e estima os custos diretos e indiretos. Essa abordagem permite uma análise detalhada dos custos de produção e ajuda a identificar oportunidades de otimização, além de possibilitar a identificação de gargalos na produção e a melhoria do fluxo de trabalho.

Um dos estudos pioneiros, que alcança um equilíbrio bem-sucedido entre o nível de informação necessário e precisão na estimativa do tempo de usinagem, é uma abordagem baseada em características/processos apresentada por [6]. Neste trabalho, o autor classifica as características de usinagem em quatro categorias: rotacionais, prismáticas, de lâmina e rotativas. Cada característica possui um processo específico associado, que pode ser usinagem bruta ou acabamento, juntamente com equações geométrica para cálculo de volume. Aplicando o princípio da taxa de remoção de material (*Material Removed Rate* - MRR), é possível calcular o tempo final somando todos os componentes dos diferentes processos. O autor também leva em consideração o tempo necessário para a aproximação da ferramenta e um fator de alargamento para características rotativas, como furos. O estudo inclui resultados experimentais em uma amostra de peça usinada, demonstrando resultados aceitáveis. A Figura 13 mostra a classificação proposta.

Figura 13: Classes de características



Fonte: extraído de [6]

2.5.2.6 Métodos Paramétricos

As técnicas paramétricas visam equilibrar a complexidade do cálculo com a viabilidade de aplicação precoce e baseiam-se na identificação e quantificação de parâmetros

que têm um impacto significativo no custo total do produto. Estes parâmetros podem variar desde características físicas simples, como peso ou volume, até aspectos mais complexos relacionados aos processos de fabricação. A premissa central dessas técnicas é a existência de uma relação funcional entre o custo e os parâmetros selecionados, permitindo a criação de modelos matemáticos ou estatísticos para prever os custos com base em dados disponíveis.

Especialmente útil em fases de design onde as especificações do produto começam a se tornar mais claras, a precisão dos métodos paramétricos depende da qualidade dos dados de entrada e da validade dos modelos utilizados. Quando bem implementados, esses métodos podem fornecer estimativas rápidas e confiáveis dos custos de produção, permitindo que fabricantes e projetistas tomem decisões informadas sobre design, fabricação e estratégias de precificação.

Especificamente em usinagem, com a exclusão de estudos mais recentes envolvendo aprendizado profundo, um dos poucos estudos que buscam estabelecer uma relação estatística entre custo e seus fatores determinantes é apresentado em [47]. Nesse estudo, o autor desenvolve um modelo que se baseia em dados de **tempo de processo**, seguindo a metodologia descrita em [48], para construir uma regressão utilizando uma rede neural MLP simples. Essa regressão leva em consideração características geométricas da peça, como dimensões, bem como características do material, como usinabilidade, e aspectos do processo, como o número de operações. Embora o autor tenha obtido resultados satisfatórios na tarefa de regressão, o foco principal do trabalho foi investigar os impactos na qualidade do modelo, explorando diferentes aspectos, como a função de ativação, a função de perda e a quantidade de amostras utilizadas no treinamento. O modelo foi desenvolvido especificamente para o processo de torneamento, e e as variáveis de entrada para uma nova estimativa além de numerosas eram de difícil obtenção.

Construindo sobre pesquisas anteriores sobre complexidade, dimensões e tolerâncias apresentadas em [49], um modelo de regressão múltipla foi introduzido no estudo [19]. Este modelo leva em consideração essas grandezas para estimar o tempo de usinagem de forma estatística. Quando comparado a um método analítico baseado em características descrito por [50], a abordagem demonstrou resultados satisfatórios. Em um experimento com um conjunto de dados contendo 80 peças mecânicas, o erro médio encontrado foi de aproximadamente 25%, o que segundo o autor, é considerado adequado para orientar diversas atividades de *design*, incluindo decisões de "fazer ou comprar". Além disso, essa abordagem é considerada menos trabalhosa do que o método de referência.

Mais recentemente, o estudo [51] apresenta uma abordagem que utiliza redes neurais para estimar o tempo de usinagem em sistemas de usinagem CNC para peças de produção complexas. O estudo foi motivado por discrepâncias entre os tempos de usinagem calculados e os tempos reais. As características de entrada consideradas no modelo incluem a complexidade da peça, o número de etapas de usinagem, o tamanho da série de produção, a participação do operador e as paradas de máquina. O estudo explora cinco algoritmos de rede neural diferentes, incluindo BPNN (Rede Neural de Retropropagação), MNN (Rede Neural Multicamada), RBFNN (Rede Neural de Função de Base Radial), GRNN (Rede Neural de Regressão Generalizada) e SOMNN (Rede Neural de Mapa Auto-Organizável), para desenvolver modelos de previsão. BPNN, MNN e RBFNN mostraram baixos erros, com o MNN superando o BPNN na fase de validação.

De fato, trabalhos dessa natureza são menos comuns na literatura, e isso pode ser atribuído tanto à complexidade do modelo, seja em termos de processamento ou arquitetura, quanto à disponibilidade limitada de dados, ou provavelmente a ambos. Esses fatores destacam ainda mais a importância da pesquisa neste campo.

2.6 Tempo de processo

Como destacado na seção 2.4, o tempo de processo é, de fato, o fator de maior impacto na estimativa de custos para peças usinadas. Essa seção se dedica a se aprofundar um pouco mais nesse conceito.

2.6.1 Definição

Segundo [52], o tempo total de usinagem pode ser dividido em dois tipos de elementos: os independentes da quantidade de peças a serem produzidas e os diretamente relacionados a cada peça individualmente. Esses elementos incluem o tempo de configuração, manuseio, usinagem, desmontagem, paralisação e margens. Os elementos produtivos envolvem o corte da peça, enquanto os não produtivos são essenciais, mas não envolvem o corte do metal.

2.6.2 Elementos de Tempo Fixo (Independentes do Nível de Produção)

Tempo de Configuração: Preparação da máquina, incluindo estudo de desenho, obtenção de ferramentas, instalação, alinhamento e produção da primeira peça para

garantir precisão. Ocorre uma vez por lote.

Tempo de Desmontagem: Remoção de ferramentas e limpeza da máquina após um lote. Ocorre uma vez por lote.

Tempo de Paralisação: Tempo perdido devido a falta de operador, quebras, falta de materiais e outras interrupções na produção.

2.6.3 Elementos Relacionados à Peça (Não Produtivos)

Tempo de Manuseio: Operador carrega e descarrega a peça.

Tempo de Retorno da Ferramenta: Devolução da ferramenta à posição inicial, geralmente parte do manuseio.

Tempo de Troca de Ferramenta: Substituição de ferramentas desgastadas, calculado com base na vida útil ou parte das margens.

Margens: Tempo extra para necessidades pessoais, inspeção e outros, frequentemente calculado como uma porcentagem do tempo padrão de operação.

2.6.3.1 Elemento Produtivo

O tempo de usinagem, também conhecido como tempo de processo, refere-se ao intervalo durante o qual a máquina de usinagem está ativamente envolvida na produção da peça. Ele compreende o período desde o momento em que a ferramenta de corte entra em contato com a peça até a conclusão da operação, quando a ferramenta é retirada da peça. Este é o componente mais crítico do tempo total de usinagem e pode ser calculado com base nas taxas de remoção de material discutidas nas seções 2.1 e 2.2.

Neste trabalho, a nomenclatura ”tempo produtivo”será usada de forma intercambiável com ”tempo de usinagem”e ”tempo de processo”para se referir ao período em que ocorre efetivamente o corte de metal. Por outro lado, o termo ”tempo improdutivo”ou ”tampo de *setup*”abrangerá todos os outros intervalos de tempo relacionados ao processo que não envolvem diretamente o corte de metal.

2.6.4 Outras estratégias de cálculo

Além dos métodos abordados nas metodologias de estimativa de custos para peças usinadas, que se baseiam em características construtivas e as baseadas em operação,

existem outras abordagens utilizadas para estimar o tempo de máquina. Estas incluem métodos de simulação, análise de programas de controle numérico (NC), entre outros.

O trabalho apresentado em [53] desenvolveu um modelo analítico para estimativa de tempo de máquina utilizando uma abordagem sucessiva de decomposição temporal. Inicialmente, o tempo de processamento é dividido em cinco fatores: tempo de corte, tempo de movimentação da ferramenta, tempo de troca da ferramenta, tempo de rotação do magazine da ferramenta e um componente de tempo auxiliar associado ao operador da máquina. Cada um desses fatores passa por uma subdivisão adicional para obter uma estimativa precisa. O tempo de corte, como a componente mais significativa, é posteriormente dividido em três fatores, sendo o principal chamado de "tempo de utilização". Essa subdivisão leva em consideração a taxa de remoção de material, que depende das propriedades do material e dos parâmetros de corte. Além disso, o tempo de corte leva em conta o tempo necessário para a aproximação e retração da ferramenta em relação à peça de trabalho. O sistema proposto foi desenvolvido em *Visual Basic* e forneceu estimativas precisas de tempos de máquina para operações de desbaste e contorno na fresagem de face, com superfícies interpoladas lineares ou circulares. O sistema foi validado por meio de uma aplicação industrial para usinagem de matrizes. No entanto, deve-se observar que o sistema operava de maneira semelhante a uma simulação, e toda a estratégia de fabricação precisava ser predefinida.

O sistema desenvolvido por [1] é baseado em programas NC para estimar o tempo de usinagem. Este sistema inclui um módulo de reconhecimento de características baseado em grafos que permite a geração rápida de programas NC com base em características identificadas e a subsequente extração de parâmetros geométricos de uma base de conhecimento. A metodologia para estimar o tempo de usinagem visa equilibrar precisão e eficiência, considerando diferentes fatores dependendo do tipo de processo. Para operações de desbaste e perfuração, que envolvem usinagem em baixa velocidade, a estimativa é baseada no comprimento do trajeto da ferramenta e na velocidade de controle, permitindo um coeficiente de correção direta. Por outro lado, as fases de acabamento, caracterizadas pela usinagem em alta velocidade, requerem uma abordagem mais abrangente. Isso inclui considerar as características da máquina, informações geométricas e detalhes do trajeto da ferramenta para aumentar a precisão. Os tempos de aproximação e retração da ferramenta são determinados usando coeficientes derivados da análise de dados experimentais, enquanto o tempo de corte é calculado separadamente.

3 APRENDIZADO PROFUNDO

A jornada das redes neurais profundas (Deep Neural Networks - DNNs) começa com a inspiração nos processos biológicos do cérebro humano. A ideia de simular a forma como os neurônios interagem e aprendem levou ao desenvolvimento de redes neurais artificiais nos anos 1940 e 1950. No entanto, foi apenas nas últimas décadas que as DNNs ganharam destaque, impulsionadas pelo aumento da capacidade computacional e pela disponibilidade de grandes volumes de dados. Essas redes, caracterizadas por suas múltiplas camadas ocultas, permitem a aprendizagem de representações complexas de dados, tornando-se fundamentais em diversas aplicações, como reconhecimento de imagem, processamento de linguagem natural e tomada de decisão autônoma.

Seguindo essa tendência, as Redes Neurais Convolucionais (Convolutional Neural Networks - CNNs) - que são objeto central de estudo desse trabalho - emergiram como a espinha dorsal para o processamento de imagens e visão computacional. Inspiradas na organização do córtex visual de animais, as CNNs utilizam uma abordagem hierárquica para extrair características progressivamente mais complexas dos dados. Elas são formadas por camadas convolutivas que aplicam filtros à entrada, resultando na criação de mapas de características. Essas camadas convolutivas são alternadas com camadas de agrupamento, cujo objetivo é reduzir a dimensionalidade dos dados. Essa estrutura permite que as CNNs sejam eficientes na detecção de padrões espaciais, tornando-as ideais para tarefas como **reconhecimento de imagem**, **segmentação semântica** e **análise de vídeo**.

O objetivo deste capítulo é explorar o funcionamento das redes neurais convolucionais em diferentes tipos de dados, incluindo aqueles que não seguem uma estrutura de grade, ou seja, dados que não possuem a mesma dimensão em todas as amostras. A seção 3.1 tem como objetivo construir uma base sólida dos princípios fundamentais dos algoritmos de aprendizado de máquina, desde a criação do perceptron até algoritmos de otimização e estratégias de estabilização de modelos, como a normalização por exemplo. Na seção 3.7, apresentaremos os conceitos fundamentais das redes convolucionais, começando com

dados bidimensionais e expandindo para dados tridimensionais. Por fim, na seção 4, exploraremos os conceitos de convolução aplicados a dados gráficos.

3.1 Aprendizado de máquina

A expansão acelerada na disponibilidade de grandes volumes de dados (*big data*) e o avanço contínuo em capacidade computacional têm impulsionado o desenvolvimento e a aplicação do aprendizado de máquina. As aplicações destes algoritmos variam desde sistemas de recomendação personalizados, até contextos mais complexos como diagnósticos médicos avançados, desenvolvimento de veículos autônomos e sistemas automatizados de precificação.

Aprendizado de máquina é uma divisão da inteligência artificial que tem como objetivo a construção de algoritmos estatísticos que possam executar tarefas sem que tenham sido explicitamente programados com regras pré-definidas. Eles são caracterizados pela sua capacidade de identificar padrões em vastas quantidades de dados, tomar decisões informadas e, em alguns contextos, prever eventos futuros com notável precisão. Esses algoritmos se diversificam em categorias específicas, cada uma adaptada a diferentes tipos de problemas e conjuntos de dados e possuem três classes de distinção, nomeadamente: Aprendizado supervisionado, Aprendizado não supervisionado e Aprendizado por reforço.

3.1.1 Aprendizado supervisionado

Os algoritmos de aprendizado supervisionado são utilizados para tarefas de regressão e classificação e se caracterizam por terem à sua disponibilidade, durante o estágio de treinamento, dados rotulados. Esses dados chamados de "valor real"(*ground truth*) são importantes nesse tipo de aprendizado para que possa haver uma comparação e medição da discrepância com o valor estimado. O ajuste iterativo dos parâmetros do modelo para minimizar essa discrepância é o que caracteriza o aprendizado de fato. Enquanto a classificação envolve a estimativa de resultados em uma saída discreta, a regressão assume um caráter contínuo. Aplicações de visão computacional entram no espectro da classificação enquanto modelagem do clima é um exemplo clássico de uma tarefa de regressão.

3.1.2 Aprendizado não supervisionado

É dito que o aprendizado não é supervisionado quando não existem valore reais a disposição para comparação durante o estágio de treino de um algoritmo de *machine learning*. Por essa razão existe uma maior restrição das tarefas capazes de serem executadas com a utilização dessa técnica. O aprendizado não supervisionado é empregado para agrupamento (*clusterização*) ou redução de dimensionalidade. A clusterização compreende em agrupar dados semelhantes mesmo que não haja uma rega pré-estabelecida, como no caso de um modelo de recomendação de conteúdo. A redução de dimensionalidade, bastante utilizada em processamento de sinais, tem por objetivo a transformação, de um espaço de alta dimensão para um **espaço latente**, sem que haja perdas significativas das propriedades fundamentais que descrevem o dado.

3.1.3 Aprendizado por reforço

O aprendizado por reforço é caracterizado pela presença de um agente e um ambiente. A partir de uma interação aleatória com o ambiente (estado inicial), o agente é responsável por "bonificar"ou "punir"o algoritmo baseado no objetivo desejado, que se torna seu mecanismo de aprendizado. Um exemplo comum de aplicação prática é o desenvolvimento de modelos treinados para jogar xadrez por exemplo.

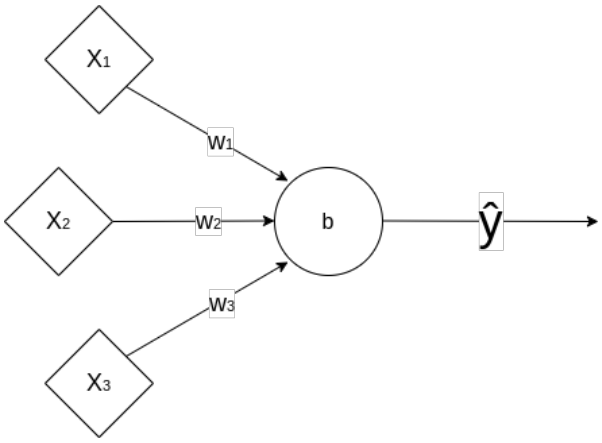
Existem ainda abordagens híbridas como o aprendizado semi-supervisionado quando apenas uma parte dos rótulos (valores reais) estão disponíveis durante o estágio de treinamento. Alguns algoritmos que se encontram debaixo do guarda-chuva do aprendizado de máquina são: a regressão linear, os SVMs (*suppot vector machines*) e as rede neurais.

3.2 Redes neurais

As redes neurais são algoritmos que buscam inspiração na estrutura do cérebro humano e por esse motivo buscam reproduzir a dinâmica de aprendizado através de unidades fundamentais de processamento e suas conexões. Neurônios artificiais são uma evolução de um conceito estatístico apresentado em 1943 no trabalho pioneiro de Pitts e McCulloch [54]. Mais tarde, baseado nas ideias de Donald Hebb [55], foi consolidado e sofisticado no trabalho de Frank Rosenblatt entitulado *The Perceptron: a probailistic model for information storage and organization in the brain* [56]. Esses neurônios rudimentares eram projetados para processar diversas entradas binárias na forma de *vetores de característica*

$x \in \mathbb{R}^{n_x}$ e produzir uma saída de mesma natureza $\hat{y} \in [0, 1]$. O modelo de Pitts e McCulloch possuía uma limitação prática pois para representar funções mais complexas, mais unidades de processamento precisariam ser interconectadas [57]. Para contornar essa limitação de escalabilidade e mitigar a necessidade do aumento da rede, o modelo final de Rosenblatt se mostrou mais genérico já que a "importância"de cada entrada (*input*) é expressa através de números reais chamados de pesos (*weights*) $w \in \mathbb{R}^{n_x}$.

Figura 14: Perceptron de Rosenblatt



Fonte: elaborado pelo autor

A saída do neurônio é determinada pela soma ponderada das entradas e pesos conforme a equação 3.1. Em um modelo simplificado de apenas um perceptron, significaria dizer que a variável com maior peso teria a maior influência sobre a saída do modelo.

$$\hat{y} = g\left(\sum_{i=1}^{m} x_i w_i + b\right) \tag{3.1}$$

Equação de saída do perceptron

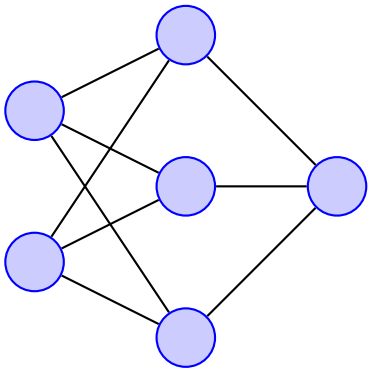
Além disso, também faz parte do modelo uma função de ativação g, que no caso do Perceptron é uma função degrau (equação 3.2). Se a saída for maior ou menor que um valor limite, denominado de viés (*bias*) b, isso determinará seu estado que pode ser ativo ou inativo. Um perceptron inativo tem em sua saída *booleana* 0, e quando ativo sua saída é 1. O viés pode ser entendido como uma medida da facilidade de ativação da unidade fundamental de processamento. É interessante constatar que dependendo dos valores atribuídos para os pesos e o valor limite (viés), uma variável com peso muito alto poderia tornar o valor das outras variáveis insignificantes para a tomada de decisão.

$$\hat{y} = \begin{cases} 0, & \text{se } w \cdot x + b \leq 0 \\ 1, & \text{se } w \cdot x + b > 0 \end{cases} \tag{3.2}$$

Função de ativação do perceptron

Uma das maiores vantagens desse modelo é sua flexibilidade. Da mesma forma que é possível expressar qualquer lógica binária através de uma configuração específica de portas NAND [58], apenas alterando os pesos e os vieses, diferentes modelos podem ser criados através da combinação de uma mesma estrutura fundamental.

Figura 15: Combinação de perceptrons em múltiplas camadas (MLP)

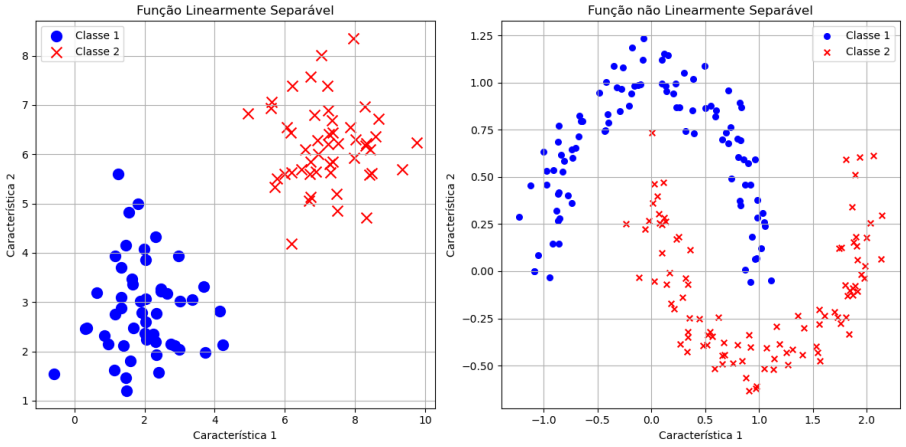


Fonte: elaborado pelo autor

No entanto, apesar das evoluções significativas, o perceptron de Rosenblatt herdou do seu precedente outra característica ainda mais restritiva à sua aplicabilidade a problemas complexos. Ainda que fosse possível implementar as três funções lógicas fundamentais (NOT, AND e OR), o que ficou conhecido como ”o problema XOR”evidenciava a incapacidade do modelo resolver problemas de classificação que não fossem **linearmente separáveis** [57]. A função degrau de Heaviside [59] não permitia que existisse qualquer configuração possível de entradas, pesos e viéses que pudessem funcionar como um XOR lógico [60] usando uma única unidade de processamento. Isso abriu caminho para o desenvolvimento de outras formas de ativação dos neurônios artificiais.

3.3 Funções de ativação

Figura 16: Função linearmente separável (esq.) Função não linearmente separável (dir.)



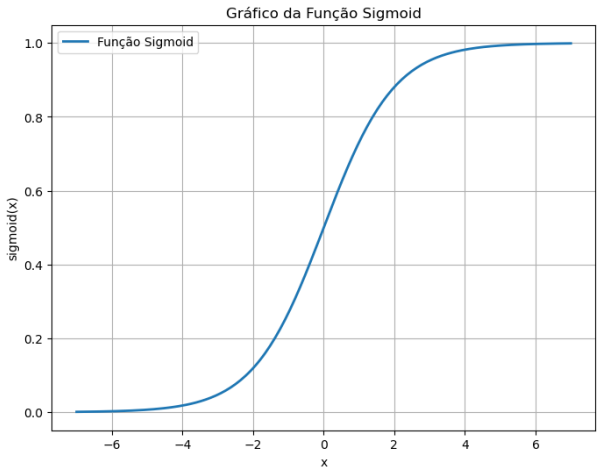
Fonte: elaborado pelo autor

Quando consideramos a natureza da função de ativação utilizada pelo perceptron, a incapacidade de resolver problemas não linearmente separáveis, como o ilustrado na figura 16, não é a única restrição imposta. Havia um inconveniente ainda maior. Suponha que a combinação de neurônios da figura 15 seja projetada para uma tarefa de classificação. Idealmente, deveria haver uma forma de ajustar gradualmente pesos e vieses com objetivo de classificar corretamente todas as categorias. O problema principal é que perceptrons não são capazes de sofrer ajustes graduais e uma pequena variação em sua estrutura pode fazer com que seu estado passe de predominante para totalmente inoperante. E embora isso possa ser desejável para a correta classificação de uma categoria específica pode estar mudando o comportamento da rede em uma forma muito complexa de controlar [58].

Procurando endereçar esses problemas, foi introduzido uma componente de não linearidade, conhecida como função de ativação, que substituiria a função degrau do Perceptron original conferindo mais flexibilidade e expressividade aos modelos [61], além de conferir uma estabilidade maior. Historicamente a função de ativação não linear mais utilizada é a função *sigmoid* (ou logística), descrita pela figura 17. Agora a saída do neurônio está **compreendida** entre $[0, 1]$, isso implica que seu estado pode estar em uma qualquer ponto dessa curva contínua e não mais radicalmente ligado ou desligado. A função sigmoid é definida como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

Figura 17: Função de ativação Sigmoid ou Logística



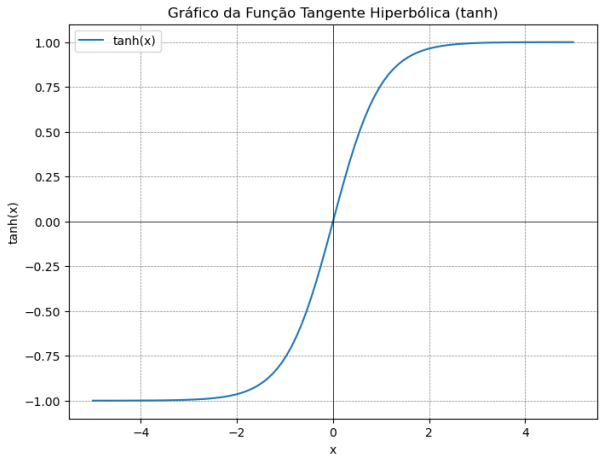
Fonte: elaborado pelo autor

É justamente essa maior **suavidade** da curva quando comparada à função degrau, que confere a habilidade de que pequenos ajustes na configuração do neurônio gerem pequenas variações em sua saída [58]. Como pode ser visto na 17 o fato dessa função limitar a saída do neurônio entre 0 e 1 de uma forma suave, faz com que ela seja conveniente para uma predição probabilística ou uma tarefa de classificação binária.

Também no espectro de funções sigmoidais, a função de tangente hiperbólica (*tanh*) paresentada na figura 18, apresenta muitas similaridades à função logística padrão. Ambas possuem a característica de serem monotonicamente crescente (a medida que move-se da esquerda para a direita no domínio da função, os valores da função também aumentam ou parmanecem iguais) ou então o fato de serem assíntotas (tendem um valor finito a medida que a variável dependente se aproxima de ∞). No entanto apresenta ainda outros benefícios como a sua propriedade simétrica ao redor da origem o que faz com que sua saída seja mais propensa a estar compreendida, na média, em um número próximo de zero [62]. A função *tanh* é definida conforme a equação 3.4:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.4}$$

Figura 18: Função de ativação Tangente Hiperbólica

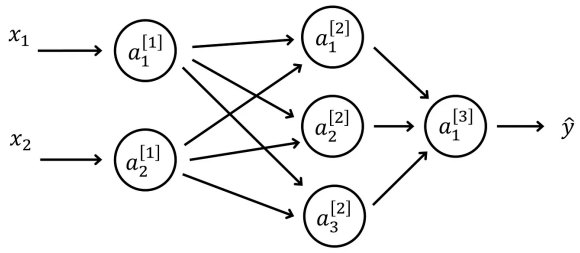


Fonte: elaborado pelo autor

3.4 O algoritmo de *Backpropagation*

Originalmente desenvolvido, em 1960, para a otimização de trajetórias de vôos [63], o algoritmo de *backpropagation* pode ser entendido como o mecanismo pelo qual a maioria das redes neurais modernas efetivamente **aprende**. O objetivo do aprendizado de uma rede neural é ajustar os pesos e os vieses de cada neurônio de tal forma que a saída da rede se aproxime o máximo possível dos valores reais (rótulos) para um determinado conjunto de dados de treinamento. Como mostrado na figura 19, os modelos de aprendizado profundo estão estruturados em camadas (*layers*) e possuem três definições distintas. A primeira camada possui a mesma dimensão dos dados de entrada e é denominada de "camada de entrada". De forma análoga a última camada, conhecida como "camada de saída", geralmente possui a mesma dimensão do resultado esperado (o número de classes para uma tarefa de classificação ou unidimensional para uma tarefa de regressão). As camadas que se encontram entre as de entrada e saída são denominadas "camadas ocultas"e cumprem um papel importante nesse tipo de algorítimo. À medida que mais camadas ocultas são adicionadas à rede, aumentamos sua profundidade, resultando em uma representação dos dados mais complexa e abstrata. Isso permite que a rede capture características e padrões mais intrincados e, **potencialmente**, melhore sua capacidade de realizar tarefas de aprendizado mais sofisticadas. Dado um neurônio j na camada l uma notação comum para sua referenciação é a_j^l.

Figura 19: Representação de neurônios em suas respectivas camadas



Fonte: [64]

Isso faz com que seja possível definir a ativação de um neurônio a partir das saídas da camada anterior, todos os pesos envolvidos e o viés geral da seguinte forma:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) \tag{3.5}$$

ou em sua forma vetorial:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{3.6}$$

Durante a fase de propagação direta (*feedforward*), a rede neural faz previsões com base nos pesos e vieses atuais e compara essas previsões com os rótulos reais do conjunto de treinamento. A discrepância entre as previsões da rede e os rótulos reais é medida por uma função do custo C. Existem muitas funções de custo utilizadas no domínio do aprendizado de máquina e geralmente elas são específicas para o tipo de tarefa e os dados disponíveis. As funções MAE (*Mean Absolute Error*) 3.7 e RMSE (*Root Mean Square Error* 3.8 são regularmente empregadas para avaliação de modelos, principalmente para tarefas de regressão, podendo ser mais ou menos adequadas dependendo do problema em questão [65].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.8}$$

Em última instância é necessário mensurar qual impacto que cada peso e cada viés possui no erro da rede. Em outras palavras, isso significa calcular a derivada parcial [66]

do custo em razão dos pesos (equação 3.9a) e a do custo em razão dos vieses (equação 3.9b) ao longo das camadas rede.

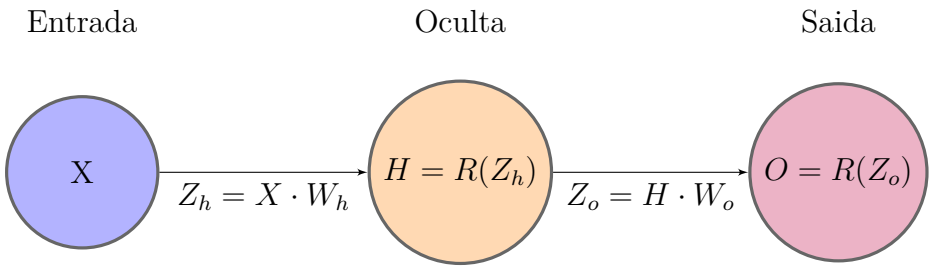
$$\frac{\partial C}{\partial w_{ij}^{l}} \quad (3.9a) \qquad \frac{\partial C}{\partial b_{j}^{l}} \quad (3.9b)$$

A regra da cadeia [67] atesta que para funções compostas, a derivada de uma função é o produto entre a derivada da função em razão da sua componente externa e a derivada da função da sua componente interna. Da forma que se: $y = f(g(x))$ então, segundo a regra da cadeia:

$$\frac{dy}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} \tag{3.10}$$

No exemplo da equação 3.10 foi aplicada a regra da cadeia para uma função contendo uma única ordem de composição mas a regra também é válida para funções de ordens maiores. Sendo assim, esse é o mecanismo utilizado para propagar as derivadas parciais do erro em relação a cada peso e a cada viés. Na rede neural 20, de apenas uma camada oculta e um neurônio em cada camada, teríamos o cálculo da derivado custo em razão do peso W_h conforme a equação 3.11.

Figura 20: Propagação das derivadas parciais da camada de saída em direção a camada de entrada



Fonte: elaborado pelo autor

$$C'(W_h) = C'(y^{)} \cdot O'(Z_o) \cdot Z_o'(H) \cdot H'(Z_h) \cdot Z_h'(W_h) \tag{3.11}$$

O conjunto de todas as derivadas parciais é agrupado em um vetor chamado de gradiente do erro [68]. Este vetor contém informações sobre a inclinação (gradiente) da função de perda em relação a cada parâmetro (peso e viés) da rede neural. O gradiente do erro é utilizado para isolar cada variável para determinar o quanto ela impacta na saída como

um todo. Isso é feito percorrendo iterativamente cada camada e calculando a derivada parcial da função de custo em razão de um parâmetro de cada vez, de forma que todas as outras variáveis assumam um valor constante. O gradiente possui duas propriedades especialmente úteis:

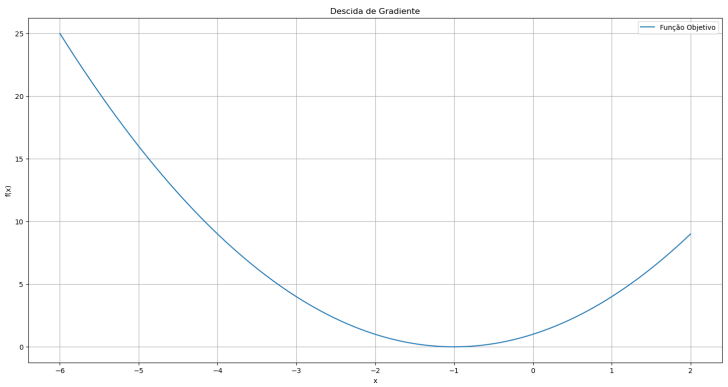
1. Sempre indica a direção do maior **aumento** de uma função [69].
2. Sempre assume valor zero quando em um ponto de máximo ou mínimo da função.

Em outras palavras, para minimizar a função de custo, é necessário **negar** o gradiente e percorrer a função no sentido oposto. Finalmente, o gradiente do erro é usado em um algoritmo de otimização, para ajustar os pesos e os vieses da rede neural. O algoritmo de otimização calcula a direção e a magnitude do ajuste necessário para reduzir o erro para todos os parâmetros da rede.

3.5 Algoritmos de otimização

Uma vez calculados e armazenados os gradientes para todos os pesos e vieses da rede, é necessário encontrar um conjunto desses parâmetros que façam o custo C o menor possível. Em uma função de apenas uma variável dependente como mostrado na figura 22 uma solução analítica poderia ser razoável, porém em uma rede neural com muitas camadas e muitos neurônios, isso se tornaria uma tarefa inviável [58]. Otimizadores, no contexto de aprendizado de máquina, são técnicas computacionais que são essencialmente responsáveis por ajustar os pesos e vieses da rede neural de forma a minimizar a função de perda.

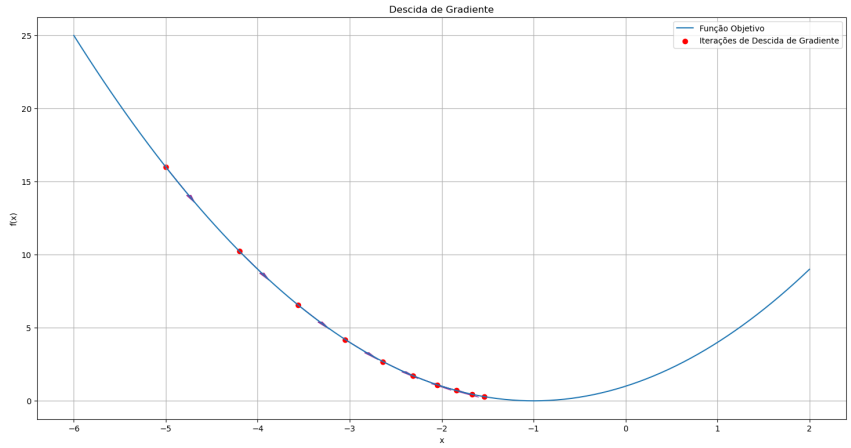
Figura 21: Otimizador por descida de gradiente



Fonte: elaborado pelo autor

Em sua forma fundamental, o algoritmo de *Descida de Gradiente* usa o gradiente do erro em razão dos pesos e vieses para definir qual a **direção** (positiva ou negativa) e a **magnitude** que deve atualizar os parâmetros para que haja a variação mais inclinada da função. Como o gradiente aponta para o maior aumento é necessário negá-lo para **minimizar** a função de custo. Uma vez calculados, a rede neural faz novamente a etapa de propagação direta com os parâmetros atualizados e o processo continua, de forma iterativa, até alcançar o ponto mínimo. Essa classe de otimização é frequentemente comparada à tarefa de encontrar o ponto mais baixo em um vale, o que pode ser visualizado como uma bola descendo uma colina. A bola move-se para baixo na direção da inclinação mais íngreme, atualizando os pesos em direção ao mínimo global (ou local) da função de perda. Usar a magnitude do gradiente diretamente na taxa de atualização pode ser problemático, atualizações significativas podem causar instabilidade e problemas de convergência. Dessa forma, escolhe-se um hiper-parâmetro chamado de taxa de aprendizado, geralmente denotadas pela letra alfa *alpha*. A taxa de aprendizado é o mecanismo de compensação do tamanho do passo em direção ao mínimo da função. Enquanto taxas de aprendizado baixas podem acarretar em tempos muito longos de treinamento, uma taxa excessivamente alta pode fazer com que o algoritmo "ultrapasse"o ponto mínimo. Na figura 22 a curva representa a função de custo enquanto os pontos vermelhos e as setas representam os parâmetros atualizados e a taxa de aprendizado, respectivamente.

Figura 22: Otimizador por descida de gradiente



Fonte: elaborado pelo autor

Atualização de peso (w):

$$w_{\text{novo}} = w - \eta \frac{\partial C}{\partial w} \tag{3.12}$$

Atualização de viés (b):

$$b_{\text{novo}} = b - \eta \frac{\partial C}{\partial b} \tag{3.13}$$

As equações 3.12 e 3.13 representam a atualização de forma matemática. Onde η é a taxa de aprendizado, $\frac{\partial C}{\partial w}$ e $\frac{\partial C}{\partial b}$ são as derivadas parciais da Custo em razão do peso e viés respectivamente.

O algoritmo de Descida de Gradiente é amplamente utilizado para treinar modelos de aprendizado de máquina e passou por muitas adaptações e evoluções na história recente. Assumindo uma rede grande o suficiente, seria necessário milhões de operações para processar uma única entrada do *dataset* de treinamento.

Para combater o tempo excessivo de treinamento para conjuntos de dados amplos, outra técnica muito utilizada é a do Gradiente Descendente Estocástico (*SGD*) que diferentemente da implementação padrão do Gradiente Descendente, atualiza os parâmetros a cada amostra do conjunto de treino, conferindo uma maior velocidade na otimização dos parâmetros. No entanto, devido sua natureza estocástica, o percurso traçado por esse tipo de otimizador é mais ruidoso ao mesmo tempo que proporciona uma exploração mais abrangente do espaço de parâmetros que pode ser benéfico em alguns casos mas também torna o modelo mais instável do ponto de vista de convergência.

Pensando em endereçar essa oscilação durante a tarefa de otimização, o Gradiente Descendente Estocástico com *Momentum* introduz a noção de velocidade e também de fricção à analogia da bola descendo a colina. Utilizando um conceito matemático de 1964 que ficou conhecido como *momentum* de Polyak, devido ao seu criador [70], o racional por trás desse mecanismo é que ele incorpora **inércia** ao movimento. Agora a bola não apenas considera a inclinação atual da colina (gradiente atual) mas também a direção e velocidade de suas movimentações anteriores. Cada passo em direção ao mínimo da função aumenta ainda mais a velocidade de convergência do modelo. No entanto, é necessário um segundo hiper-parâmetro μ para controlar a "velocidade da bola"uma vez que os gradientes mudem de direção, evitando que o otimizador ultrapasse o ponto de mínimo da função de perda.

Uma questão que surge com os algoritmos otimizadores apresentados até o momento é sua incapacidade de adaptar diferentes taxas de aprendizado a diferentes magnitudes de gradiente. Ainda que a incorporação do *Momentum* traga alguns benefícios nesse aspecto, uma abordagem mais robusta é a utilização do algoritmo de Propagação da Raiz Quadrada Média (RMSProp). Essa abordagem funciona mantendo uma média móvel do quadrado dos gradientes passados **para cada parêmetro** e então divide a taxa de aprendizado por esse valor. O cálculo da taxa de atualização do gradiente e da atualização de um

parâmetro (*weight*) ocorrem conforme as equações 3.14 e 3.15 respectivamente

$$s_{dW} = \beta s_{dW} + (1 - \beta) \left(\frac{\partial J}{\partial W} \right)^2 \tag{3.14}$$

$$W = W - \alpha \frac{\frac{\partial J}{\partial W}}{\sqrt{s_{dW}^{\text{corrected}}} + \varepsilon} \tag{3.15}$$

O algoritmo apresentado em [71] combina a adaptatividade do RMSPropo com o *momentum* SGD. O *Adaptative Moment Estimation* (ADAM) ajusta a taxa de aprendizado para cada parâmetro da rede individualmente e incropora o primeiro momento (média móvel exponencial) e o segundo momento (invariancia descentralizada), ambos calculados de forma exponencialmente ponderada - dando mais peso aos gradientes mais recentes e menos peso aos gradientes mais antigos. Com essas caracterísiticas o algoritmo é capaz de navegar eficientemente pelo terreno de otimização. Sua popularidade se dá tanto pela sua eficiência computacional quanto pela sua facilidade de implementação [72]. No entanto, essa estratégia possui uma forte tendência a zero nos passos iniciais, principalmente em taxas de aprendizados baixas, por essa razão são empregados fatores de correção no cálculo dos momentos. Como pode ser visto nas equações:

Primeiro Momento (Média Exponencial Ponderada dos Gradientes):

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.16}$$

Segundo Momento (Variância Exponencial Ponderada dos Quadrados dos Gradientes):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.17}$$

Correção de Bias para Primeiro Momento Corrigido

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.18}$$

Correção de Bias para Segundo Momento Corrigido:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.19}$$

Atualização dos Parâmetros:

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{3.20}$$

Onde:

θ_t são os parâmetros na iteração t,

α é a taxa de aprendizado,

β_1 e β_2 são hiperparâmetros de decaimento (geralmente próximos a 1),

g_t é o gradiente na iteração t,

$\hat{m}_t$ é o primeiro momento corrigido,

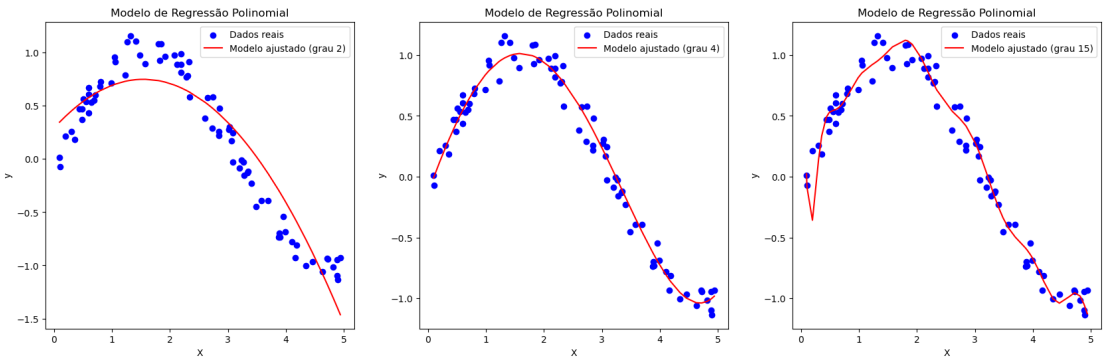
$\hat{v}_t$ é o segundo momento corrigido,

ϵ é uma pequena constante para evitar divisões por zero (geralmente 10^{-8}).

É razoável pensar que em um *dataset* grande o suficiente, existem redundâncias intrínsecas e que com um conjunto de amostras grande o suficientes é possível clacular uma aproximação do gradiente total. Essa técnica, denominada *mini-batch*, tem sido utilizada para aumentar a velocidade de treinamento de modelos muito complexos e pode ser empregada em qualquer algoritmo otimizador apresentado. Em lugar de alimentar o modelo com todos os dados de uma só vez, o conjunto de treinamento é dividido em pequenos subconjuntos chamados de mini lotes. Com valores típicos de 32, 64 ou 128 o modelo atualiza os parâmetros baseado nos gradientes do subconjunto ao invés de esperar chegar ao final da *epoch* (uma iteração completa de todos os dados do conjunto de treinamento).

3.6 Regularização

Figura 23: Possíveis ajustes de modelos estatísticos. *Underfitting* (esq.), *Fit* (centro) e *Overfitting* (dir.)



Fonte: elaborado pelo autor

Um problema comum encontrado por modelos de redes neurais com múltiplas camadas é a sua excessiva adaptação aos dados de entrada. Sua complexidade polinomial elevada torna relativamente fácil que qualquer função consiga ser explicada sem necessariamente que o modelo capture algum *insight* genuíno dos dados. Nas palavras do renomado matemático *John von Neumann*: "...com quatro parâmetros, posso ajustar um elefante, e com cinco posso fazer com que ele mexa sua tromba" [58]. Esse fenômeno de "memorização"dos dados de treinamento é chamado de sobre-ajuste *overfitting* e é o mesmo que dizer que o modelo possui uma variância demasiadamente alta. Em contrapartida, outra situação possível, é quando o modelo apresenta um viés exacerbado e sofre de subajuste (*underfitting*). O subajuste acontece quando um modelo é muito simples e não consegue capturar adequadamente os padrões dos dados de treinamento. Idealmente um modelo bem ajustado deve apresentar um bom equilíbrio entre baixo viés e baixa variância, já que frequentemente estão inversamente relacionados. A figura 23 ilustra os três ajustes possíveis para um dado modelo. As técnicas de regularização são métodos empregados para evitar o sobreajuste e o subajuste, melhorando a capacidade de generalização do modelo. A seguir, são apresentadas algumas das técnicas mais comuns de regularização utilizadas em redes neurais.

- **L1 Regularization:** Esta técnica adiciona uma penalização aos pesos da rede proporcional à soma absoluta dos valores dos pesos. Isso incentiva a rede a manter os pesos pequenos, reduzindo o risco de gradientes que desaparecem. A função de perda com L1 regularização é da forma $L = Loss + \lambda \sum |W|$, onde λ é o hiper-parâmetro de regularização.

- **L2 Regularization:** Também conhecida como regularização de peso quadrado, esta técnica adiciona uma penalização aos pesos da rede proporcional à soma dos quadrados dos valores dos pesos. Isso tem um efeito semelhante ao L1, incentivando pesos menores, mas de uma maneira um pouco diferente. A função de perda com L2 regularização é da forma $L = Loss + \lambda \sum (W^2)$, onde λ é o hiper-parâmetro de regularização.

Além disso, outra técnica eficaz para combater o *overfitting* e melhorar o treinamento é o método de **dropout**. O dropout envolve a aleatória "desativação"de um percentual de neurônios em cada camada durante o treinamento. Isso impede que os neurônios se tornem excessivamente dependentes uns dos outros, forçando a rede a aprender representações mais robustas. O *dropout* é especificado por uma taxa que controla a probabilidade de desativação de cada neurônio em cada etapa do treinamento [73].

3.7 Redes Neurais Convolucionais (CNNs)

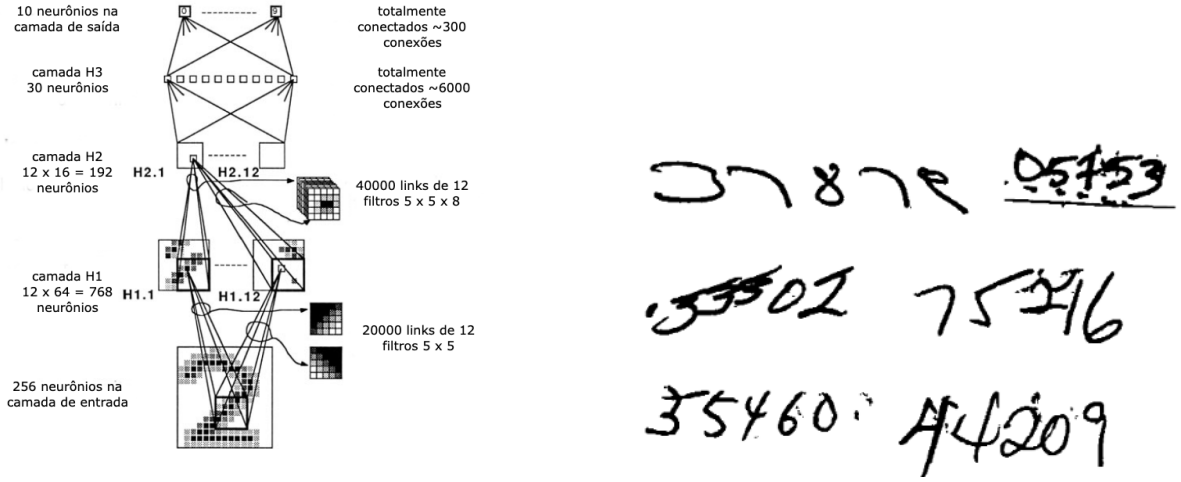
O principal problema da utilização de camadas totalmente conectadas, conforme apresentado até o momento, para o processamento de dados modelados em um espaço euclidiano, é que esse tipo de arquitetura não leva em consideração a estrutura espacial (de posicionamento) disponível. Pegando como exemplo a tarefa de processamento de imagens bidimencionais, que pode ser encarada como uma malha de pontos *pixels* sobre um eixo cartesiano, **não faria muito sentido processar pontos distantes e próximos exatamente da mesma maneira**.

Inicialmente proposto como um mecanismo de reconhecimento de padrões visuais, o trabalho intitulado de "neocognitron" [74] trouxe à luz uma nova forma de estruturação de camadas inspirado no modelo do sistema nervoso visual. A principal inovação presente, o reconhecimento de padrões invariantes à posição, sedimentou um dos três pilares do reconhecimento em estruturas euclidianas (*grids*). A estruturação do modelo em camadas sucessivas possibilitou o aprendizado de características de crescente complexidade, estabeleceu as bases para o processamento hierárquico e adaptativo de imagens.

No entanto foi apenas no início dos anos 1990, que o trabalho de Yann LeCun, intitulado "Backpropagation Applied to Handwritten Zip Code Recognition" [75] trouxe a composição dos elementos fundamentais das CNNs. Esse trabalho, que posteriormente ficou conhecido como LetNet-5, marcou um avanço significativo aplicando efetivamente a técnica de backpropagation em redes neurais para o reconhecimento de dígitos manuscritos de códigos postais de um correio do estado de Nova Iorque. Os dígitos foram localizados, segmentados e posteriormente processados em uma imagem de 16x16 pixels. O algoritmo usa as estratégias de pesos compartilhados, mapa de características e redução de dimensionalidade (*downsampling*) para alcançar uma sem precedente acurácia de 95% no conjunto de teste. A imagem 24 mostra a arquitetura adotada no trabalho bem como exemplos de dígitos manuscritos encontrados no conjunto de dados.

As redes neurais convolucionais são estruturadas em camadas de neurônios interconectados, com pesos e vieses, e que aprendem através do mecanismo de *backpropagation* da mesma forma como visto anteriormente. Porém, o sucesso das redes neurais convolucionais em processar dados em que a disposição espacial contém informação útil, ocorre principalmente em razão de três ideias básicas: campos receptivos locais (*kernels*), pesos compartilhados (*shared weights*) e agrupamento *pooling* [58].

Figura 24: Arquitetura da primeira rede neural convolucional (esq.) Exemplos de dígitos manuscritos (dir.)

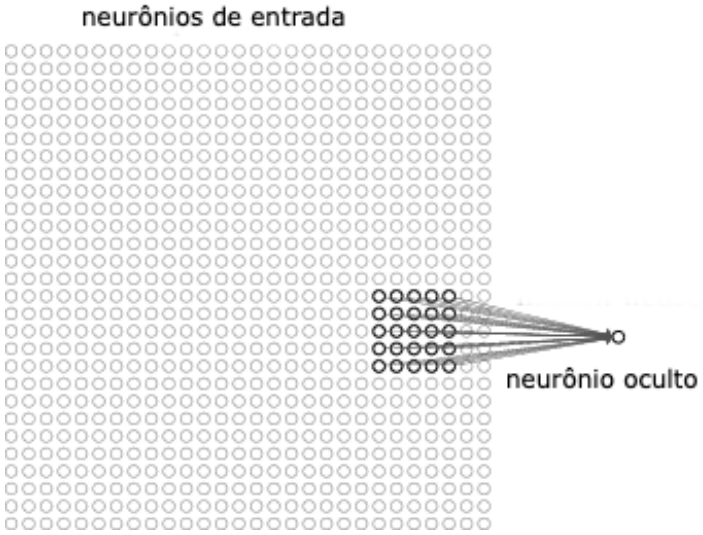


Fonte: extraído de [75]

3.7.1 Campos receptivos

Diferentemente das camadas totalmente conectadas, ao invés dos neurônios de uma camada mais internas estarem conectados a todos os neurônios da camada imediatamente mais externa, essa conexão é feita através de uma pequena ”janela”, geralmente de tamanho pré-definido *nxn*, chamada *campo receptivo* como demonstra a figura 25.

Figura 25: Campo receptivo de 5x5



Fonte: extraído de [58]

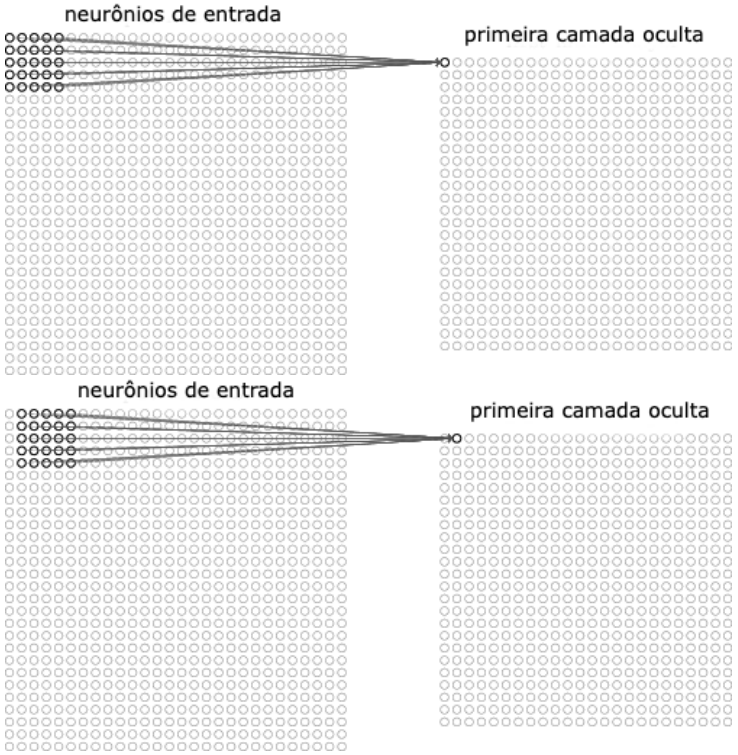
Cada uma das conexões entre a ”janela”de observação da camada anterior e o neurônio da camada atual também é regida por um viés (*b)* geral, aprendido pelo neurônio, e por

pesos que são aprendidos por cada uma das conexões após cada operação de **convolução**.

3.7.2 Operação de Convolução

Durante a operação de convolução, que acontece nas camadas convolucionais, a janela de observação é deslocada da esquerda para direita e de cima para baixo de forma a cobrir todo o espaço útil do dado de entrada, conforme mostra a figura 26. Dessa forma a cada neurônio da camada posterior é atribuída a 'responsabilidade' de observar uma região específica da camada anterior.

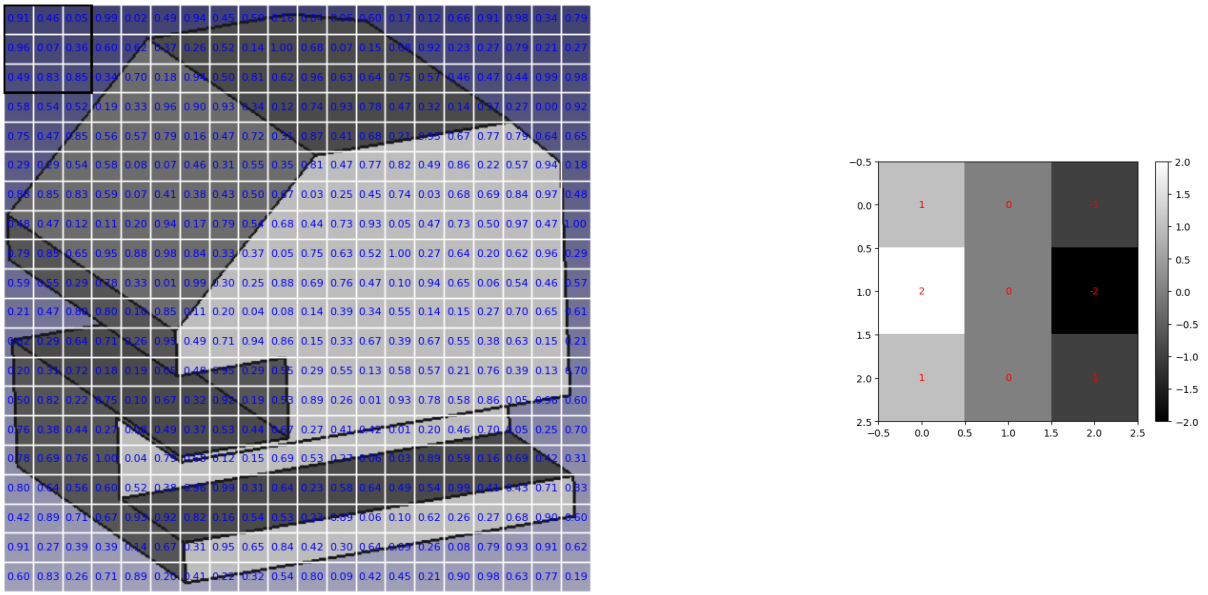
Figura 26: Deslocamento da janela de observação durante a operação de convolução



Fonte: extraído de [58]

Através dessa janela de observação, também chamada de filtro, ocorre a operação de convolução propriamente dita. Primeiramente, é calculada a multiplicação entre cada elemento do filtro e cada elemento de entrada no 'campo de visão' de um respectivo neurônio. Em seguida é computada a acumulação desses resultados, de forma a produzir um único valor para cada neurônio da camada mais interna. Como pode ser visto na figura 27.

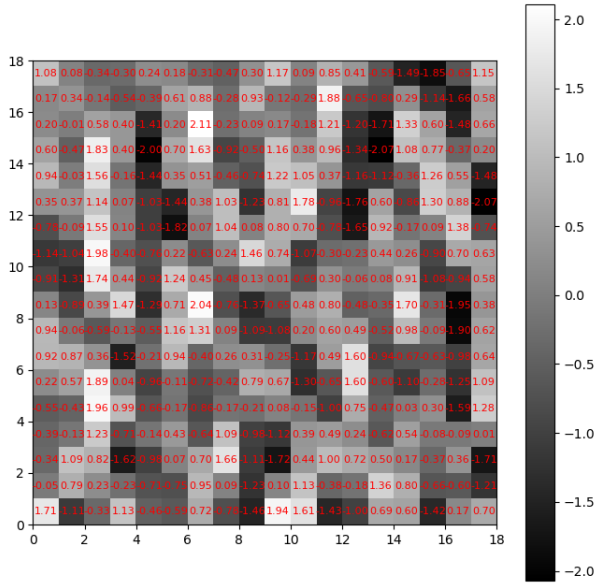
Figura 27: Operação de convolução



Fonte: elaborado pelo autor

O resultado final da operação por todos os neurônios da camada de convolução constitui o que é chamado de mapa de característica 28. Resultados convolutivos de alta magnitude indicam a presença da característica naquela região enquanto resultados de baixa magnitude indicam o contrário. Dessa forma, é como se os neurônios da camada posterior testassem a presença de uma determinada característica na camada anterior.

Figura 28: Mapa de característica formado por uma camada de convolução



Fonte: elaborado pelo autor

Em termos algébricos, a saída do neurônio na posição (j, k) de uma camada convolucional pode ser descrita conforme a equação 3.21

$$\sigma\left(b + \sum_{l=0}^{n}\sum_{m=0}^{n} w_{lm} q_{j+l,k+m}\right) \tag{3.21}$$

Onde:
σ é a função de ativação, por exemplo, a função sigmoide.
b é o viés compartilhado por todos os neurônios na camada convolucional.
$n \times n$ é o tamanho do *kernel*
w_{lm} representa os pesos do filtro ou kernel convolucional, que são aplicados às entradas locais.
$q_{j+l,k+m}$ é o valor da entrada na posição deslocada pelo filtro.

Nessa altura, é possível identificar uma propriedade intrínseca da convolução que é a redução de dimensionalidade. Ao agregar os valores resultantes da multiplicação dos pesos pela entrada, tem-se um mapa de característica invariavelmente menor do que o dado de entrada. Por essa razão o mapa de característica apresentado na figura 28 possui a dimensão de 18x18. Em condições clássicas de convolução o tamanho do mapa de característica é dado pela equação 3.22.

$$TamanhodoFeatureMap = (TamanhodaEntrada - TamanhodoFiltro) + 1 \tag{3.22}$$

Para que seja possível a captura de padrões locais e permitir que a rede aprenda características invariáveis à translação, é necessário que o mesmo filtro seja aplicado a toda entrada. Para que isso seja possível é utilizada a estratégia de **parâmetros compartilhados**

3.7.3 Parâmetros compartilhados

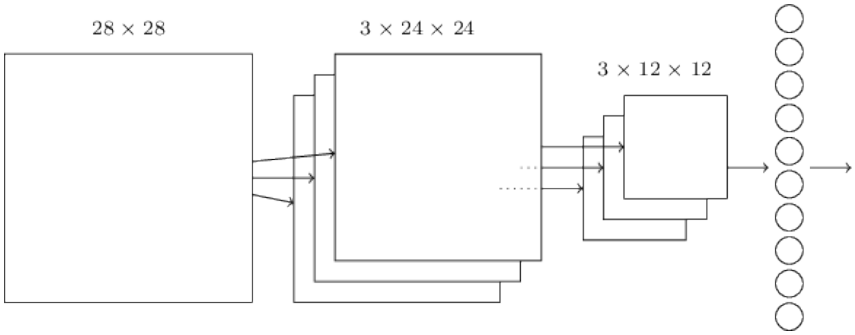
Diferentemente do que acontece nas camadas totalmente conectadas, os parâmetros aprendidos através do algoritmo de *backpropagation* são compartilhados por todos os neurônios de uma mesma camada. Dessa forma é como se todos os neurônios estivessem aplicando o mesmo filtro (*kernel*) porém em diferentes regiões da imagem. O viés apren-

dido, que também é compartilhado entre todos os neurônios, ajusta o limiar de ativação que indicará se esses padrões estão presentes ou não após a operação de convolução. O racional por trás dessa abordagem é de que a habilidade de detectar características específicas na imagem, digamos a borda de um dígito manuscrito, provavelmente convém que essa mesma característica seja detectada em outras regiões. Analogamente, isso faz com que essa arquitetura seja robusta em termos de invariância de deslocamento (*shift-invariance*) o que significa dizer que se o dígito está mais ou menos deslocado para direita não deveria influenciar na acurácia da sua identificação pelo modelo.

3.7.4 Agrupamento

Geralmente, redes neurais convolucionais utilizam mais de um filtro por camada (por exemplo quando o dado de entrada é uma imagem RGB) e executa a convolução de forma paralela para cada um deles. Nesses casos é necessária uma forma de simplificar o mapa de características sem que haja a perda das informações relevantes. Para isso aplica-se a técnica de agrupamento (*pooling*). Essas camadas são projetadas para criar uma espécie de sumário dos mapas de características extraídos da camada de convolução, reduzindo ainda mais a dimensionalidade da entrada, sem que seja aplicada nenhuma transformação externa ao dado de entrada como mostra o esboço da figura 29.

Figura 29: Operação de agrupamento

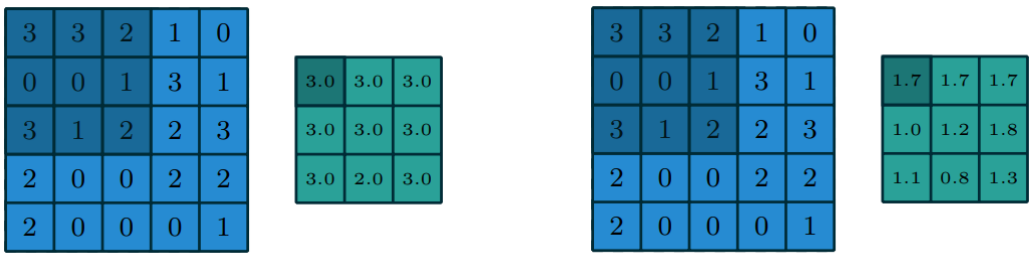


Fonte: extraído de [58]

A saída da camada de agrupamento é um mapa de característica extremamente simplificado que dá notícia se uma determinada característica está presente em uma região do mapa ou não. O racional por trás é que uma vez que uma característica é detectada, é menos importante saber sua localização precisa do que sua posição em relação a outras características. Isso além de reduzir consideravelmente a quantidade de parâmetros em camadas subsequentes, contribui para a robustez da invariância ao deslocamento da rede.

As duas técnicas mais utilizadas para o agrupamento são por máximo (*max-pooling*) ou pela média (*average-pooling*). Essas operações são aplicadas diretamente a cada elemento do mapa de cracterísticas conforme mostra a figura 30.

Figura 30: Agrupamento por máximo (esq.) e Agrupamento por média (d)



Fonte: extraído de [76]

3.7.5 O problema do desvanecimento de gradiente

Ao treinar uma rede neural usando algoritmos de otimização baseados em gradientes, o objetivo principal é ajustar os pesos da rede de forma a minimizar a função de perda. Para alcançar esse objetivo, calculamos os gradientes da função de perda em relação aos pesos e os usamos para atualizar os parâmetros da rede. No entanto, surge um problema quando esses gradientes se tornam excessivamente pequenos à medida que retrocedemos nas camadas da rede. Durante o processo de retropropagação do erro, os gradientes são multiplicados sucessivamente pelos gradientes das camadas anteriores. Se esses gradientes forem menores que 1, eles podem diminuir exponencialmente à medida que retrocedemos nas camadas, resultando em gradientes praticamente nulos nas camadas iniciais da rede. Quando os gradientes se aproximam de zero, os pesos nessas camadas não são atualizados de forma significativa, o que pode levar ao estagnamento ou interrupção completa do treinamento. Como resultado, a rede neural enfrenta dificuldades em aprender representações úteis, especialmente em arquiteturas profundas. Esse problema, conhecido como 'vanishing gradient', é particularmente relevante no treinamento de redes neurais profundas, especialmente em arquiteturas como redes neurais recorrentes (RNNs) e redes neurais convolucionais profundas (CNNs) [58].

O problema do gradiente desvanecente é bem conhecido e amplamente estudado no âmbito das CNNs (Redes Neurais Convolutivas). Na verdade, esse problema foi a principal limitação para redes convolucionais profundas antes da proposta da ResNet [77], que apresentou uma solução simples, porém eficaz. A introdução de conexões residuais

entre camadas consecutivas mitigou o problema do gradiente desvanecente, fornecendo caminhos adicionais para o gradiente. Isso possibilitou o treinamento confiável de redes residuais profundas com mais de cem camadas. A ideia foi ainda estendida pelo DenseNet [78], onde conexões adicionais são adicionadas entre camadas [79].

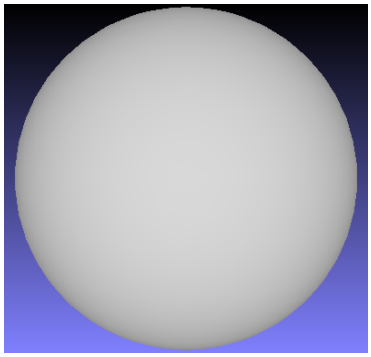
Uma abordagem eficaz para combater o desvanecimento de gradientes é a utilização da normalização em lotes (*Batch Normalization*). Originalmente projetada para acelerar o treinamento de redes neurais profundas, essa técnica também tem um efeito estabilizador no treinamento. Ela normaliza as ativações em cada camada, mantendo-as próximas de média zero e desvio padrão próximo a um. Isso ajuda a mitigar o problema de desvanecimento de gradientes, permitindo que os gradientes fluam de forma mais consistente durante o treinamento. Além disso, a *Batch Normalization* torna a rede menos sensível à seleção de hiper-parâmetros e promove um efeito leve de regularização [80].

3.8 Redes Neurais Convolucionais Volumétricas (3D-CNNs)

No decurso da década que iniciou em 2010, o avanço no campo do aprendizado profundo tem se revelado uma autêntica revolução no cenário científico, destacando-se em múltiplos campos de pesquisa que englobam desde a visão computacional até o processamento de linguagem natural. De maneira particular, as redes neurais convolucionais (CNNs) emergiram como protagonistas incontestáveis, apresentando-se de forma ubíqua nas soluções computacionais destinadas a enfrentar desafios inerentes ao domínio do reconhecimento visual. Tais desafios podem ser divididos em classificação de imagens, segmentação semântica, a detecção de objetos, principalmente. Importante ressaltar que as CNNs transcenderam as barreiras das informações visuais em duas dimensões, demonstrando uma notável capacidade de generalização para outros tipos de dados que se apresentam sob o formato de grades retangulares regulares. Tal destaque é evidenciado, por exemplo, na aplicação de convoluções unidimensionais para o processamento de sinais de áudio, bem como na utilização de convoluções tridimensionais, que abrangem tanto o espaço quanto o tempo, para analisar sinais de vídeo [18].

Ao contrário das imagens bidimensionais que se encaixam facilmente em uma grade regular (espaço euclidiano), os dados tridimensionais frequentemente têm formatos complexos que requerem mais poder de processamento. Para superar esse desafio, a primeira etapa é encontrar uma maneira eficaz de representar volumes tridimensionais de forma significativa e ao mesmo tempo computacionalmente eficiente. Em aplicações industriais

Figura 31: Representação de fronteira (*boundary representation*) de uma esfera



Fonte: elaborado pelo autor

e na interpretação automatizada de modelos CAD, existem quatro categorias principais de representação de modelos 3D: B-Rep, volumétrica, nuvem de pontos e malhas 3D. Esta seção apresenta essas representações e as principais arquiteturas que possibilitam seu processamento em modelos de aprendizado profundo.

3.8.1 B-rep

Os modelos de representação de fronteira (B-rep) são o padrão na descrição de objetos 3D em software de CAD comercial. Eles consistem em coleções de superfícies paramétricas recortadas, tornando a representação compacta e expressiva. No entanto, a complexidade das estruturas de dados e a falta de conjuntos de dados rotulados têm sido desafios para seu processamento. Além disso, frequentemente, o histórico de características paramétricas é perdido quando modelos são trocados entre diferentes aplicativos de CAD, e muitos sistemas de CAD comerciais utilizam algoritmos de segmentação para recuperar essas informações. A figura 31 ilustra essa representação.

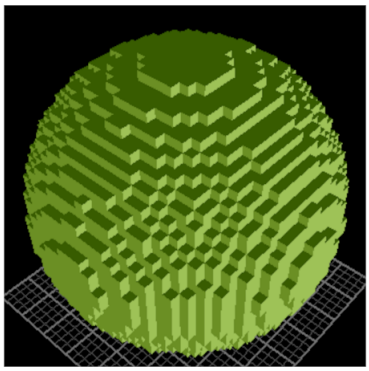
3.8.1.1 Modelo Base

O BRepNet [14] possui uma arquitetura de rede neural projetada para trabalhar diretamente com estruturas de dados de representação de fronteira (B-rep), eliminando a necessidade de aproximar modelos como malhas ou nuvens de pontos. Ele utiliza *kernels* convolucionais em relação às co-arestas, permitindo um processamento altamente especializado para B-rep. O modelo demonstra um desempenho superior na segmentação de modelos em comparação com métodos que usam outras abordagens de representação. No entanto, atualmente, há poucas aplicações que exploram essa arquitetura.

3.8.2 Voxels

Representações volumétricas fazem parte do espectro Euclidiano, já que essencialmente preservam uma estrutura em grade dos dados. Dados 3D podem ser representados de diversas maneiras. Uma delas é usando voxels, que são como pequenos blocos em um espaço tridimensional que descrevem como o objeto 3D está distribuído na cena. Além disso, informações sobre a perspectiva da forma 3D podem ser codificadas, classificando os voxels. No entanto, a representação baseada em voxels tem algumas limitações. Ela não é eficiente porque representa tanto as partes ocupadas quanto as não ocupadas da cena, o que requer muito espaço de armazenamento, tornando-a inadequada para dados de alta resolução. Uma representação mais eficiente é baseada em *octrees*, que são como voxels de tamanhos variados. Os *octrees* são uma estrutura de dados hierárquica que modela a ocupação do objeto 3D na cena tridimensional. Eles dividem a cena em cubos que estão dentro ou fora do objeto. Os *octrees* são capazes de representar detalhes finos de objetos 3D com menos cálculos do que os voxels, pois podem compartilhar o mesmo valor para grandes regiões do espaço. No entanto, tanto as representações baseadas em voxels quanto as baseadas em *octrees* não preservam a geometria dos objetos 3D em termos de suas propriedades intrínsecas e suavidade da superfície. Além disso, aumentar a resolução de dados volumétricos, significa aumentar cubicamente seu processamento [81]. A figura 32 mostra a representação volumétrica de uma efera com resolução de $32 \times 32 \times 32$.

Figura 32: Representação volumétrica de uma esfera



Fonte: elaborado pelo autor

3.8.2.1 Modelo base

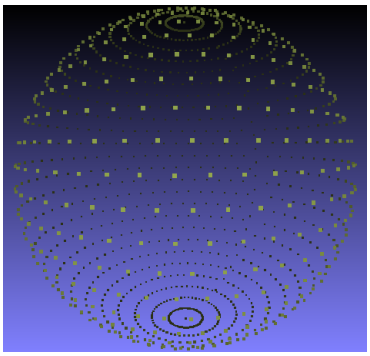
O estudo "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition" [11] foi pioneiro na aplicação de redes neurais convolucionais (CNNs) para a análise de dados 3D representados em formato de voxels. O experimento traz uma ar-

quitetura totalmente nova e compara seu desempenho com outras estratégias em dados LiDAR, RGBD e de modelos CAD.

3.8.3 Nuvem de pontos

Nuvens de pontos são conjuntos de pontos não **necessariamente** estruturados que representam a geometria de objetos 3D. Elas podem ser vistas como não Euclidianas, mas também podem ser tratadas como conjuntos de subconjuntos Euclidianos menores com uma parametrização global e coordenadas comuns. A maioria das técnicas de aprendizado visa capturar características globais para tarefas complexas, como reconhecimento e recuperação, classificando as nuvens de pontos como dados não Euclidianos. Apesar da facilidade de captura, como a utilização de sensores, o processamento de nuvens de pontos é desafiador devido à falta de estrutura e problemas relacionados à falta de informações de conectividade [81]. A figura 33 demonstra essa estrutura de dados.

Figura 33: Representação de nuvem de pontos de uma esfera



Fonte: elaborado pelo autor

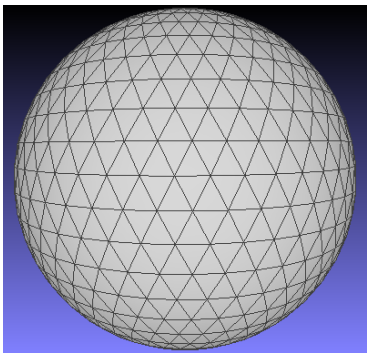
3.8.3.1 Modelo Base

O primeiro estudo significativo que mostra a aplicação de redes neurais convolucionais (CNNs) em estruturas de nuvens de pontos (point clouds) é frequentemente atribuído ao trabalho [12]. Este trabalho foi pioneiro no uso de redes neurais profundas para processar diretamente nuvens de pontos, introduzindo uma abordagem inovadora que permitiu a aprendizagem de características espaciais sem a necessidade de conversão em estruturas de grade ou imagens.

3.8.4 Malhas 3D

As malhas 3D são uma das representações mais populares para formas 3D. Uma estrutura de malha 3D consiste em um conjunto de polígonos chamados faces, que são descritos em termos de um conjunto de vértices que descrevem as coordenadas da malha no espaço 3D. Esses vértices estão associados a uma lista de conectividade que descreve como eles estão conectados entre si. A geometria local das malhas pode ser caracterizada como um subconjunto do espaço Euclidiano, seguindo dados estruturados em grade. No entanto, em uma perspectiva global, as malhas são consideradas dados não Euclidianos, onde as propriedades conhecidas do espaço Euclidiano, como invariância à translação, operações do espaço vetorial e sistema de parametrização global, não são bem definidas. Aprender com malhas 3D é desafiador por duas razões principais: métodos de Aprendizado Profundo (DL) não foram facilmente estendidos para tais representações irregulares. Além disso, esses dados geralmente sofrem com problemas como ruído, dados ausentes e problemas de resolução [81].

Figura 34: Representação de malha 3D uma esfera



Fonte: elaborado pelo autor

3.8.4.1 Modelos base

O estudo intitulado MeshCNN [13] foi pioneiro ao utilizar diretamente representações de malha triangular como dados de entrada em uma rede neural 3D CNN. Esta abordagem inovadora introduziu uma rede neural convolucional que opera diretamente em malhas triangulares, diferenciando-se de outras técnicas que exigem a transformação das malhas em formatos alternativos, como nuvens de pontos ou imagens. O MeshCNN processa diretamente a estrutura da malha, tratando as arestas das malhas triangulares como análogas aos pixels em uma imagem 2D. Isso permite a aplicação de operações convolucionais diretamente sobre a estrutura da malha.

Outra abordagem para operar sobre essa mesma representação pode ser encontrada no estudo [82]. O trabalho se destaca por sua metodologia de construir grafos dinâmicos a partir de estruturas de malha. O modelo apresentado foi intitulado de Dynamic Graph CNN (DGCNN) e aborda o desafio transformando a malha em um grafo e atualizando dinamicamente as conexões desse grafo durante o processo de aprendizagem.

4 REDES NEURAIS GRÁFICAS (GNNS)

As Redes Neurais Convolucionais (CNNs) trouxeram avanços notáveis em vários problemas de reconhecimento de padrões. Elas aprendem bancos de filtros de resposta ao impulso finito em uma hierarquia de camadas, com cada uma contribuindo com informações mais abstratas. Sua simplicidade as torna ideais para problemas estruturados como imagens, vídeos ou voz, onde os dados são homogêneos em termos de número, localização e força dos vizinhos. No entanto, muitos problemas, como os encontrados em áreas como farmacêutica, segurança interna e finanças, são desestruturados. Nesses casos, as CNNs padrão podem ter dificuldades devido à heterogeneidade dos dados, como variação no número de amostras e vizinhança [83].

Dados de usuários em redes sociais, dados genéticos em redes biológicas, registros em redes de telecomunicações e documentos de texto podem ser representados como grafos para modelar relacionamentos complexos. No entanto, a aplicação das redes neurais convolucionais (CNNs) a esses grafos é desafiadora, pois as CNNs são projetadas para estruturas regulares. Um dos principais desafios é definir filtros de grafos localizados eficientes para avaliação e aprendizado. Isso é importante para permitir o uso de CNNs em contextos não euclidianos [16].

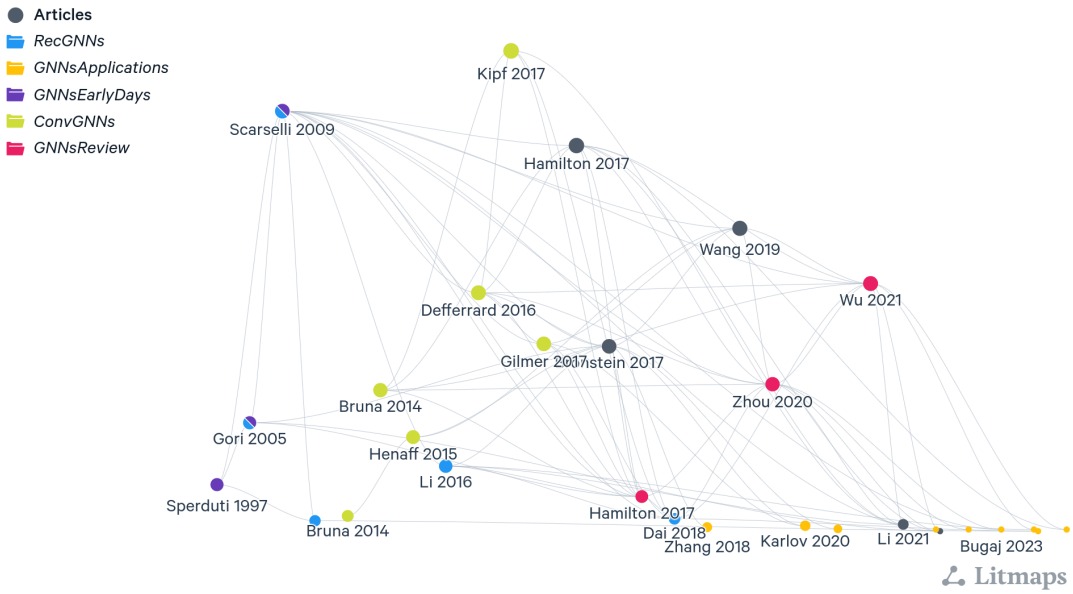
Esta seção tem o objetivo de introduzir os conceitos relacionados às Redes Neurais Convolucionais Gráficas (GCNs). Para isso, começaremos explorando brevemente os princípios da teoria dos grafos na seção 4.1. Em seguida, na seção 4.3.1, apresentaremos os fundamentos do aprendizado em grafos com as Redes Neurais Recorrentes (RecGNNs). Finalmente, na seção 4.3.2, abordaremos a convolução em grafos.

4.1 Teoria dos Grafos

Grafos são estruturas que consistem em nós e arestas e que podem ser utilizados para modelar sistemas complexos, onde as relações entre os elementos são tão importantes quanto os próprios elementos. Os nós, que representam as entidades fundamentais dentro

de um grafo, encapsulam os elementos individuais sob análise. As arestas, que são as conexões entre esses nós, oferecem uma perspectiva pela qual as relações são definidas. Em muitos casos, os grafos são utilizados para representar relações complexas e estruturas de dados interconectadas, como redes sociais, moléculas em química ou mapas de referências de artigos acadêmicos. A figura 35 apresenta um diagrama de grafos compreendendo artigos acadêmicos sobre Redes Neurais Gráficas

Figura 35: Exemplo de mapa de referência de artigos acadêmicos sobre GNNs. Cada nó indica um artigo publicado e cada aresta representa uma citação (elaborado pelo autor)

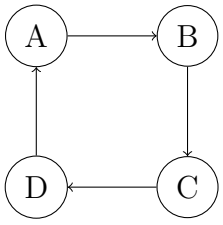


Fonte: elaborado pelo autor

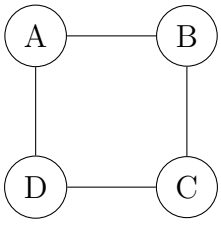
4.1.0.1 Tipos de grafos

Essa estrutura gráfica, definida por nós e arestas, possui configurações diversas, incluindo grafos direcionados, onde as arestas têm uma orientação unidirecional, grafos não direcionados, que representam interações bidirecionais, e grafos ponderados, que atribuem significado quantitativo às arestas. Os grafos direcionados, também conhecidos como dígrafos, são representados através de setas unidirecionais que explicitam a direção das relações entre os nós. Por outro lado, os grafos não direcionados preservam a simetria nas conexões, permitindo interações recíprocas sem indicar uma direção específica. A terceira categoria, grafos ponderados, adiciona uma camada adicional de complexidade, atribuindo valores numéricos às arestas para refletir a intensidade ou custo associado a cada relação [84]. A figura 36 apresenta exemplos de grafos direcionados, não direcionados e ponderados.

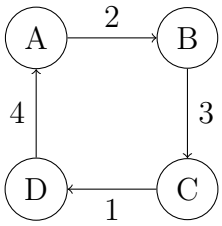
Figura 36: Exemplos de Grafos



(a) Grafo Direcionado



(b) Grafo Não Direcionado



(c) Grafo Ponderado

Fonte: elaborado pelo autor

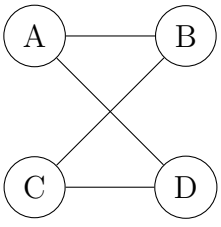
Um grafo direcionado $G = (V, E)$ consiste em um conjunto finito de vértices V e um conjunto de arestas E, onde cada aresta é uma dupla ordenada (u, v), indicando uma relação direcional do vértice u para o vértice v. Formalmente:

$$E \subseteq \{(u, v) \mid u, v \in V\} \tag{4.1}$$

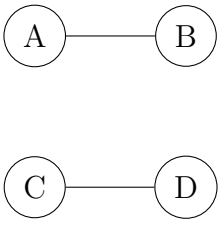
Um grafo não direcionado $G = (V, E)$ consiste em um conjunto finito de vértices V e um conjunto de arestas E, mas as arestas não têm direção específica.

No que diz respeito à conectividade, os grafos podem ser classificados como conectados, caracterizados pela presença de caminhos entre todos os pares de nós, ou desconectados, onde os componentes podem existir de forma relativamente isolada.

Figura 37: Exemplos de Grafos Conectado e Desconectado em Formato Quadrado



(a) Grafo Conectado



(b) Grafo Desconectado

Fonte: elaborado pelo autor

4.1.0.2 Grafos como representação de dados

Para realizar análises computacionais em grafos, é necessário transformar essas estruturas em representações matemáticas que possam ser manipuladas algebricamente. A estrutura de dados principal para o processamento de grafos em análises computacionais são as matrizes. Uma representação fundamental na teoria dos grafos é a matriz de adjacência, uma matriz quadrada que elucida a conectividade entre nós. Uma matriz quadrada **A** onde o elemento A_{ij} é um valor não-nulo se houver uma aresta do nó i para o nó j. Em grafos não direcionados, essa matriz é simétrica. A matriz de adjacência pode ser ponderada, caso as arestas tenham pesos.

4.1.0.3 Matriz de Adjacência para Grafos Não Direcionados

Dada uma matriz de adjacência A para um grafo não direcionado G com n vértices, onde $A_{i,j}$ é 1 se houver uma aresta entre i e j e 0 caso contrário, a matriz de adjacência é simétrica. Formalmente:

$$A = (a_{ij})_{n\times n} \quad sendo: a_{ij} \quad = \begin{cases} 1, & \text{se } (i,j) \in E \\ 0, & \text{caso contrário} \end{cases} \tag{4.2}$$

O grau de um nó, que indica quantas arestas estão conectadas a ele, e a distinção entre grau de entrada (número de arestas que entram em um nó) e grau de saída (número de arestas que saem de um nó) são importantes para caracterizar a topologia dos grafos. Além disso, conceitos como vizinhança (nós diretamente conectados a um dado nó) e alcance (caminhos entre nós) contribuem para entender as estruturas dos grafos.

4.1.0.4 Matriz de Grau para Grafos Não Direcionados

Dada uma matriz de grau D para um grafo não direcionado G com n vértices, onde $D_{i,i}$ é igual ao grau do vértice i (número de arestas incidentes ao vértice), a matriz de grau é representada por:

$$D = (d_{ij})_{n\times n} \quad sendo: d_{ij} = \begin{cases} \text{grau do vértice } i, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases} \tag{4.3}$$

4.2 Teoria Espectral dos Grafos

Os autovalores e autovetores de uma matriz são conceitos fundamentais da álgebra linear. Um autovetor de uma matriz é um vetor cuja multiplicação pela matriz resulta em um múltiplo escalar do próprio vetor, conhecido como autovalor correspondente. No contexto de grafos, os autovalores e autovetores da matriz Laplaciana podem ser interpretados como representações das frequências e modos de vibração do grafo, respectivamente, semelhantes aos conceitos de análise de Fourier em sinais. Na teoria dos grafos, a matriz Laplaciana desempenha um papel semelhante ao da transformada de Fourier em sinais, pois é composta pelos autovetores que representam as frequências características dos grafos [85]. A matriz Laplaciana é definida da seguinte maneira:

$$L = D - A \tag{4.4}$$

onde D é a matriz diagonal de graus (que contém o número de conexões de cada nó) e A é a matriz de adjacência (que indica se há uma ligação direta entre dois nós). A matriz Laplaciana é elementar porque captura informações sobre a topologia do grafo, sendo utilizada em muitas aplicações, incluindo a detecção de comunidades e a otimização de rede.

4.2.1 Matrizes Simétricas

Uma matriz é considerada simétrica se for igual à sua própria matriz transposta. Em outras palavras, uma matriz A é simétrica se $A = A^T$, onde A^T representa a matriz transposta de A. Uma matriz simétrica possui algumas propriedades importantes no contexto da teoria espectral de grafos. Por exemplo, os elementos diagonais da matriz simétrica são iguais, e os elementos fora da diagonal são reflexos uns dos outros em relação à diagonal principal.

$$A = \begin{bmatrix} 2 & -1 & 3 \\ -1 & 5 & 0 \\ 3 & 0 & 1 \end{bmatrix}$$

4.2.2 Matrizes Semi-definidas Positivas

Uma matriz A é considerada semidefinida positiva se, para qualquer vetor não nulo x, a desigualdade $x^T Ax \geq 0$ é satisfeita. Isso significa que o produto interno de x com a matriz A é sempre não negativo. Dada a matriz A, podemos verificar sua semi-definição positiva usando um vetor x arbitrário:

$$A = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 6 \end{bmatrix}$$

$$x^T Ax = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 12 \geq 0$$

4.3 O modelo de Rede Neural de Grafos (GNNs)

As Redes Neurais Gráficas (Graph Neural Networks, GNNs) são modelos de aprendizado profundo desenvolvidos para dados representados na forma de grafos, como redes sociais, estruturas moleculares, redes de transporte, entre outros. As redes neurais são estruturadas arquiteturalmente de formas diferentes dependendo da tarefa a ser resolvida. Recentemente, várias aplicações começaram a se beneficiar dessa tecnologia, como, por exemplo, a estimativa de estado em **sistemas de distribuição de energia** [86], a previsão de **tráfego de veículos** [87], a análise de similaridade entre **artistas musicais** [88], entre outros. As tarefas comumente abordadas incluem:

- **Classificação de Nós:** Determinar a categoria de um nó com base em suas características e na estrutura do grafo.
- **Classificação de Grafos:** Classificar todo o grafo em uma ou mais categorias.
- **Predição de Arestas:** Prever a existência ou a formação de uma aresta entre dois nós, útil em sistemas de recomendação e análise de redes sociais.
- **Segmentação de Grafos:** Identificar subgrafos ou comunidades dentro de um grafo maior.

- **Aprendizado de Representação de Grafos:** Aprender uma representação vetorial compacta do grafo ou de seus componentes.

A escolha da arquitetura mais apropriada geralmente depende do problema a ser resolvido, como será discutido nas próximas seções.

4.3.1 Redes Neurais Gráficas Recorrentes

A evolução das Redes Neurais Gráficas (GNNs) pode ser entendida através da progressão das formulações apresentadas em dois estudos fundamentais de Gori [89] e Scarselli [90]. No trabalho [89], a abordagem focava na representação de grafos através de redes neurais recorrentes, incorporando uma perspectiva inovadora para o tratamento de dados em domínios gráficos. Esta abordagem estabeleceu a base para o processamento de grafos direcionados e acíclicos, introduzindo conceitos como a representação vetorial dos nós e a propagação recursiva de informações através do grafo.

Com muitos conceitos trazidos do trabalho pioneiro [89], em 2009 os autores de [90] avançaram nessa formulação, apresentando um modelo mais abrangente que expandia a capacidade dos GNNs para processar uma variedade maior de tipos de grafos. Nesse trabalho, foi introduzido no mecanismo de atualização de estado, uma função de transição local que levasse em conta as arestas conectadas aos nós, permitindo que os GNNs capturassem mais eficazmente as complexidades estruturais. No modelo GNN proposto, cada nó do grafo é associado a um estado, que é uma representação vetorial do objeto ou conceito que o nó representa. Este estado é atualizado dinamicamente com base nas relações topológicas entre os nós. Um vetor de saída é definido para cada nó, dependendo do seu estado e rótulo, utilizando uma função de saída paramétrica. O modelo GNN resolve um sistema de equações para atualizar os estados dos nós e calcula as saídas com base nesses estados. O aprendizado é realizado adaptando os parâmetros da rede para aproximar uma função alvo utilizando-se dos dados de treinamento, isso é feito minimizando uma função de erro quadrático e utilizando a estratégia de gradiente descendente.

Baseado no teorema do ponto fixo de Banach [91], o objetivo do modelo é convergir para um estado estável único que sofre a menor alteração possível quando recorrentemente submetido à mesma entrada. A última peça fundamental do modelo é a exigência de que o processo de atualização das representações - a função global de transição - seja uma "contração" [92]. Isso é essencial para garantir que independente do estado inicial dos nós, após um número suficiente de iterações, os estados convergirão para o ponto fixo da função.

Quando comparado ao trabalho anterior [89], essa expansão torna o modelo mais flexível e abrangente, mas sem uma mudança radical de paradigma, ainda fazendo parte dos modelos de redes neurais gráficas recorrentes segundo a taxonomia apresentada em [93]. Do ponto de vista evolutivo, os vetores de características x_i e os vetores de saída o_i têm o mesmo significado em ambos os modelos, mas a função de transição f_w de [90] também inclui $l_{\mathrm{co}[i]}$, que são os rótulos das arestas conectadas ao nó i, proporcionando a possibilidade de uma representação mais rica da estrutura do grafo. A seguir é apresentada a conceituação matemática fundamental que governa o modelo descrito por [90].

$$X_n = f_w(l_n, x_{\mathrm{ne}[n]}, l_{\mathrm{ne}[n]}) \tag{4.5}$$

Atualização dos estados dos nós

onde X_n é o estado do nó n, f_w é a função de transição paramétrica, l_n é o rótulo do nó n, $x_{\mathrm{ne}[n]}$ são os estados dos nós vizinhos, e $l_{\mathrm{ne}[n]}$ são os rótulos dos nós vizinhos.

$$O_n = g_w(X_n, l_n) \tag{4.6}$$

Função de saída

onde O_n é o vetor de saída do nó n e g_w é a função de saída paramétrica.

$$E(w) = \sum_{i=1}^{p} (t_i - p_w(G_i, n_i))^2 \tag{4.7}$$

Função de erro quadrático

onde $E(w)$ é a função de erro, p_w é a função paramétrica do GNN, (G_i, n_i, t_i) são os exemplos de treinamento, com G_i sendo um grafo, n_i um nó do grafo, e t_i a saída desejada.

Com o intuito de contornar as limitações quanto à eficiência de treinamento e também quanto à capacidade de generalização (tanto para grafos de tamanhos distintos quanto para estruturas variáveis), alguns trabalhos surgiram subsequentemente. O modelo Graph Echo State Network (GraphESN) [94] incorpora uma separação entre a função de transição de estado e a função de predição. Essa otimização elimina a necessidade do cálculo de toda a rede o cada atualização de estado, resultando em um modelo mais eficiente mas também mais **flexível** (com uma capacidade maior de generalização). Ao fixar os estados

dos nós, o GraphESN consegue direcionar toda sua atenção para o treinamento apenas de uma camada de saída para realizar predições.

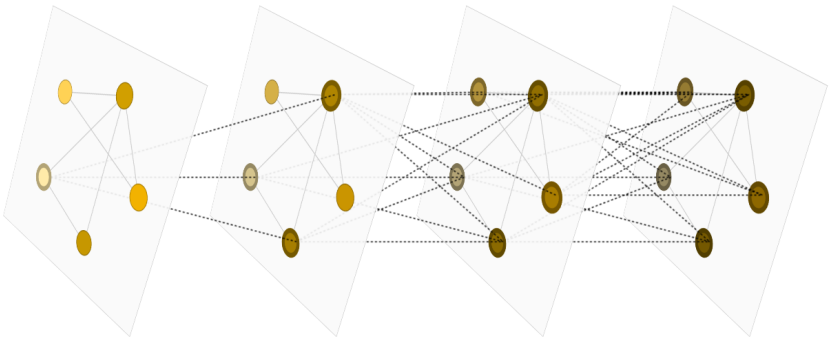
Em seguida houve o surgimento do Gated Graph Neural Network (GGNN) [95] que importou das redes recorrentes convencionais a ideia de aplicar unidades recorrentes com portões (Gated Recurrent Units - GRUs) aos dados gráficos, simplificando o processo de treinamento. Esta arquitetura admite que o modelo mantenha informações por longos períodos sem a necessidade de passar por todos os passos temporais, permitindo ser fixado em um número menor. Usando o aprendizado de parâmetros do modelo através do algoritmo de *backpropagation no tempo*, esse avanço abordou diretamente o desafio do armazenamento em memória.

Por fim, a Stochastic Steady-state Embedding (SSE) [96] é um algoritmo de aprendizado que atualiza os estados ocultos dos nós de forma recorrente e estocástica, oferecendo uma solução escalável para grandes grafos e mantendo a estabilidade do modelo. Ao introduzir um elemento de aleatoriedade no processo de treinamento, essa abordagem ataca de forma arquitetural os mínimos locais durante a otimização. Outro aspecto interessante dessa arquitetura é a capacidade de lidar com alterações dinâmicas na estrutura do grafo, uma vez que não está vinculado a uma única estrutura fixa, tornando-o adaptável a grafos que evoluem ao longo do tempo. Além disso, essa abordagem pode ser paralelizada facilmente conferindo eficiência computacional ao modelo.

Embora as Recurrent Graph Neural Networks (RecGNNs) tenham sido pioneiras na modelagem de dados estruturados em grafos e ainda que essas inovações representam marcos importantes na evolução das GNNs, elas enfrentam desafios significativos em termos de eficiência computacional, escalabilidade e generalização. A complexidade inerente à atualização recorrente de estados e a dificuldade de gerir grandes grafos, juntamente com o risco de dependência de ordem e problemas como desvanecimento do gradiente, revelam lacunas arquiteturais que inclusive estão presentes nas Redes Neurais Recorrentes (RNNS) tradicionais. Essas limitações abrem caminho para o surgimento das Convolutional Graph Neural Networks (ConvGNNs), que prometem abordagens mais eficientes e escaláveis, livres dos entraves das RecGNNs. A seguir, exploraremos como as ConvGNNs endereçam essas questões, introduzindo uma nova era na análise de dados de grafos, com um enfoque particular em sua capacidade de processar eficientemente informações em larga escala e sua adaptabilidade a uma variedade de estruturas de grafos.

4.3.2 Redes Neurais Gráficas Convolucionais (ConvGNNs)

Figura 38: Visualização de convolução em grafos



Fonte: extraído de [97]

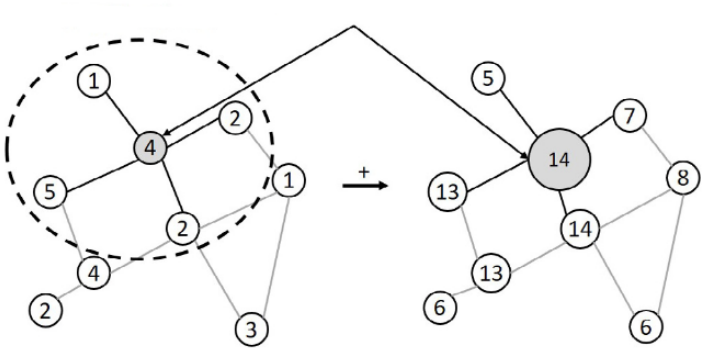
As Convolutional Graph Neural Networks (ConvGNNs) surgiram como uma resposta às limitações das Recurrent Graph Neural Networks (RecGNNs), ainda que estejam profundamente relacionadas. Ao incorporar conceitos das Redes Neurais Convolucionais (CNNs) convencionais, as ConvGNNs oferecem uma abordagem que supera os desafios de generalização e dependência de ordem inerentes às RecGNNs, eliminando a necessidade de passagens recorrentes que exigem alta capacidade computacional ao mesmo tempo que são capazes de capturar tanto os aspectos globais quanto locais das estruturas dos grafos. Em vez de iterar os estados dos nós com restrições contrativas, como é típico nas RecGNNs, as ConvGNNs lidam com as dependências cíclicas mútuas entre nós de forma arquitetural, utilizando um número fixo de camadas mas com pesos diferentes em cada uma delas. A estrutura das ConvGNNs permite um processamento paralelo e independente da ordem dos nós, trazendo uma robustez maior para o treinamento em grafos com topologias variadas e contornando o fenômeno do desvanecimento do gradiente por meio de sua arquitetura direta e hierárquica.

Segundo [98], a taxonomia das ConvGNNs pode ser categorizada com base em sua capacidade de capturar diferentes aspectos da estrutura do grafo. Inicialmente, temos as abordagens que operam no domínio espectral, e utilizam as propriedades dos autovalores e autovetores do Laplaciano do grafo, para realizar operações de convolução. Por outro lado, as ConvGNNs espaciais aproximam a convolução diretamente no domínio do grafo, focando na agregação das características dos vizinhos de um nó. Enquanto a abordagem espectral proporciona uma análise da estrutura global do grafo, a abordagem espacial é eficaz para tarefas que envolvem informações locais dos nós e suas vizinhanças.

4.3.2.1 Abordagem espacial

As abordagens espaciais definem convoluções diretamente no grafo com base na sua própria topologia, agregando informações da vizinhaça do nó. O principal desafio das abordagens espaciais é definir a operação com vizinhanças de tamanhos diferentes e manter a invariância local das CNNs. Além dos modelos pioneiros, como o modelo NN4G [99] e a convolução por difusão [100], existem verdadeiros *frameworks*, como a *Message Passing Neural Network* (MPNN) [101], que se propõem a ser suficientemente flexíveis para se comportarem como diferentes tipos de modelos. A figura 39 ilustra o conceito de convolução espacial em grafos.

Figura 39: Convolução espacial (localizada)



Fonte: extraído de [102]

A arquitetura PATCHY-SAN [103] tenta linearizar um grafo baseado no conceito de campo receptivo das CNNs. Uma operação de hashing inspirada nas propriedades das CNNs é definida e utilizada para aprender características em impressões digitais moleculares. As Redes Neurais Sequenciais de Grafos com Portões (Gated Graph Sequential Neural Networks) apresentaram um modelo de Rede Neural Recorrente do tipo Long Short-Term Memory (LSTM) aplicado à verificação de programas e tarefas básicas de raciocínio lógico, estendendo um modelo de rede neural que aprendeu uma representação de grafos. O DeepWalk aprende representações latentes de vértices de grafos usando caminhadas aleatórias para extrair padrões estruturais em redes sociais.

A própria arquitetura das bordagens espaciais fornecem a localização do filtro por meio do tamanho finito do *kernel*. Portanto, embora a convolução de gráficos no domínio espacial seja viável, ela enfrenta o desafio de combinar bairros locais. Consequentemente, não existe uma definição matemática única de tradução em gráficos do ponto de vista espacial [16].

4.3.2.2 Abordagem Espectral

Essa abordagem usa os conceitos da teoria espectral dos grafos que essencialmente realiza uma decomposição do sinal de entrada (grafo) em uma combinação de elementos mais simples. No contexto de processamento de imagens ou sinais esse conceito é muito utilizado através da Transformada Discreta de Fourier que decompõe as senoides em suas frequências fundamentais o que em essência converte o sinal do domínio do tempo para o domínio da frequência. Em espaços euclidianos, as convoluções são operadores lineares que são diagonalizados pela base de Fourier $e^{i\omega \cdot t}$, onde ω e t são elementos de $\mathbb{R}^d$, que é uma representação matemática de como uma função pode ser expressa como a soma de ondas senoidais. No entanto, não existe uma operação correspondente para os grafos devido à falta de uma estrutura de *grid*. A seguir será apresentado como a teoria espectral dos grafos foi empregada para resolver esse problema [85, 104].

O trabalho pioneiro conduzido em [15] busca trazer exatamente esse conceito para o reino dos grafos. A distribuição da frequência em um grafo pode ser entendida como quantas vezes diferentes valores ou características específicas aparecem nas conexões ou propriedades dos nós do grafo. Para um grafo não-direcionado e sem pesos, a distribuição da frequência tem uma relação direta com a distribuição de graus. Através de uma sólida fundamentação matemática, é construído um primeiro racional do que os autores intitulam como "construção espectral". Segundo o texto, a *construção espectral* baseia-se nas propriedades das convoluções no domínio de Fourier. Para estender o conceito de convoluções a grafos, os autores propõem o uso do Laplaciano do grafo, que é um operador que fornece uma análise harmônica para esse domínio. No contexto dos grafos, o Laplaciano desempenha um papel similar à transformada de Fourier em espaços euclidianos, ajudando a definir uma base para transformações e operações convolucionais. A ideia central é que, assim como as convoluções em um espaço Euclidiano são simplificadas no domínio de Fourier, as convoluções em grafos podem ser simplificadas usando o Laplaciano do grafo. O teorema da convolução demonstra que uma convolução no domínio espacial para processamento de sinal é equivalente a uma multiplicação no domínio da frequência [105]. Esse teorema também pode ser aplicado a estruturas gráficas. Para processamento de sinais, a transformação de um sinal para o domínio da frequência requer a utilização da Transformada Discreta de Fourier (DFT), onde uma multiplicação matricial é realizada entre o sinal e a DFT. No entanto, essa base pressupõe uma grade regular e, portanto, não pode ser aplicada a estruturas gráficas irregulares. Para realizar uma convolução espectral em um grafo, é necessário utilizar apenas alguns autovetores, correspondendo aos menores autovalores do laplaciano do grafo. Enquanto a Análise de Componentes Princi-

pais (PCA) é frequentemente empregada em visão computacional para calcular os maiores autovalores e seus autovetores correspondentes [106], no contexto do laplaciano do grafo, os autovetores associados aos menores autovalores são os que carregam informações sobre a estrutura e a geometria suave do grafo [15]. Portanto, é a base de autovetores U do laplaciano que é utilizada para a convolução espectral, que pode ser computada através de L:

$$L = U\Lambda U^T \tag{4.8}$$

Onde:
Λ é a matriz diagonal de autovalores
U a matriz de autovetores ordenada por autovalores

A partir daí, a convolução espectral de um sinal de grafo x é realizada pela transformação do sinal para o domínio espectral, multiplicando cada frequência espectral por um filtro h, e depois transformando o resultado de volta para o domínio espacial. Matematicamente, isso é expresso pela fórmula:

$$x * h = U^T \cdot (Ux \odot h) \tag{4.9}$$

Onde:
U é a matriz cujas colunas são os autovetores do laplaciano do grafo L;
U^T é a matriz transposta de U;
$\odot$ representa o produto elemento a elemento, também conhecido como produto de Hadamard.

O uso da matriz Laplaciana implica em duas limitações-chave. Primeiramente, usar a a matriz de adjacência para a convolução (ou aplicação de filtros) faz com que apenas as características dos nós vizinhos sejam somadas mas não as características do próprio nó selecionado para a operação. E segundo, que a matriz de adjacência não necessariamente está normalizada o que causa uma mudança de escala a cada operação de convolução. Endereçando essas limitações, [107] propôs uma matriz laplaciana normalizada e simétrica conforme a equação 4.10. A adição da matriz de adjacência com a matriz identidade adiciona arestas cíclicas no grafo (ou seja, coloca-se um $'1'$ na diagonal da matriz para cada nó, indicando que há um laço reflexivo ali) o que garante a disponibilidade de informações sobre o nó central da operação de convolução durante a operação. De forma complementar, normalizar a matriz de adjacência **garantindo que a soma das linhas é sempre igual a um**, resolve o problema de escala.

$$L_{\text{norm}} = I - D^{-1/2} W D^{-1/2} \tag{4.10}$$

Enquanto isso já significava um grande avanço da aplicabilidade das ConvGNNs, a abordagem espectral ainda enfrentava três limitações significativas devido à sua fundação na decomposição espectral da matriz Laplaciana.

1. Se houver uma pequena alteração na estrutura do grafo, como a adição ou remoção de nós ou arestas, os autovetores da matriz Laplaciana também mudarão. Como resultado, os filtros aprendidos se tornariam obsoletos, e o modelo precisaria ser treinado novamente para a nova estrutura do grafo. Isso limita a capacidade do modelo de lidar com grafos dinâmicos.

2. Os filtros aprendidos são específicos para a estrutura do grafo em que o modelo foi treinado. Eles não podem ser aplicados a grafos com uma estrutura diferente. Isso torna o modelo menos versátil, pois não pode ser facilmente transferido para diferentes domínios.

3. A decomposição espectral da matriz Laplaciana de um grafo é computacionalmente cara, com uma complexidade de tempo de $O(n^3)$, onde n é o número de nós no grafo (cresce cubicamente). Isso torna o modelo ineficiente para grafos muito grandes, pois a computação dos autovetores e autovalores pode ser demorada e requer recursos computacionais significativos.

Para contornar essas limitações, o estudo [16] apresenta uma técnica que envolve a aproximação de filtros espectrais usando polinômios de Chebyshev [108]. Esta abordagem começa com o cálculo do Laplaciano normalizado do grafo, denotado por L, que é essencial na captura da estrutura do grafo. A matriz L é então reescalada para que seus autovalores caiam no intervalo $[-1, 1]$. Tal reescala é realizada através da fórmula $\tilde{L} = \frac{2L}{\lambda_{max}} - I$, com λ_{max} representando o maior autovalor de L e I a matriz identidade. Com a matriz Laplaciana reescalada, inicia-se o cálculo dos polinômios de Chebyshev, que são definidos pela relação de recorrência conforme a equação 4.11.

$$\begin{gathered} T_0(x) = 1, \quad T_1(x) = x, \quad \textit{seguido} \quad \textit{por} : \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \textit{para} \quad n \geq 1 \end{gathered} \tag{4.11}$$

No contexto do nosso problema, os polinômios são aplicados a $\tilde{L}$, resultando em $T_0(\tilde{L}) = I$ e $T_1(\tilde{L}) = \tilde{L}$, e subsequentemente $T_{n+1}(\tilde{L}) = 2\tilde{L}T_n(\tilde{L}) - T_{n-1}(\tilde{L})$ para os

polinômios de ordens superiores. A convolução de um sinal de grafo x com um filtro aproximado por polinômios de Chebyshev g_θ é dada pela equação 4.12

$$x * g_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{L}) \cdot x \tag{4.12}$$

Onde:

x é o sinal no grafo,

$*$ denota a operação de convolução,

g_θ é o filtro parametrizado pelos coeficientes θ,

$\sum$ representa a soma sobre K termos,

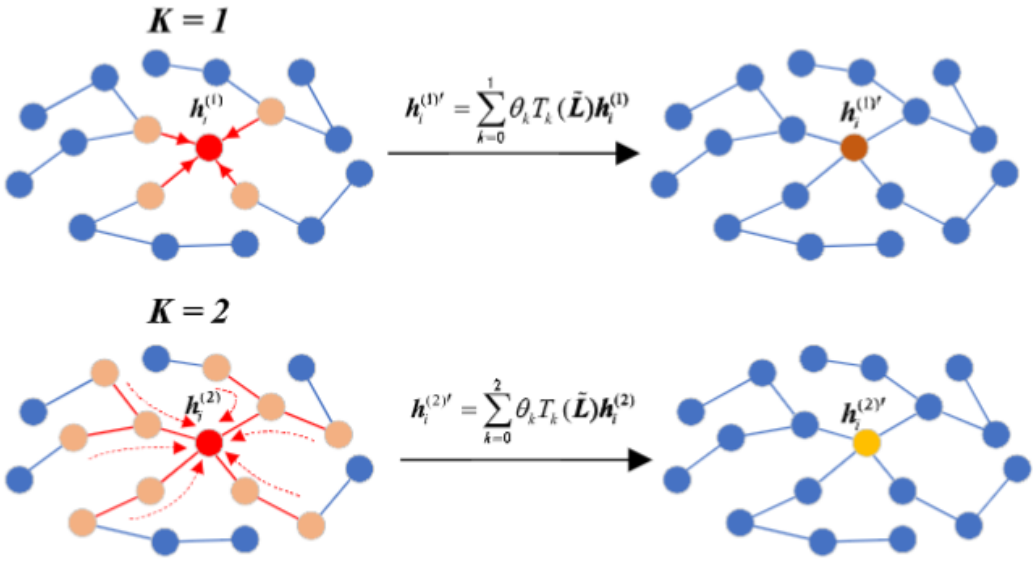
θ_i são os coeficientes do filtro,

$T_i(\tilde{L})$ são os polinômios de Chebyshev do Laplaciano escalados $\tilde{L}$,

K é a ordem da aproximação polinomial.

Um aspecto interessante dessa arquitetura é a existência do hiperparâmetro K. Esse parâmetro denota o grau polinomial da recursão para o cálculo dos filtros e pode ser interpretado como a distância em que a agregação de informação opera. Em outras palavras, quanto maior o valor de K, mais distante do nó central da convolução as informações estão sendo agregadas. A figura 40 ilustra esse conceito para $K = 1$ e $K = 2$.

Figura 40: Efeito do grau polinomial na operação de convolução



Fonte: extraído de [109]

5 APRENDIZADO PROFUNDO NA INDÚSTRIA DE MANUFATURA

O campo de aprendizado de máquina (*Machine Learning)* teve um grande impacto na indústria de manufatura, especialmente na era da Indústria 4.0. Essa revolução industrial incentiva o uso de sensores e dispositivos inteligentes para coletar dados de produção em tempo real. As técnicas de ML permitem processar esses dados para melhorar a eficiência da manufatura e oferecer *insights* preditivos. Isso é aplicável em várias áreas, como inspeção, manutenção, melhoria da qualidade e otimização de processos. No entanto, ainda existem desafios, como o gerenciamento de grandes volumes de dados, o uso de dados em tempo real e questões de segurança [110].

Especificamente na área de processamento geométrico, é notável o uso do aprendizado profundo para o reconhecimento automático de características de usinagem. Este capítulo tem como objetivo explorar as metodologias recentes nesta área, bem como uma de suas aplicações, que é a estimativa de informações pertinentes à fabricação.

5.1 Reconhecimento automático de características de usinagem

O reconhecimento automático de características usináveis (FAR) tem sido um campo bastante ativo de pesquisa na área da engenharia e refere-se ao processo de identificar e interpretar elementos geométricos em modelos de CAD que são relevantes para processos de manufatura. Em outras palavras, visa traduzir as entidades geométricas de baixo nível dos modelos CAD em um conjunto de 'características' apropriadas e com atributos que estejam relacionados às necessidades do planejamento e execução da fabricação [111]. Ao longo das últimas três décadas, foram desenvolvidos vários sistemas de reconhecimento de características, incluindo abordagens baseadas em regras lógicas, grafos, decomposição de volume, baseadas em dicas, redes neurais e híbridos. Apesar dos problemas que ainda restringem as aplicações práticas, como a falta de robustez e complexidade computacional,

a pesquisa em reconhecimento de características continua sendo relevante até os dias de hoje. Este campo emergiu nesse contexto, como uma resposta direta às necessidades de otimização de processos de manufatura, buscando a automatização e a eficiência na transformação de desenhos em instruções de usinagem, sendo assim, os esforços em pesquisa tem sido principalmente motivados no contexto da integração de sistemas CAD/CAM e na elaboração automática de Plenajamento de processos (CAPP) [7].

A integração do reconhecimento de características usináveis com sistemas CAD/CAM é um aspecto fundamental no avanço da manufatura digital. Esta integração permite que informações detalhadas de design sejam eficientemente convertidas em instruções de fabricação precisas. Ao identificar características usináveis diretamente dos modelos CAD, o processo de planejamento e programação para manufatura CAM torna-se mais rápido e menos suscetível a erros, potencializando a eficiência do fluxo de trabalho. A necessidade desta integração tem sido uma constante no desenvolvimento de plataformas CAD e CAM, sendo um facilitador direto para sistemas de tecnologias assistidas por computador (CAX). Essa sinergia entre o design assistido por computador e a manufatura assistida por computador habilita a capacidade de produzir peças complexas com maior precisão e em menos tempo. A adoção de sistemas integrados CAD/CAM, apoiada pelo reconhecimento avançado de características, representa um passo significativo em direção à manufatura inteligente, onde a tomada de decisões e o planejamento são otimizados para atender às demandas de uma produção mais ágil e personalizada [112].

Historicamente, o desenvolvimento de técnicas de reconhecimento de características acompanhou os avanços tecnológicos nos sistemas CAD/CAM, visando aprimorar a interpretação de geometrias complexas e a subsequente geração de planos de usinagem eficazes [113]. Este processo evoluiu de abordagens manuais para técnicas altamente automatizadas, incorporando avanços em inteligência artificial e aprendizado de máquina. As características usináveis são componentes fundamentais, que podem ser encarados como elementos constituintes, em peças de manufatura e que determinam os processos de usinagem necessários para sua produção. Estas características, como furos, ranhuras ou rebaixos, são geralmente identificadas a partir de modelos CAD, e a sua identificação correta, é ferramenta indispensável para uma correta elaboração da estratégia de manufatura. [114].

À medida que a indústria avança em direção à manufatura 4.0, o reconhecimento de características usináveis torna-se ainda mais relevante, integrando-se com sistemas de manufatura inteligentes e adaptativos [112]. O reconhecimento de características permite a automação de várias etapas no processo de manufatura, desde o planejamento inicial até

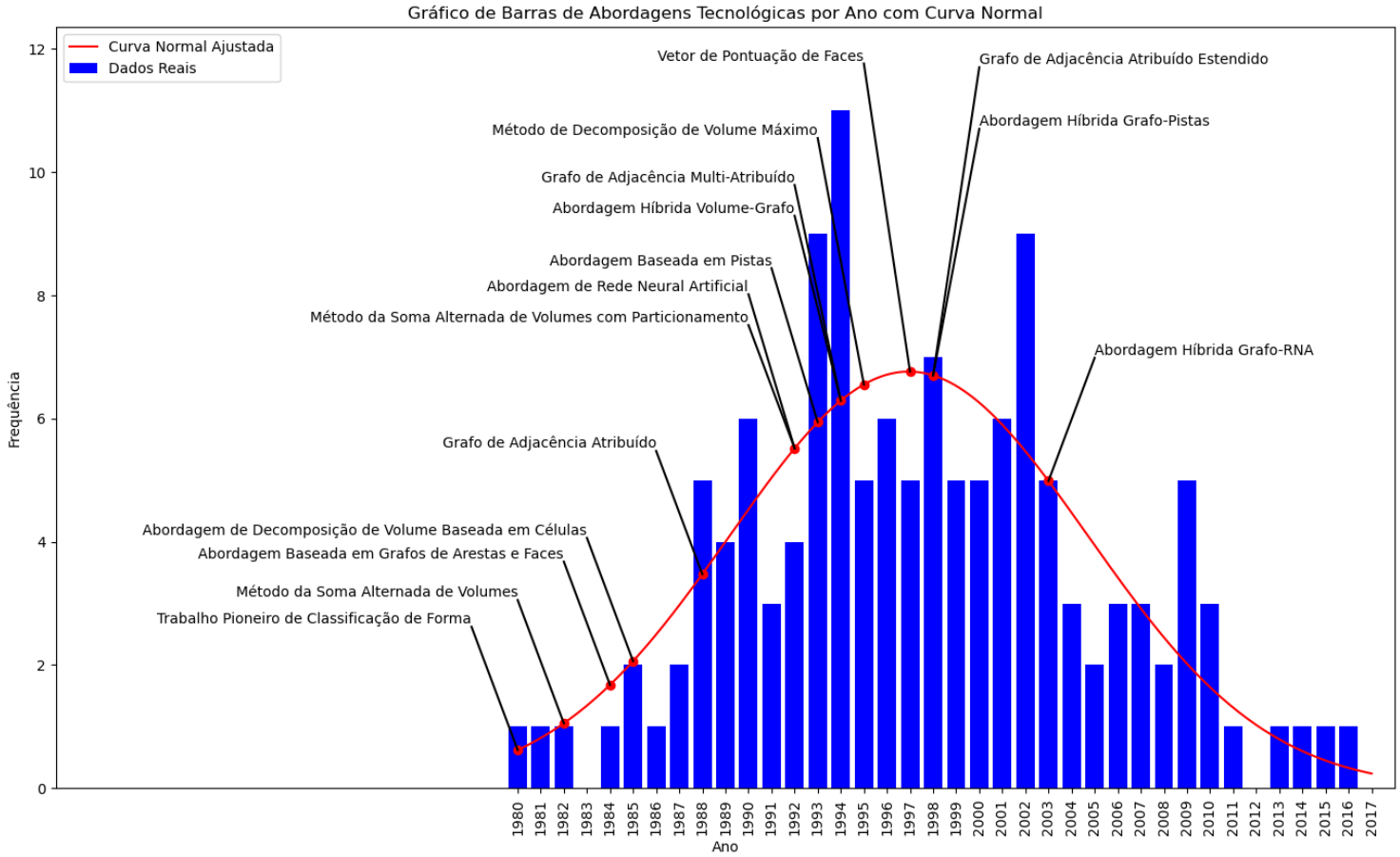
a usinagem final. Isso pode incluir a seleção automática de ferramentas, a determinação de sequências de operações de usinagem e a otimização dos parâmetros de corte. Como resultado, pode haver uma redução significativa no tempo de preparação e no tempo de ciclo de usinagem, levando a uma maior produtividade e redução de custos [30]. Portanto, a relevância deste campo estende-se além da mera conversão de desenhos em peças usinadas; ele abrange a otimização de processos, redução de custos, melhoria da qualidade do produto final.

Este trabalho proporciona uma visão abrangente dos métodos de reconhecimento automático de características na área da usinagem em duas seções. A seção 5.1.1 discute a evolução dos métodos convencionais, enquanto a seção 5.1.2 apresenta abordagens mais recentes e sofisticadas que fazem uso de algoritmos de aprendizado profundo.

5.1.1 Métodos convencionais

O reconhecimento de características de usinagem é um campo em constante evolução dentro da fabricação assistida por computador (CAM). As técnicas desenvolvidas ao longo das décadas demonstram um progresso contínuo, ajustando-se às necessidades emergentes da indústria de manufatura e incorporando avanços tecnológicos e computacionais. A Figura 41 fornece uma visão cronológica dos principais trabalhos relacionados a esse tema.

Figura 41: Número de trabalhos relacionados por ano e principais marcos da pesquisa de técnicas de reconhecimento de características de manufatura

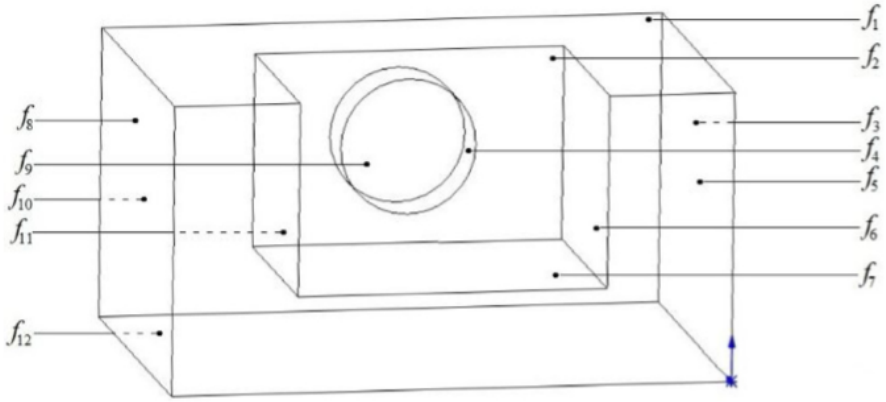


Fonte: adaptado de [7]

Inicialmente, o reconhecimento de características era baseado em abordagens heurísticas e algoritmos específicos para tipos de características predefinidas, onde as características geométricas das peças eram identificadas através de regras codificadas de forma rígida e específica [115,116]. Estes métodos mostraram-se limitados devido à sua inflexibilidade e a necessidade de intensa intervenção humana para definir e ajustar as regras para cada nova peça ou configuração. Esses sistemas evoluíram para incorporar 'sugestões', onde pistas incompletas são utilizadas para identificar potenciais características, oferecendo maior flexibilidade na interpretação de modelos complexos [28, 117]. Ainda que essa abordagem aumentasse a capacidade dos algoritmos baseados em regras, era ainda bastante inflexível pois as 'sugestões' eram definidas por classe de características.

O avanço para abordagens baseadas em grafos ofereceu uma nova perspectiva no reconhecimento de características. Nestes métodos, as representações geométricas são transformadas em grafos, onde os nós e as arestas representam componentes geométricos e suas relações. As características são reconhecidas por meio da identificação de subgrafos que correspondem a padrões específicos dentro do grafo maior que representa a peça de trabalho como um todo. Pesquisas notáveis nesta área incluem os trabalhos de [8], que introduziram o conceito de Grafos de Adjacência com Atributos (AAGs) para capturar relações geométricas complexas, facilitando a identificação de características e propondo com uma forma computacionalmente eficiente de representar modelos 3D. Posteriormente, várias metodologias exploraram essa estrutura para codificar modelos CAD 3D e suas características, abrangendo tanto faces quanto arestas [9, 10].

Figura 42: Modelo CAD com faces identificadas

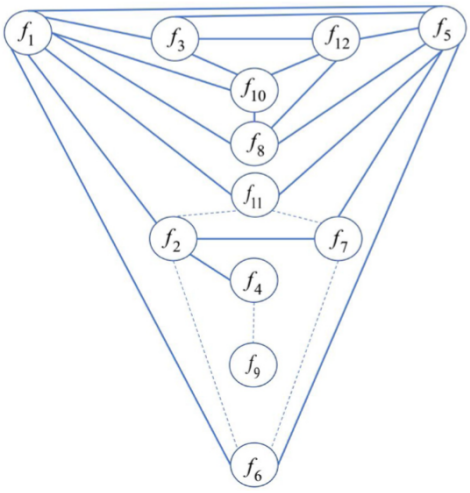


Fonte: extraído de [118]

A figura 42 representa um sólido com suas faces devidamente identificadas. A aborda-

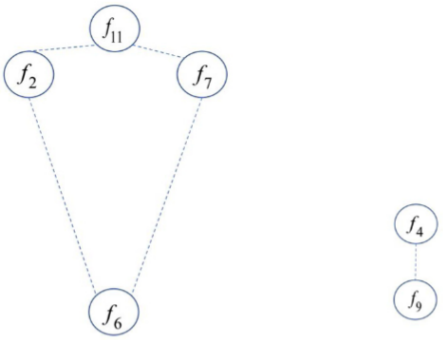
gem de subgrafo mínimo para o reconhecimento de características utilizando grafos envolve a criação de um grafo de adjacência com atributos que leva em consideração a concavidade das arestas, como exemplificado na figura 43. Nessa representação, as linhas pontilhadas indicam arestas côncavas, as quais são preservadas durante a análise, enquanto as demais arestas são eliminadas, juntamente com as faces que não possuem conexões côncavas. Como resultado, obtém-se as características isoladas, conforme evidenciado na figura 44.

Figura 43: AAG



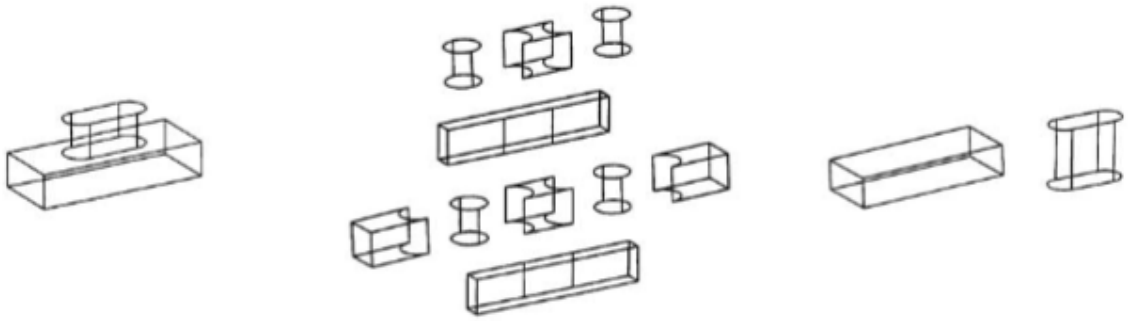
Fonte: extraído de [118]

Figura 44: Canal (esq.) e Furo (dir.)



Posteriormente, as abordagens baseadas em decomposição volumétrica representaram um avanço significativo, focando na divisão de um objeto em volumes menores para facilitar o reconhecimento de características como pode ser visto na figura 45. Esta metodologia é particularmente útil para identificar características que não são facilmente discerníveis através de métodos baseados em regras ou grafos. A abordagem de decomposição de volume consiste em dividir o material a ser removido de um modelo sólido em volumes intermediários, que são posteriormente combinados para gerar características de usinagem. Existem duas principais categorias de métodos de decomposição: a decomposição de casco convexo, eficaz para peças poliédricas, mas com dificuldades em superfícies curvas [119,120]; e a decomposição baseada em células, que divide o volume em células simples e as combina para formar características de usinagem [98, 121, 122]. Embora a decomposição baseada em células seja adequada para planejamento de fabricação e codificação NC, sua representação de características de volume carece de informações geométricas e topológicas, não sendo adequada para reconhecer a classe específica de características e apresentando desvantagens computacionais.

Figura 45: Decomposição volumétrica



Fonte: extraído de [98]

A partir da última década, houve um avanço significativo com a adoção de técnicas de aprendizado de máquina em técnicas de reconhecimento automático de características de usinagem. Algoritmos como redes neurais artificiais começaram a ser explorados para reconhecer características de maneira mais adaptativa, aprendendo com exemplos ao invés de seguir regras rígidas. Essa mudança foi um ponto de virada, pois permitiu o processamento de uma maior diversidade de características com menos esforço manual.

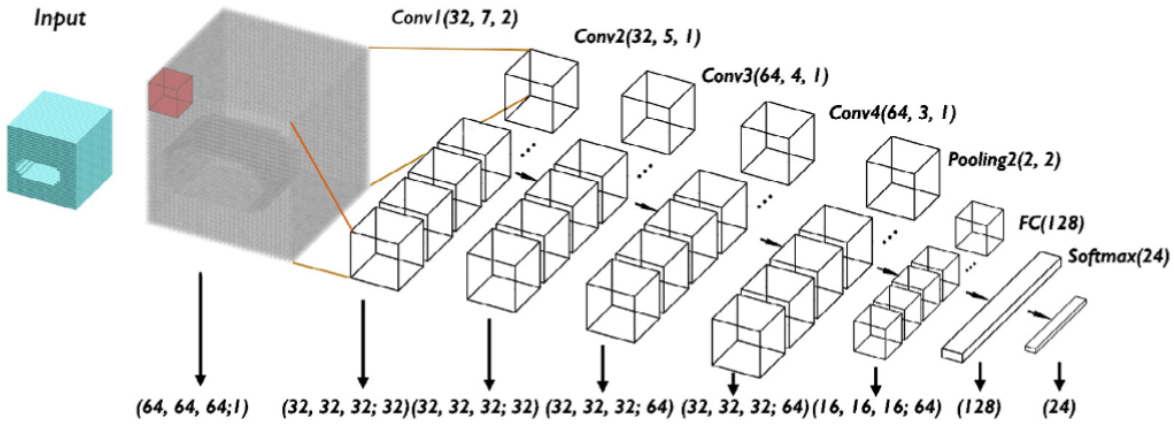
5.1.2 Reconhecendo peças com inteligência artificial

Com o avanço da inteligência artificial, e principalmente com os algoritmos de aprendizado profundo (*Deep Learning*), as técnicas de aprendizado de máquina têm sido cada vez mais aplicadas ao reconhecimento de características usináveis em modelos 3D. Estes métodos envolvem o treinamento de modelos computacionais a partir de uma grande quantidade de dados. Para o contexto de reconhecimento de características usináveis, existem muitos métodos empregados e que estão profundamente conectadas ao formato do dado disponível para o processamento. Além das abordagens baseadas em regras e de redes neurais menos complexas (MLPs) abordadas anteriormente, as Redes Neurais Convolucionais (CNNs) e Redes Neurais Gráficas (GNNs) têm ganhado cada vez mais espaço para desempenhar essa tarefa.

Com a popularização dos algoritmos de aprendizado profundo para tarefas de segmentação semântica, o foco se deslocou para diferentes formatos de dados. No trabalho pioneiro de [123], modelos 3D foram transformados em grades de ocupação volumétricas (*voxels*), e uma arquitetura convolucional com kernels tridimensionais foi capaz de reconhecer com precisão 24 tipos de características usináveis. Para este estudo, um con-

junto de dados contendo mais de 144.000 amostras foi gerado sinteticamente usando estratégias de aumento de dados. Como o conjunto de dados consistia em blocos quadrados contendo apenas uma característica por amostra, esse algoritmo usou a segmentação por *watershed* [124] para a análise de modelos CAD com várias características. Esta técnica é particularmente eficaz para identificar características isoladas em modelos CAD, beneficiando-se da clareza e da uniformidade da representação em voxel e do consolidado sucesso que essa arquitetura já possuía na esfera de imagens 2D. No entanto, um ponto fraco dessa abordagem é a perda de informações topológicas e geométricas durante a conversão do modelo CAD para a representação em voxel, o que pode levar a uma redução no desempenho de classificação. Embora o modelo tenha mostrado limitações no reconhecimento de características interagentes, ele abriu caminho para a exploração de camadas convolucionais nesse contexto. A Figura 46 mostra a arquitetura da rede convolucional 3D desenvolvida. Vale ressaltar que esse método possui uma exponencial escalada dos recursos computacionais necessários ao aumentar a resolução para contornar problemas de desempenho de classificação

Figura 46: Arquitetura 3D CNN - FeatureNet



Fonte: extraído de [123]

Subsequentemente à utilização de 3DCNNs, as redes baseadas em múltiplas vistas, como o MsvNet [125], ofereceram uma perspectiva interessante, convertendo modelos CAD 3D em uma série de imagens 2D. Estas imagens são então processadas por redes neurais visuais 2D já bem estabelecidas, o que permite a utilização de *transfer learning* [126] para alavancar resultados contundentes de arquiteturas de sucesso para essa tarefa. Esta abordagem é forte na identificação de características, mas enfrenta o desafio de perda de fidelidade durante a conversão o que pode limitar sua utilidade no planejamento de

processos subsequentes, uma vez que produz caixas de delimitação cúbicas aproximadas. Reconhecendo essas limitações, os autores propuseram a SsdNet [127] como uma evolução do MSVNet. A SsdNet é uma rede neural baseada em múltiplas vistas, inspirada no algoritmo de detecção de objetos *Single Shot Multibox Detector* (SSD) [128], a SsdNet modifica a arquitetura e a saída do SSD original para se adequar ao reconhecimento de características 3D em modelos CAD. Essa adaptação permite que a SsdNet lide eficientemente com a localização e o reconhecimento de características altamente interseccionadas, um desafio significativo para a MsvNet.

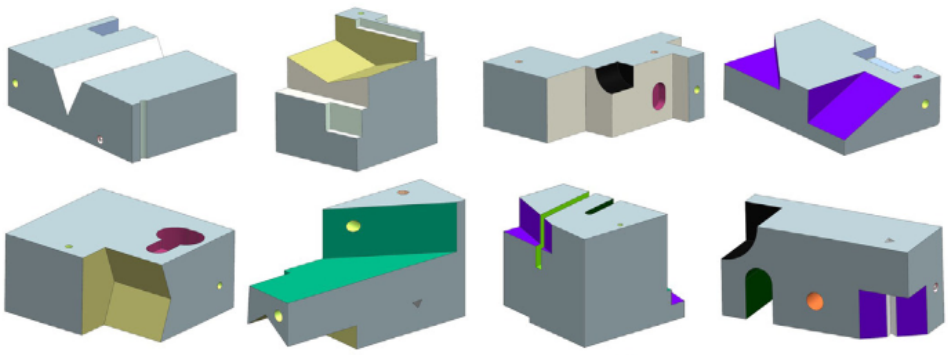
As redes baseadas em nuvem de pontos também ganharam espaço nesse domínio. Utilizando a arquitetura PointNet, o estudo apresentado em [129] utiliza dados de nuvem de pontos para o reconhecimento automático de características usináveis. Esta abordagem é capaz de capturar a forma e a superfície dos objetos de forma mais direta e intuitiva. No entanto, ela também enfrenta desafios: pequenas faces de características podem ser subamostradas durante a geração da nuvem de pontos, e os modelos de nuvem de pontos não capturam relações topológicas entre entidades B-Rep, como vértices, arestas e faces.

Mais recentemente, o estudo [130] apresenta um método que utiliza uma combinação de modelos MeshCNN [13] e Faster RCNN [131], empregando uma rede neural convolucional para o reconhecimento automático de características de usinagem com base em dados de malha de superfície. Um conjunto de dados de malha de características de usinagem é coletado a partir do modelo CAD e utilizado para treinar a rede neural. Apesar da pequena quantidade de dados necessária e do processamento de dados simples, o reconhecimento de características é baseado na tecnologia de Definição Baseada em Modelo (*Model Based Definition*), que nem sempre está disponível em aplicações reais.

Atualmente, as pesquisas estão se voltando para as redes neurais gráficas (GNNs), que são particularmente adequadas para dados estruturados em forma de grafos, como é o caso dos modelos CAD de peças usinadas. As GNNs podem capturar relações complexas entre as características de uma peça, permitindo uma análise mais detalhada e um reconhecimento mais preciso. Dentro das redes baseadas em grafos, temos como exemplo [132], proposto por uma equipe da *Autodesk*. Esta abordagem se destaca por operar diretamente em estruturas de dados B-Rep, explorando sua similaridade estrutural com grafos. Em uma abordagem inovadora, o modelo codifica a geometria dos parâmetros u, v das superfícies e curvas, além da topologia do Grafo de Adjacência de Faces (FAG). Apesar de suas vantagens, o UV-Net enfrenta desafios como a necessidade de um framework de aprendizado complexo e a dificuldade de representar eficientemente estruturas B-Rep variadas.

O trabalho pioneiro de [17] não apenas demonstra a eficácia das redes neurais convolucionais gráficas na abordagem dos desafios de segmentação semântica que é reconhecimento de características de usinagem, mas também contribui significativamente para a comunidade ao disponibilizar um extenso conjunto de dados sintéticos denominado MFCAD. Nessa técnica, cada face do modelo CAD é representada como um nó em um grafo, com arestas conectando faces que compartilham uma borda. Além disso, durante a etapa de criação da representação gráfica, uma equação paramétrica é calculada para descrever o vetor normal e a distância da origem. Esta representação gráfica, focada na geometria das faces, permite uma representação mais robusta, proporcionando resultados expressivos na identificação de características individuais e interagentes. Essa abordagem baseou-se nos avanços de [82] em relação à Rede Neural Convolucional de Gráficos Dinâmicos (DGCNN) e possui a limitação de ter sido projetado para operar apenas em sólidos com faces planas.

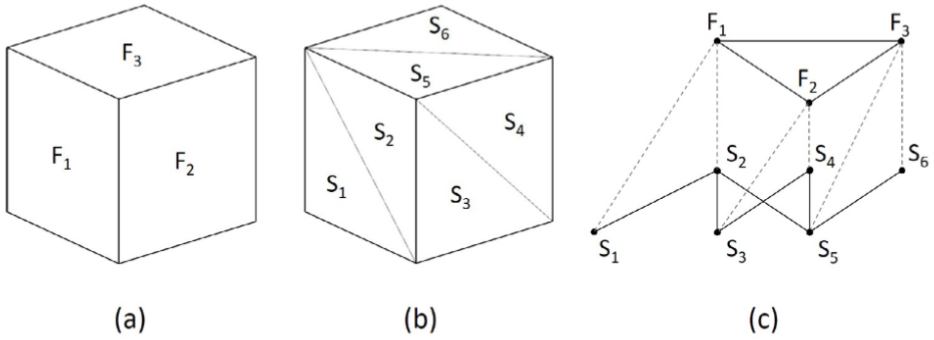
Figura 47: Amostras de modelo CAD do *dataset* MFCAD++



Fonte: extraído de [102]

Os mesmos autores expandiram essa pesquisa para incorporar uma representação hierárquica de modelos CAD, conforme demonstrado por um trabalho recente [102]. Visando superar as limitações de tipos de faces do modelo anterior, sua abordagem combina representações de grafos e de malhas, codificando efetivamente informações topológicas e geométricas. O trabalho opera dentro de um *framework* de *Graph-CNN* proposto por [83]. Embora o conjunto de dados *MFCAD* tenha permanecido sintético, esforços foram feitos para melhorar sua consistência lógica, melhorando sua usabilidade para fins de pesquisa e desenvolvimento. A Figura 48 mostra a representação hierárquica proposta no trabalho.

Figura 48: Representação gráfica hierárquica de um cubo. (a) B-rep, (b) Mesh, (c) Grafo hirárquico



Fonte: extraído de [102]

Mais recentemente em [133] o autor sugere uma rede neural multi tarefas para abordar o reconhecimento de características de usinagem. Esses vetores são extraídos usando uma arquitetura baseada na arquitetura PointNet [12], que codifica a geometria espacial das faces em um espaço de 32 dimensões. O principal desafio é a alta dimensionalidade, que pode obscurecer as relações significativas entre pontos de dados. O modelo bastante complexo divide a tarefa em três etapas, nomeadamente: segmentação de instância (localização da características), segmentação semântica (classe da característica) e reconhecimento de face de fundo.

O estudo propõem um método de segmentação de características de usinagem em imagens de peças mecânicas. Eles usam uma matriz de similaridade $S^{(1)}$ para avaliar a probabilidade de faces pertencerem à mesma característica. Reduzem a dimensionalidade dos dados com a utilização de uma rede neural MLP, gerando $S^{(2)}$. A média de $S^{(1)}$ e $S^{(2)}$ cria $S^{(3)}$, equilibrando detalhes e generalização. Aplicam um algoritmo de agrupamento por deslocamento-médio (*mean-shift*) [134] para gerar $S^{(4)}$, representando grupos iniciais de faces. Em seguida, aplicam uma regra limiar a $S^{(3)}$ e $S^{(4)}$ para criar $S^{(5)}$, que define agrupamentos finais de faces em características de usinagem.

O módulo de segmentação semântica recebe uma matriz de dimensões $N_f \times 32$, onde N_f é o número de faces na peça. Ele produz um rótulo de classificação para cada face, resultando em um mapa de segmentação semântica de dimensões $N_f \times N_c$, onde N_c é o número de classes diferentes de características de usinagem. O processo começa com uma extração de características, usando duas redes MLP (perceptrons multicamadas) para aprimorar as características das faces. Essas MLPs, ativadas por funções de ativação leaky ReLU, são seguidas por uma camada de max-pooling para obter um vetor de carac-

terísticas global que representa a peça como um todo. Em um ciclo de retroalimentação (*feedback loop*) inovador, o vetor de características global é combinado com os vetores de características individuais de cada face, incorporando informações locais e contextuais. Esse vetor combinado passa por outra transformação MLP, seguida por uma função de ativação softmax, gerando a matriz $N_f \times N_c$. Cada linha dessa matriz contém uma distribuição probabilística das classes de características de usinagem para cada face, atribuindo rótulos semânticos. A arquitetura deste módulo é complexa, com várias camadas MLP que refinam progressivamente as características. Começa com uma camada MLP de 64 filtros e se expande para camadas adicionais com 64, 128 e 1024 filtros antes do estágio de max-pooling. Após a concatenação de características, as últimas etapas da MLP incluem camadas com 512, 256 e 128 filtros, antes de ramificar para funções softmax e sigmoid para segmentação semântica e identificação da face inferior, respectivamente.

Por fim, o módulo de reconhecimento da face inferior apresenta uma arquitetura semelhante ao módulo de segmentação semântica, mas difere na função de ativação adotada. Ele utiliza a função de ativação sigmoide devido à natureza da tarefa, que envolve uma classificação binária, em contraste com o módulo anterior que lida com classificação multiclasse. Essa etapa é fundamental para a aplicação prática da tecnologia pois a informação resultante é de extrema importância as etapas subsequentes do processo de usinagem, como o planejamento do trajeto da ferramenta e outras operações relacionadas.

Além disso, no campo do reconhecimento de características de usinagem, várias representações de dados têm sido utilizadas por modelos fundamentais de aprendizado profundo, que aproveitam as capacidades convolucionais 2D e 3D. Exemplos notáveis dentro dessas representações incluem a assinatura do kernel de calor (HKS) apresentada por [7], a abordagem de visualização seccional múltipla (MSV) de [125], dados de nuvem de pontos demonstrados por [129] e, em desenvolvimentos mais recentes, a utilização de malhas 3D, como no trabalho de [130].

As abordagens de reconhecimento de características na usinagem têm pontos fortes e fracos variados, refletindo a complexidade em constante evolução do campo. À medida que a tecnologia avança, essas abordagens continuam a se desenvolver, oferecendo novas soluções para os desafios da manufatura assistida por computador. A integração de diferentes técnicas de reconhecimento de características é um campo de pesquisa ativo, visando melhorar a precisão e eficiência por meio da combinação de várias abordagens. Isso reflete uma tendência de aumento na automação e precisão na manufatura, alinhada com a necessidade de processos mais eficientes.

5.2 Estimando custos com inteligência artificial

Após as comprovações contundentes da capacidade das redes neurais profundas em reconhecer padrões em objetos tridimensionais, não demorou para que pesquisadores explorassem essa tecnologia para tentar suprir as limitações das metodologias tradicionais de estimativa de custo de peças usinadas. Em uma série de trabalhos bastante expressiva *Ning et al* apresentou os avanços alcançados em reconhecimento de padrões para auxiliar em tarefas de regressão nesse domínio. Segundo os autores, modelos de regressão tradicionais descrevem relações de mapeamento complexas com um pequeno número de amostras. No campo da usinagem, a diversidade das características construtivas possíveis e a heterogeneidade das relações entre essas características com os processos fabris, apresentam-se desafiadores para os modelos tradicionais pois o aumento na quantidade e dimensões dos dados tornando o mapeamento complexa. Além disso, essas características possuem uma expressividade baixa o que leva aos modelos uma limitada robustez em variações de condições de contorno. E por isso, o método de aprendizado profundo teria a capacidade de aprender relações de mapeamento complexas e características de dados de alto nível a partir de um grande número de dados automaticamente. Portanto, um modelo de regressão baseado em uma rede neural profunda pode efetivamente reduzir o impacto dos problemas mencionados na previsão de dados e melhorar a precisão da regressão [135].

O uso de técnicas avançadas de aprendizado profundo na estimativa de custo de usinagem é um campo em desenvolvimento. Essas técnicas oferecem uma abordagem mais precisa e adaptativa, o que se torna crucial diante das crescentes complexidades e demandas do mercado. Nesta seção, vamos explorar o impacto do aprendizado profundo na estimativa de custos de usinagem, apresentando as metodologias recentes e pesquisas relevantes.

5.2.1 Modelos existentes

Um dos primeiros trabalhos que vieram a explorar a capacidade de abstração das redes neurais profundas para estimar o custo de processo em peças usinadas, é o apresentado em [20]. O trabalho conduz uma série de pesquisas operacionais para definir quais os parâmetros de entrada relevantes para uma rede neural estimar o tempo de processo de peças usinadas. A argumentação dos autores é que o tempo total de produção (TPT - *Total Production Time*), é componente fundamental e determinante do custo total de produção (TPC - *Total Production Cost*). Durante a pesquisa 10 empresas de usinagem foram consultadas para responder duas questões centrais à modelagem. Quais eram os

requisitos do projeto de uma peça usinada que mais impactava no custo de produção, e como esses fatores poderiam ser quantificados, **em intervalos**, de forma a simplificar mas ao mesmo tempo sendo representativos. A tablea 2 indica a metodologia adotada no trabalho.

Tabela 2: Normalização utilizada no trabalho

No.	Variáveis de Entrada	Descrição	Faixa	Valor
1	Volume	Diâmetro, mm	10–150	0.25
		Comprimento, mm	50–180	0.25
			181–250	0.5
			251–300	1
2	Maquinabilidade	Duro	Aço Fundido	0.5
		Médio	Latão	0.6
		Fácil	Alumínio	0.4
3	Dureza	Dura	Teflon	0.2
		Média	Aço Suave	0.5
		Fácil	Molibdênio	0.7
4	Ferramenta Auxiliar	Quatro castanhas		1
		Mandril de três castanhas		0.5
		Dois centros		0.25
		Centros móveis		0.9
5	Rugosidade da Superfície, Ra, µm	Muito Baixa	0.2–0.8	1
		Baixa	1.6–3.2	0.5
		Média	3.2–6.3	0.2
6	Material da Ferramenta	Alta velocidade	Aço	0.8
		Médio	Carboneto	0.5
		Baixa	Cerâmica	1
7	Características de Torneamento	Externo	Acabamento	0.3
			Desbaste	0.2
			Corte	0.4
			Partir	0.7
		Interno	Acabamento	0.3
			Desbaste	0.2
			Furação	0.5
			Broca	0.7
			Rosca	0.9
			Torneamento	1

Fonte: elaborado pelo autor

Após a normalização dos dados, os autores conduzem o experimento estatístico baseado em uma amostra de 153 peças considerando apenas operações de torneamento. A arquitetura da rede neural desenvolvida possui três camadas ocultas e o número de neurônios é definido empiricamente através da análise dos resultados. Testes também são executados em relação ao algoritmo otimizador de gradientes. Os testes conduzidos mostram números de neurônios que variam de 10 a 18 e dois algoritmos otimizadores a Regularização Bayesiana (*trainbr)* [136] e o Levenberg-Marquardt (*trainlm)* [137]. A configuração que obteve os melhores resultados foi com a utilização de 10 neurônios, função de ativação tansig e otimizador Levenberg-Marquardt. Os resultados apresentados ates-

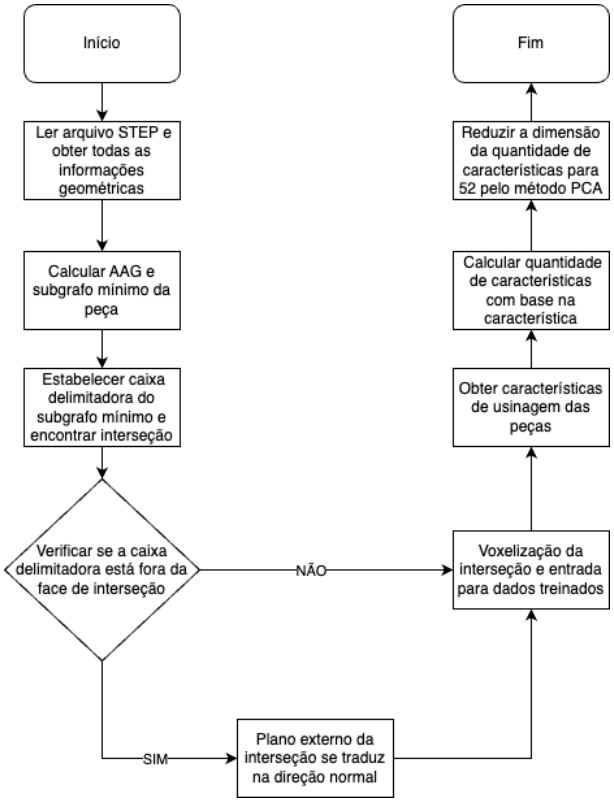
tam baixos erros na saída do modelo bem como uma considerável significância estatística. Os autores também reivindicam que esse foi o primeiro estudo dessa natureza e por essa razão não seria possível obter uma base de comparação.

Em [135], o estudo mostrou a superioridade de uma rede neural convolucional 3D (3D-CNN), perante uma rede neural convolucional 2D (2D-CNN), na tarefa de predição do custo de uma peça usinada. A rede 2D foi alimentada com imagens de 224px x 224px x 24px e contou com alguns mecanismos de aperfeiçoamento do *dataset* como rotação, redimensionamento e normalização das cores da imagem. Os modelos de entrada da rede 3D passaram por um processo de *voxelização* desenvolvido pelos próprios autores através de *octrees* e também contou com o aumento da quantidade de dados através da rotação nos três eixos. Além disso, diferentes resoluções foram testadas. O trabalho conclui que a rede 2D apenas possui informações da superfície da peça e por isso é incapaz de ”entender completamente”as características relevantes para a determinação do custo. Os autores indicam o potencial da aplicação do método na indústria ainda que reconheçam que existe uma lacuna considerável a ser coberta antes de poder ser empregado em peças com caracteríticas complexas ou que possuam requisitos de tolerância ou rugosidade.

Em outro trabalho subsequente, os mesmos autores apresentaram um estudo que combinava o reconhecimento de características usináveis e uma forma inovadora de representar essas características denominada *feature quantities*. O objetivo final do trabalho é estabelecer uma relação entre essas representações e o custo de produção da peça usinada através de algoritmos de regressão. Como é comum nesse tipo de algoritmo, a estratégia de reconhecimento de característica, conta com uma etapa de segmentação anterior ao processamento. A metodologia empregada envolve calcular o grafo de faces adjacentes (AAG) e determinar através de cálculos geométricos a concavidade ou convexidade de todas as arestas do modelo. Para a determinação final, as arestas convexas são retiradas para a geração dos mínimos sub-grafos das características. A partir daí a técnica utiliza do conceito de sólido delimitador para determinar o plano de intersecção com a peça e isolá-las antes de serem voxelidas (da mesma forma que no trabalho anterior) para posteriormente serem alimentadas à rede neural convolucional 3D. Diferentemente do trabalho anterior em que a 3D-CNN foi utilizada para regressão, nesse trabalho, o uso da rede foi feito para a tarefa de **segmentação semântica**. O treinamento da rede foi feito através de um *dataset* contendo 14 características usináveis isoladas e voxelizadas que também passaram pelo processos de ampliação rotacional nos 3 eixos X, Y e Z. Após a determinação das características usináveis, os dados passam por um pré-processamento que cria o vetor característica que será inserido no modelo de regressão. Primeiramente as

características reconhecidas são quantificadas em termos de número de ocorrências, e conceitos geométricos relevantes como volume, área e dimensões. Após esse processo os dados passam por uma estratégia de redução **linear** de dimensionalidade através da análise de componentes principais (PCA). A imagem 49 apresenta um fluxograma simplificado do algoritmo de extração e representação de características usináveis.

Figura 49: Algoritmo de extração e representação de características usináveis

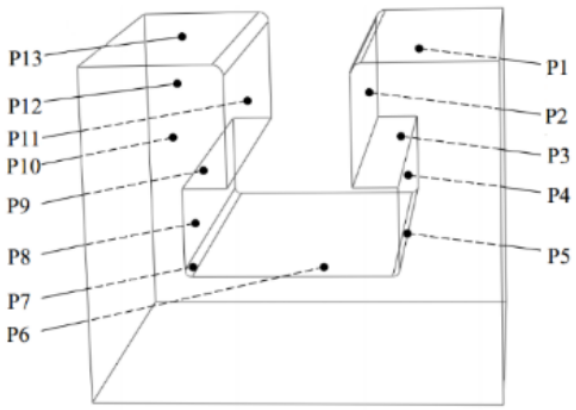


Fonte: extraído de [118]

O estudo ainda faz uma comparação entre dois algoritmos para a tarefa de regressão, SVM e *Backpropagation* (ANN). Ambos os dados recebem dados normalizados e são treinados usando a estratégia de validação cruzada (*cross validation*). Para o SVM, a função de kernel utilizada é a RBF (*Radial Basis Function*) e a melhor configuração foi capaz de produzir um erro de pouco mais de 21% enquanto a rede neural usando o otimizador *Adam* e neurônios sigmoidais, gerou um erro menor, em torno de 8%. A **precisão** de ambos os algoritmos foi avaliada usando a função MAPE (*Mean Absolute Percentage Error*). Ainda que os autores reconheçam a necessidade de adicionar informações de precisão, como tolerância e rugosidade, não existe menção à dificuldade de reconhecer características interagentes, o que sabidamente é um desafio, e dada às características do algoritmo proposto provavelmente seria um obstáculo a ser superado

O teceiro estudo aponta para a direção de cobrir as lacunas deixadas nos dois trabalhos anteriores, buscando endereçar tanto a capacidade de reconhecimento de características complexas e interagentes como também em codificar mais informações pertinentes a fabricação como tolerância e rugosidade. A abordagem adotada combina reconhecimento de padrões sintáticos e restrições de características (*feature constraints*) para desempenhar a segmentação semântica. Padrões sintáticos, em oposição a vetores de características, é uma abordagem *estrutural* de modelamento que adota uma linguagem formal que se apoia em estruturas gramaticais. Padrões complexos geralmente são formado por uma junção de padrões estruturais mais simples, formando uma hierarquia que termina em padrões elementais, também chamados de *primitivos*. O reconhecimento de um padrão desconhecido pode ser executado por um *parser* (uma implementação formal de um autômato) [138]. As restrições de características por sua vez são baseadas nas propriedades geométricas e topológicas de um modelo 3D e possuem um "vocabulário"que agregam conceitos como "Aresta de Junção"(*Joint-Edge*), Paralelo (*Parallel*), Coplanar, entre outros. O algoritmo testa cada uma das regras pré definidas com a finalidade de encontrar o padrão de correspondência. Para melhorar a eficiência, o algoritmo utiliza de "dicas"e uma direção de busca específica com a finalidade de identificar a primeira face de uma característica específica. Todas as faces são percorridas através das arestas de junção antes de descartar a ocorrência de uma característica específica. A imagem 50 ilustra como seria a regra de definição da uma característica usinável segundo a metodologia proposta.

Figura 50: Definição da característica *canal em "T"*



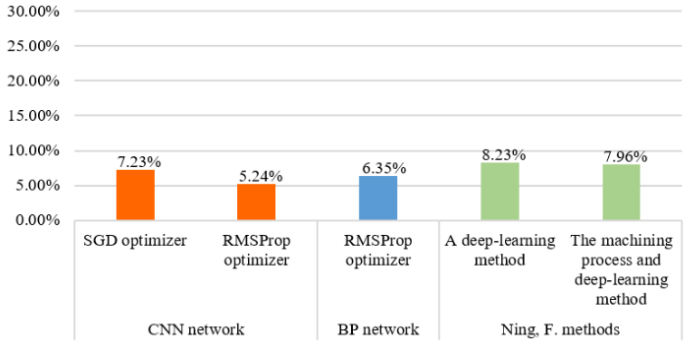
The T-Groove feature faces include P2–P9 and P11. The defining order of the faces is P11→P9→P8→P7→P6→P5→P4→P3→P2. The T-Groove syntax pattern is defined as follows: Joint-Edge (Joint-Edge (Joint-Edge (Joint-Edge (Joint-Edge (Joint-Edge (Joint-Edge (Joint-Edge (P11, P9,1.5π)P9(π), P8,0.5π)P8(π),P7, π)P7(π), P6,π)P6(π), P5,π)P5(π), P4,π)P4(π), P3,0.5π)P3(π), P2,1.5π)P2(π) ∩Parallel(P11,P8) ∩Parallel(P11,P4) ∩Parallel(P11,P2) ∩Parallel(P9,P3) ∩Coplanar(P7,P5).

Fonte: extraído de [139]

Após a etapa de reconhecimento o algoritmo faz a representação dessas características de acordo com as regras pré determinadas para cada uma delas e combina com características globais da peça antes de alimentar os algoritmos de regressão. Para esse trabalho os autores exploraram uma rede convolucional (2D-CNN) e uma rede neural com 4 cama-

das densas. Experimentos foram feitos acerca do otimizador e resultados melhores foram obtidos com a a utilização do RMSProp (*Root Mean Square Propagation* quando comparado ao SGD (*Stochastic Gradient Descendent).* Usando o mesmo *dataset* e a mesma metodologia de avaliação, os autores compararam o desempenho final não apenas para as duas estratégias de regressão mas também com os dois trabalhos anteriores. O gráfico comparativo é apresentado na figura 51.

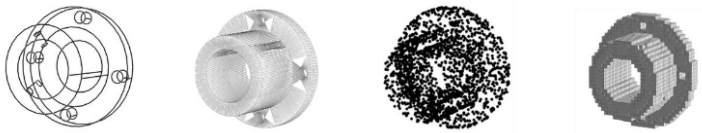
Figura 51: Comparação do desempenho de todos os trabalhos



Fonte: extraído de [139]

Outro estudo da mesma época [140] analisou comparativamente diversos modelos disponíveis para a tarefa de estimativa de custo de fabricação, utilizando um *dataset* de 1006 peças obtidas a partir de um *e-commerce* de peças usinadas global chamado MiSUMi [141]. Os autores partiram da hipótese que o custo de fabricação ocupa uma proporção fixa do preço encontrado na plataforma online, sob o argumento de que existe uma forte correlação entre o custo de produção e o preço. Também foi extraído os modelos 3D em formato *step* e a liga metálica usada para fabricação de cada um dos produtos. Os modelos passaram por três etapas de pré processamento com objetivo de serem usados nos modelos. Primeiramente foi gerado um malha 3D (*mesh*) usando a interface de programação de aplicações (API) do FreeCAD [142]. Em seguida, a malha 3D foi convertida em voxels seguindo o algoritmo proposto em [143]. Por último através de uma amostragem aleatória dos triângulos da malha foi gerado uma nuvem de pontos usando o algoritmo proposto em [144]. A figura 52, extraída do texto, mostra um exemplar do dataset nos quatro formatos.

Figura 52: Pré processamento dos modelos 3D



Fonte: extraído de [140]

Após normalização dos dados (volume e custo) e codificação da variável categórica (material) o estudo traz um *benchmark* de 5 modelos já consolidados bem como a proposta de uma nova arquitetura utilizando uma 3DCNN baseada em voxels. Diferentes estratégias de normalização, função de perda e performance foram testadas. A tabela 3 extraída do estudo relata os resultados encontrados.

Tabela 3: Comparação entre os modelos testados

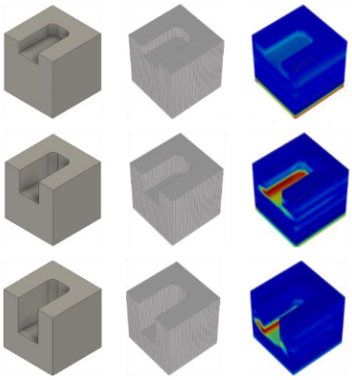
Características dos modelos				Performance	
Arquitetura	Entrada	Normalização	Função de perda	RMSE	MAPE
Voxnet	Voxel(32)	Min-max	MAE	**2716.24***	29.76
	Mat, Vol	Log	MSLE	3109.07	18.58
PointNet	Point(2048) Mat, Vol	Log	MSE	**3503.78**	**21.17**
Dering and Tucker (2017)	Voxel(32)	Min-max	MAE	**1615.86**	21.09
	Mat, Vol	Log	MSE	2014.87	12.93
Williams et al (2019)	Voxel(64) Mat, Vol	Log	MAE	**2165.35**	**14.12**
Ning et al (2020)	Voxel(64)	Log	MSE	1047.47	**10.63**
	Mat, Vol	Log	MSLE	**1002.55**	16.30
Proposed	Voxel(32)	Min-max	MAE	**1233.06**	20.58
	Mat, Vol	Log	MAE	1290.41	**8.76**

Fonte: elaborado pelo autor

Um aspecto bastante interessante desse estudo é a utilização uma estretágeia de *XAI* (*Explainable AI*) chamada Grad-CAM (*Gradient-weighted Class Activation Mapping*) inicialmente proposto por [145]. Essa técnica de interpretação de redes neurais convolucionais pode ser aplicada em redes convolucionais e ajuda a entender quais partes dos dados

de entrada são mais importantes para a decisão do modelo. O Grad-CAM utiliza os gradientes da saída da rede neural em relação às ativações da última camada convolucional para criar um mapa de ativação. Esses gradientes indicam quais partes das ativações da camada anterior tiveram um impacto maior na decisão final da rede. O mapa de ativação resultante pode ser redimensionado, por interpolação, e sobreposto à imagem original, permitindo destacar as regiões da imagem que foram mais importantes para a decisão do modelo, criando assim um mapa de calor interpretável. Isso é especialmente útil para explicar por que a rede classificou uma imagem de uma determinada maneira, fornecendo **transparência** e **interpretabilidade** ao processo de tomada do modelo. A figura 53 mostra o resultado do experimento em que o modelo consegue identificar a região com a maior influência no custo.

Figura 53: Visualização do impacto da profundidade no custo através da técnica de 3D Grad-CAM



Fonte: extraído de [140]

Mais recentemente, no estudo apresentado por [109], foi concebida uma arquitetura de aprendizado profundo que se destina à estimativa de custos de usinagem. Esta arquitetura, batizada como Cost Estimation Network (CEN), compreende dos seguinte módulos interconectados: Processamento de características usináveis, convolução, fusão e regressão. O algoritmo CEN proposto emprega diferentes paradigmas de aprendizado profundo com o objetivo de alcançar o **estado da arte** em estimativa de custo de peças usinadas. A sua abordagem integrativa, lança mão da flexibilidade dos grafos, para explorar o holismo das redes neurais convolucionais gráficas (ConvGNNs) e combiná-la com a discriminação local da rede neural de passagem de mensagens (MPNN) em uma arquitetura multi-tarefas (*multi-task*).

Inicialmente, a etapa de pré processamento conta com as informações extraídas de um algoritmo de reconhecimento de características de usinagem, que não foi mencionado no

estudo, e cria um Attributed Adjacency Graph (AAG). Essa representação, além de possuir as características geométricas e topológicas da características usináveis reconhecidas, inclui também as relações entre essas características. Podendo ser de quatro tipos diferentes: Interação de Dependência, Interação de Adjacência, Interação de Padrão, Interação de Dado.

Em seguida, o algoritmo proposto usa uma arquitetura MLP (Multi Layer Perceptron) para digitalizar tanto as informações de nós (faces) quanto suas conexões (arestas). As informações dos vértices digitalizadas compreendem o tipo da característica (como furos ou rebaixos), as informações geométricas (como dimensões e orientação) e informações de precisão (rugosidade, geométrica e dimensional).

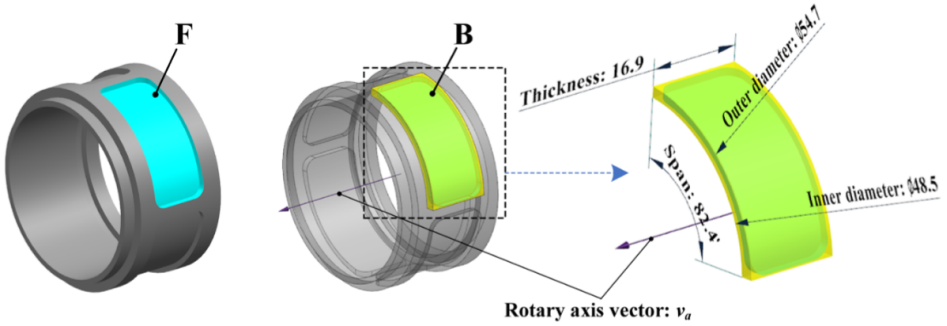
No módulo de convolução gráfica, foram implementadas linhas paralelas de convolução, incluindo a Message Passing Neural Network (MPNN) e a Chebyshev Convolution (ChebConv). Essas linhas capturam e integram nuances locais e globais na estrutura do grafo. Um módulo de fusão é utilizado para combinar as informações do material utilizado (codificado em one-hot) com a saída da convolução gráfica, que é então inserida em um módulo de regressão para prever o custo de fabricação. A arquitetura também incorpora técnicas de aprendizado residual para melhorar o fluxo de gradientes e é complementada por um processo modificado de Grad-CAM, proporcionando interpretabilidade para as previsões de custos em cenários de regressão. Para avaliar o desempenho do modelo, o estudo introduz a função de perda Root Mean Squared Logarithmic Error (RMSLE) 5.1, que se mostra especialmente adequada em comparação com MSE, MAE e RMSE convencionais. Ela ajuda a mitigar o impacto desproporcional de erros em previsões de custos muito altos, ao mesmo tempo em que lida eficazmente com custos próximos ou iguais a zero.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + \gamma) - \log \hat{y}_i + \gamma)^2} \tag{5.1}$$

Além das contribuições metodológicas no tratamento de grafos, o estudo de *Zhang et al.* apresenta uma inovação detalhada no que concerne à representação espacial de componentes na estimativa de custo. Para retratar com precisão as características de usinagem rotativas — como furos, ranhuras anulares — os autores desenvolveram um conceito inovador de sólido delimitador anular. Esse novo tipo de *bounding box* é descrito por parâmetros geometricamente relevantes: diâmetro interno, diâmetro externo, espessura e extensão angular 54. Juntamente com o vetor de orientação, essas informação fazem parte do AAG criado da peça. O modelo proposto pela equipe, a Cost Estima-

tion Network (CEN), apresentou bons resultados utilizando um *dataset* com custos de fabricação das peças geradas pela simulação por meio de um software baseado no Siemens NX. Os autores também apresentaram um estudo comparativo com outros dois trabalhos de referência [135] e [140]. O algoritmo proposto demonstrou resultados superiores quando comparado a ambos. Segundo os próprios autores, a superioridade do CEN pode ser atribuída principalmente à sua habilidade de incorporar informações de precisão e à perda de resolução associada às estratégias de redes neurais convolucionais 3D baseadas em voxel em ambos os estudos comparados.

Figura 54: Sólido delimitador anular



Fonte: extraído de [109]

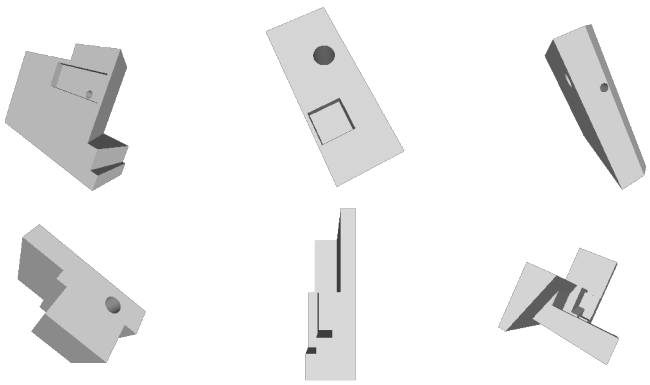
6 MATERIAIS E MÉTODOS

No capítulo anterior, consolidamos a base teórica necessária para compreender o conjunto de ferramentas empregado neste trabalho. Esse alicerce teórico, agora estabelecido, nos permite avançar para a exploração do experimento prático, como mencionado anteriormente, projetado para a estimativa de tempos de processo de peças usinadas baseado em redes neurais gráficas.

6.0.1 Criação dos dados de treinamento

Usando o algoritmo descrito em [102] como base e a ferramenta de modelagem 3D PythonOCC [146], foram gerados 200 modelos CAD sintéticos. Restrições de complexidade foram aplicadas e apenas algumas características específicas foram utilizadas. O propósito dessas restrições é simplificar o cálculo do tempo de usinagem para que possa ser feito de forma direta e precisa. A figura 55 mostra alguns exemplos dos modelos gerados.

Figura 55: Exemplos dos modelos gerados

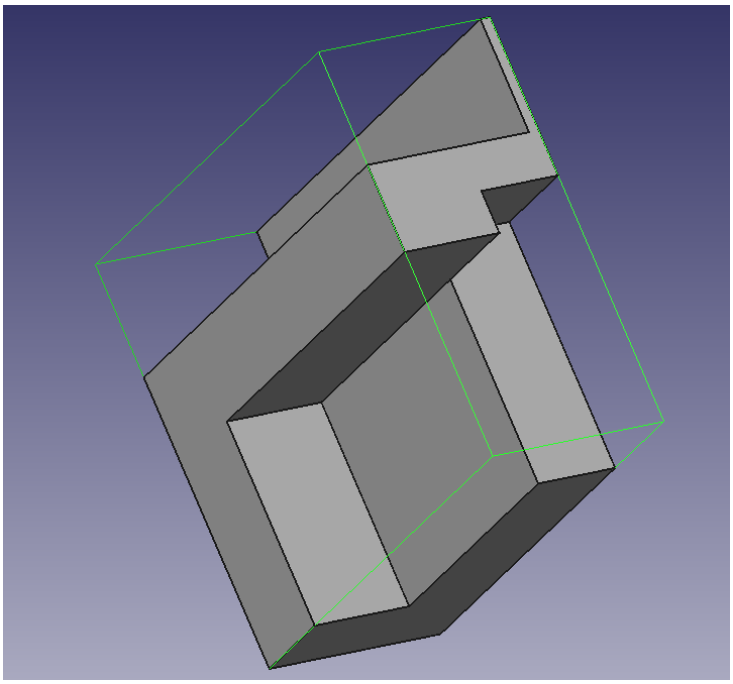


Fonte: elaborado pelo autor

Após a geração dos modelos, é calculado o tempo de usinagem com base na identificação das características, seleção do processo e cálculo da taxa de remoção de material conforme descrito no capítulo 2. Também é calculado o volume total de material removido pela diferença entre o volume do sólido e o volume da caixa delimitadora. O conceito de

sólido delimitador é apresentado na Figura 56. O tempo total de usinagem é fornecido em minutos, e o volume removido é medido em mm^3. Para simplificação, todas as peças foram consideradas feitas do mesmo material, aço SAE 1020. A Tabela 4 fornece um exemplo de como os dados são obtidos.

Figura 56: Sólido delimitador (*bouding box)*



Fonte: elaborado pelo autor

Tabela 4: Conjunto de dados contendo tempo de processo e volume removido

Nome do arquivo	Material	Tempo de processo	Volume removido
111.step	SAE 1020	8.90	17671.61
286.step	SAE 1020	16.67	62392.41
165.step	SAE 1020	15.77	50947.41
218.step	SAE 1020	9.21	43066.38
92.step	SAE 1020	8.65	42383.17

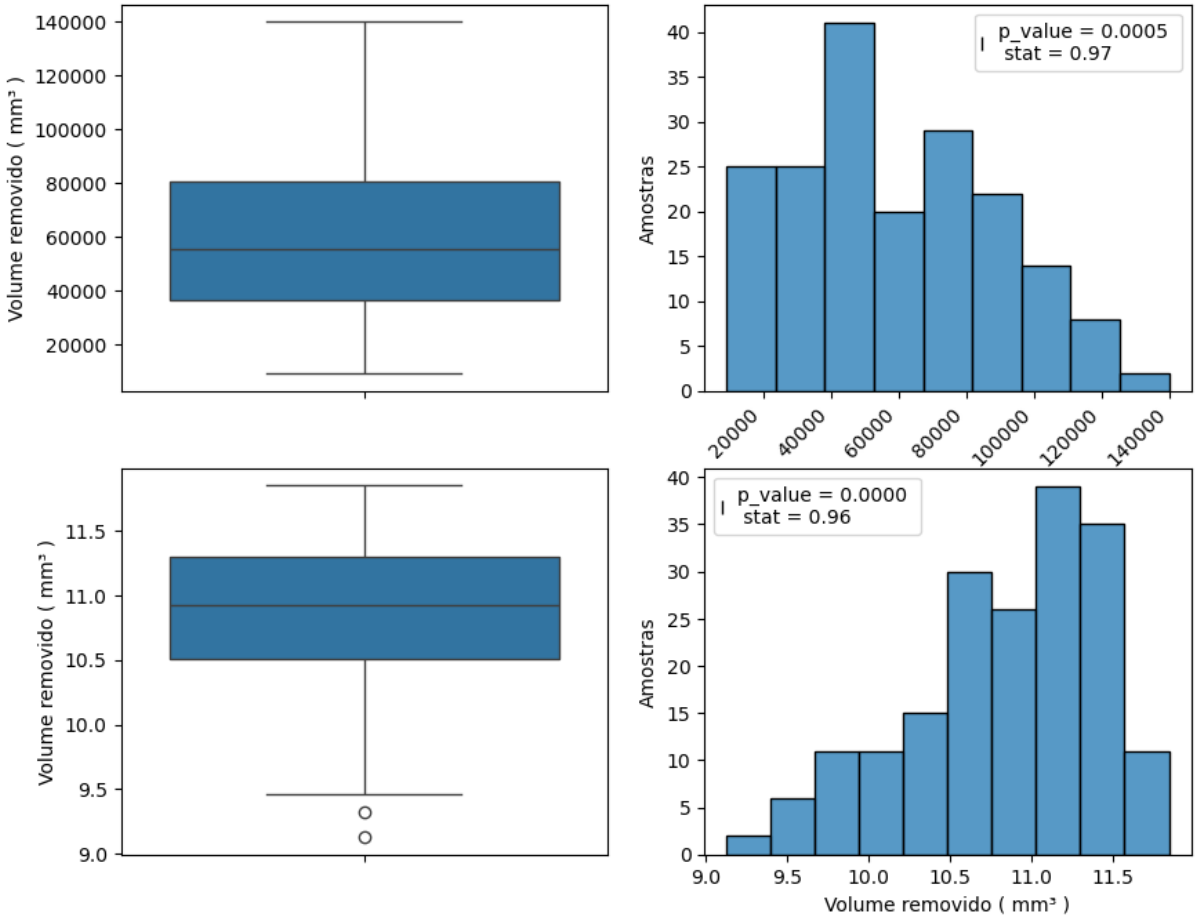
6.0.2 Normalização dos dados

Para preparar os dados para entrada em modelos de aprendizado profundo, é desejável um processo de normalização. Este processo visa transformar os dados para se aproximar de uma distribuição normal, possibilitando que algoritmos de otimização convencionais, como o *backpropagation*, funcionem de forma eficaz. Sem a normalização, o modelo pode ser altamente suscetível a instabilidades e até mesmo de convergência.

6.0.2.1 Volume removido

Para os dados do volume removido foi adotada a normalização logarítmica conforme a equação $volume_n = \ln(1 + x)$. A figura 57 apresenta os dados de **volume removido**, em milímetros cúbicos, antes e depois de aplicar a normalização.

Figura 57: Normalização dos dados - Volume removido



Fonte: elaborado pelo autor

6.0.2.2 Tempo de processo

Para os dados do tempo de processo, que é a variável dependente do modelo, foi adotada a normalização Box-Cox, essa estratégia tem a características de ser especialmente adequada para dados nitidamente assimétricos [147]. A transformação de Box-Cox é definida como segundo a equação 6.1. A figura 58 apresenta os dados de **tempo de processo**, em minutos, antes e depois de aplicar a normalização.

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{se } \lambda \neq 0 \\ \log(y), & \text{se} \lambda = 0 \end{cases} \tag{6.1}$$

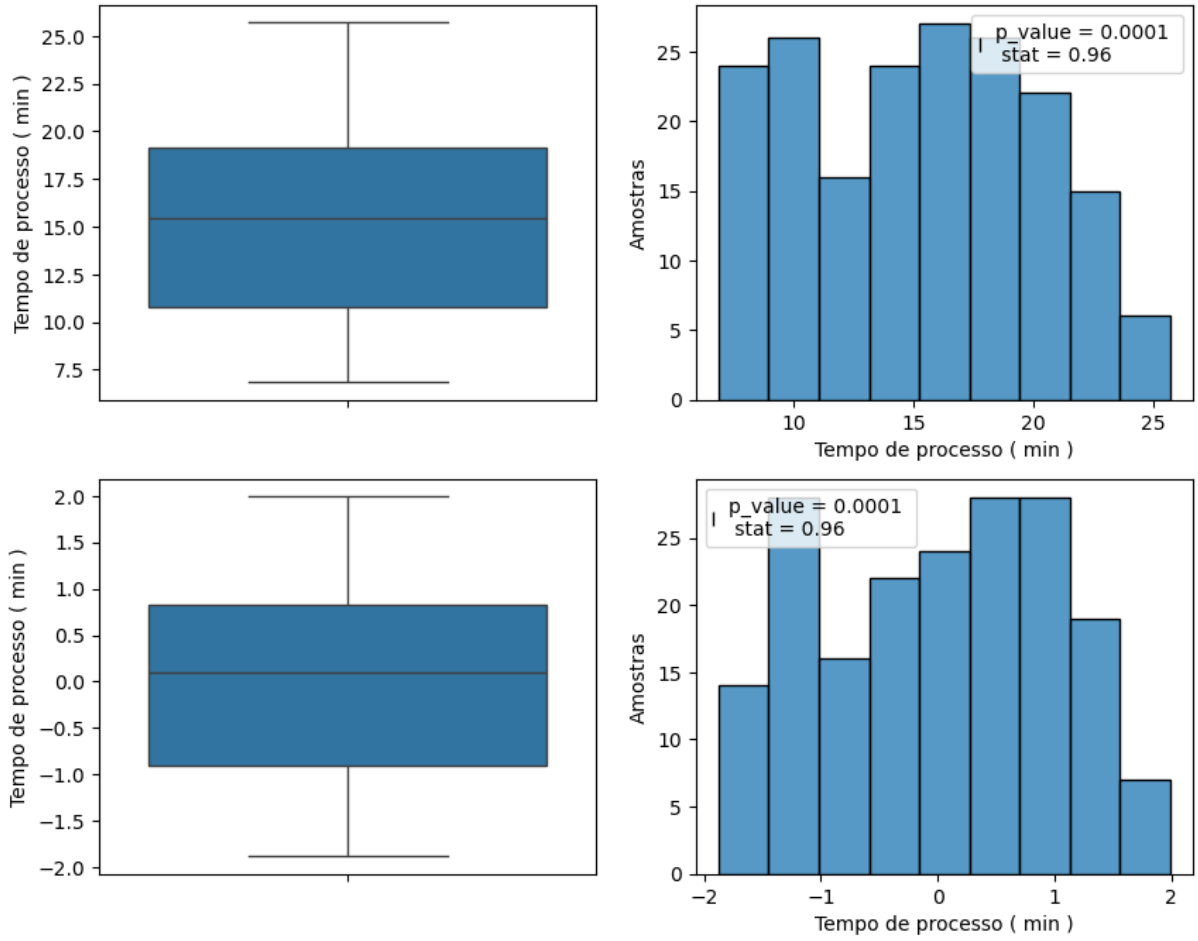
Onde:

$y(\lambda)$ - O valor transformado usando o parâmetro λ

y - O valor original a ser normalizado

λ - O parâmetro de transformação, escolhido para otimizar a normalização dos dados

Figura 58: Normalização dos dados - Tempo de processo



Fonte: elaborado pelo autor

6.1 Pré-processamento dos modelos

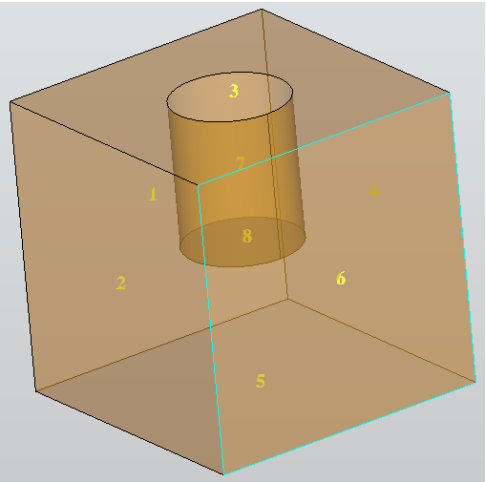
Para a manipulação dos modelos, foi adotada a ferramenta PythonOCC [146] que é uma biblioteca *open source* que se baseia no Open CASCADE Technology (OCCT)

[148] A ferramenta oferece uma interface Python para acessar muitas das funcionalidades do OCCT, permitindo a criação e manipulação eficiente de modelos 3D, operações geométricas avançadas e visualização interativa. Através dessa biblioteca será criado o AAG (*Attributed Adjacency Graph)* de cada um dos modelos.

6.1.1 Identificação das faces

O primeiro passo necessário para criar uma representação do modelos 3D, baseado em um grafo de adjacência, é identificar corretamente as faces do modelo de uma forma que seja facilmente acessada pelas diferentes etapas do experimento. A estratégia adotada nesse trabalho é a inserção de um identificador único numérico no próprio modelo CAD. Dessa forma não existe a necessidade de persistir essa informação em algum elemento externo já que ela sempre estará disponível em tempo de execução através do próprio arquivo que está sendo manipulado. A figura 59 mostra um sólido 3D com suas faces devidamente identificadas e sua identificação exibida ao centro de cada uma delas. Na imagem, a face de número **6** se encontra destacada.

Figura 59: Identificação das faces de um modelo 3D



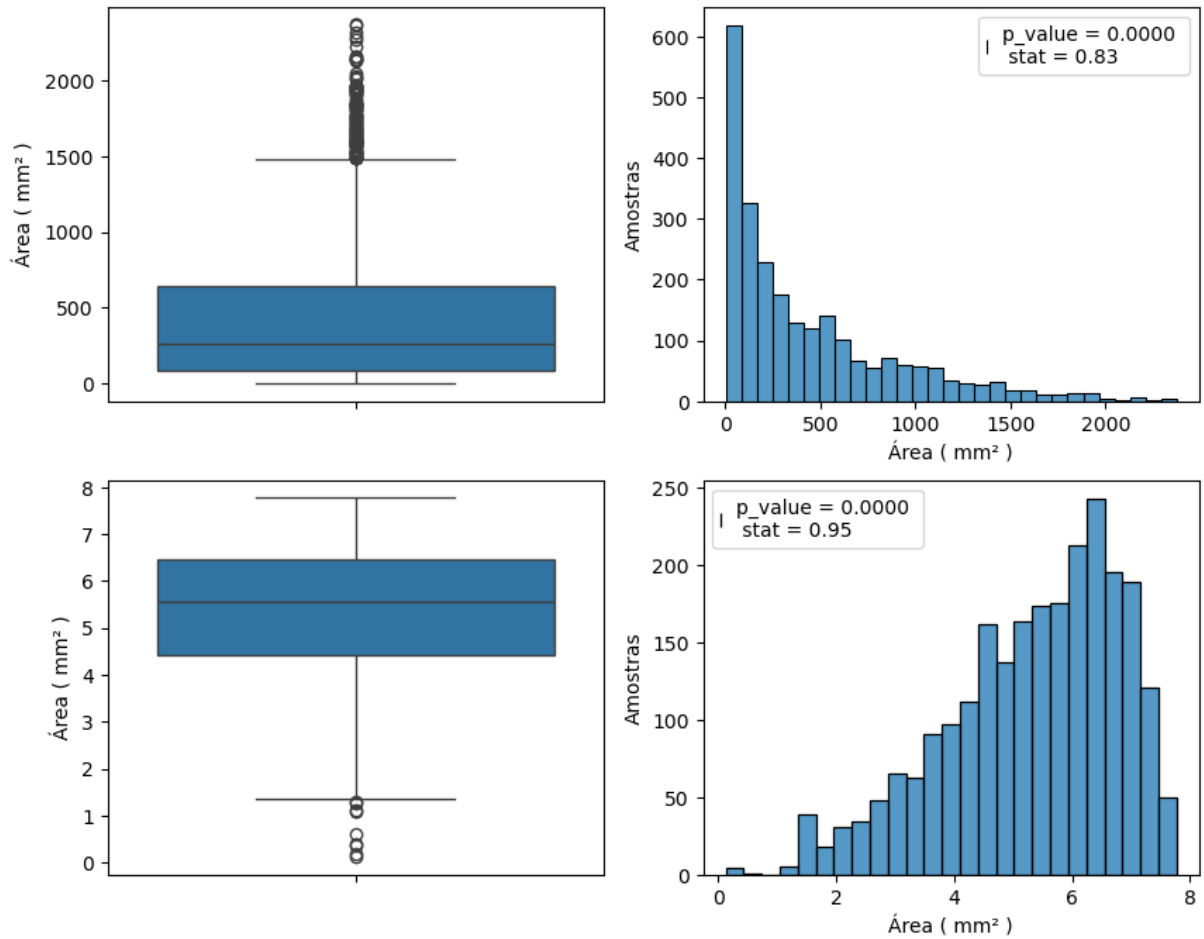
Fonte: elaborado pelo autor

6.1.2 Extração de características

Após a devida identificação, o algoritmo percorre as faces do sólido extraindo as características necessárias, nomeadamente: **área** e **geometria**. Após a extração ambas características também passo por um processamento. A área da superfície da área é normalizada e a geometria, devido sua natureza categórica, é codificada.

Para os dados da área foi adotada a mesma normalização logarítmica utilizada para o volume removido. A figura 60 apresenta os dados de **área**, em milímetros quadrados, antes e depois de aplicar a normalização.

Figura 60: Normalização dos dados - Área da face



Fonte: elaborado pelo autor

A codificação da geometria é feita utilizando a técnica de *one-hot encoding* e o resultado obtido, apresentado na tabela 5 Uma estatística da contagem de tipos de superfície encontrada em todo o conjunto de dados é apresentado na figura 61

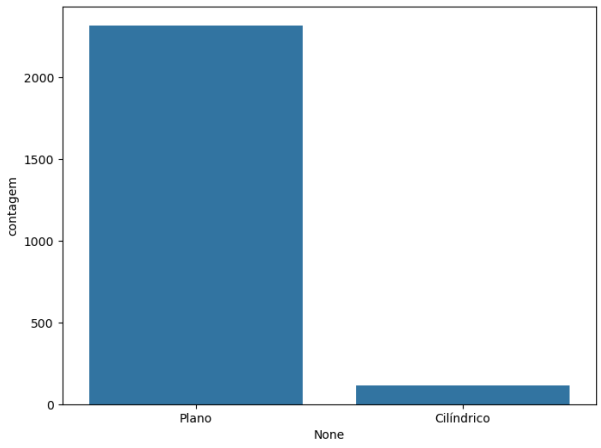
Tabela 5: Codificação da geometria da face

Geometria	Codificação
Plano	0
Cilindro	1

6.1.3 Estatísticas do dataset

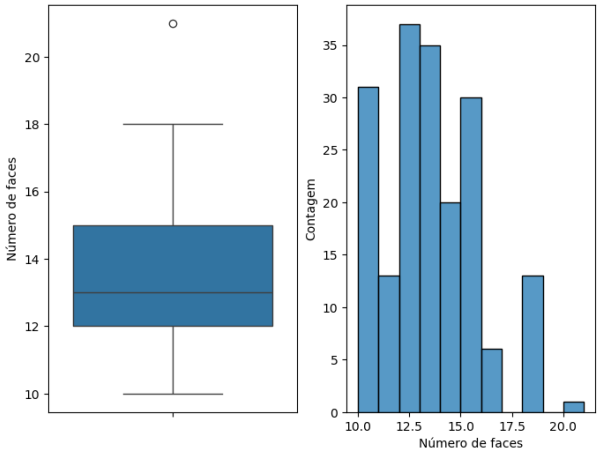
Dentro do conjunto de dados, podemos obter valiosas estatísticas, incluindo a contagem dos diferentes tipos de superfícies, como ilustrado na Figura 61, bem como a distribuição da quantidade de faces, conforme representado na Figura 62.

Figura 61: Contagem de tipos de superfície



Fonte: elaborado pelo autor

Figura 62: Contagem de faces por modelo



Fonte: elaborado pelo autor

6.1.4 Formatação dos dados

A representação de grafos é um componente essencial em muitas aplicações de aprendizado de grafos, incluindo redes neurais convolucionais em grafos (GCNs). O PyTorch Geometric (PyG) é uma extensão do PyTorch que oferece suporte específico para a manipulação eficiente de grafos em tarefas de aprendizado de máquina. Dessa forma a

informação gráfica deve ser estruturada em um formato compatível com essa ferramenta.

6.1.4.1 Conectividade

Uma estrutura chave no PyG é o `edge_index`, que é utilizado para armazenar informações de conectividade entre os nós do grafo. A representação padrão do `edge_index` em PyG envolve uma matriz bidimensional onde cada coluna representa uma aresta no grafo. Os pares de nós conectados por uma aresta são armazenados em duas linhas correspondentes à origem e ao destino da aresta. Por exemplo:

Figura 63: Representação das arestas

$$E_1 = [1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 4 \quad 4 \quad 4 \quad 4 \quad 5 \quad 5 \quad 5 \quad 5 \quad 7 \quad 7 \quad 8]$$

$$E_2 = [2 \quad 3 \quad 4 \quad 5 \quad 1 \quad 3 \quad 5 \quad 6 \quad 1 \quad 2 \quad 4 \quad 6 \quad 7 \quad 1 \quad 3 \quad 5 \quad 6 \quad 1 \quad 2 \quad 4 \quad 6 \quad 3 \quad 8 \quad 7]$$

Fonte: elaborado pelo autor

Sejam os arrays E_1 e E_2, cada par $(E_1[i], E_2[i])$ representa uma aresta do grafo, onde $E_1[i]$ é o nó de origem e $E_2[i]$ é o nó de destino para $0 \leq i < n$, e n é o número total de arestas. Para a primeira coluna, temos a aresta $(E_1[0], E_2[0]) = (1, 2)$, onde o nó de origem é 1 e o nó de destino é 2. A figura 65 ilustra o grafo correspondente a lista de arestas $E_1 e E_2$.

Figura 64: Modelo identificado

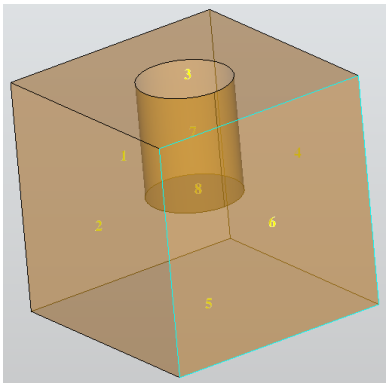
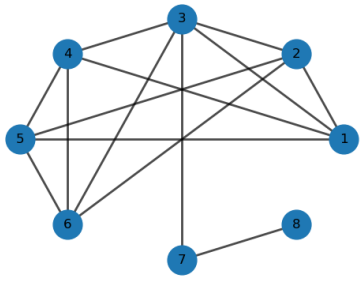


Figura 65: Grafo do modelo



Fonte: elaborado pelo autor

Esta representação pode parecer redundante à primeira vista, pois cada aresta é representada duas vezes (uma vez para cada direção). A redundância facilita operações simétricas, simplifica cálculos e contribui para a eficiência de algoritmos. Operações em

grafos não direcionados, por exemplo, podem ser realizadas sem a necessidade de transformações adicionais. Além disso, a estrutura bidimensional do `edge_index` simplifica a implementação de operações de agregação, permitindo cálculos eficientes sobre ambas as direções da aresta, concedendo simplicidade operacional em algoritmos relacionados a aprendizado de grafos.

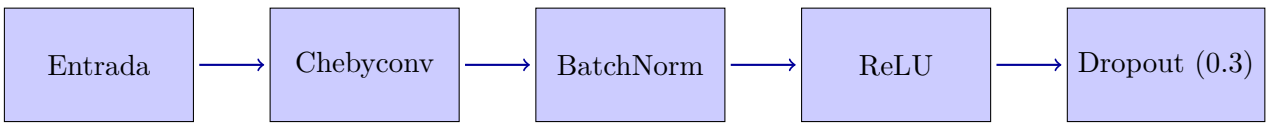
6.2 Arquitetura do modelo

Para a tarefa de criar uma representação do modelo CAD e estimar o tempo de processo, optou-se por utilizar uma rede neural gráfica com camadas de convolução do tipo *ChebConv*. A suposição subjacente é que, devido à sua análise espectral do grafo com filtros localizados, essa abordagem possa capturar características topológicas do modelo, como padrões de conectividade das faces, e criar uma representação significativa da peça como um todo. A ideia é realizar um tipo de reconhecimento de características usináveis primitivo, o que poderia contribuir para a relevância do modelo estatístico. Após a definição da estratégia de convolução, a maioria dos parâmetros arquiteturais foi ajustada de forma empírica, com base na análise do processo de treinamento e nos resultados obtidos.

6.2.1 Módulo de convolução

O módulo de convolução segue uma arquitetura padrão, composta por uma camada de convolução, uma camada de normalização em lote (Batch Normalization), uma função de ativação ReLU e uma camada de *Dropout* com uma taxa de 30%. O diagrama na Figura 66 ilustra essa composição de camadas.

Figura 66: Arquitetura do módulo de convolução



6.2.1.1 O parâmetro K

O parâmetro 'K' na convolução de Chebyshev determina quantos termos de Chebyshev são utilizados na operação de convolução. Isso afeta a quantidade de informações consideradas nos nós do grafo. À medida que o valor de 'K' aumenta, a camada ChebConv é capaz de capturar informações de nós mais distantes no grafo. No entanto, a escolha

específica de 'K' depende da natureza do problema e da estrutura do grafo e é geralmente otimizada empiricamente durante o treinamento para obter os melhores resultados.

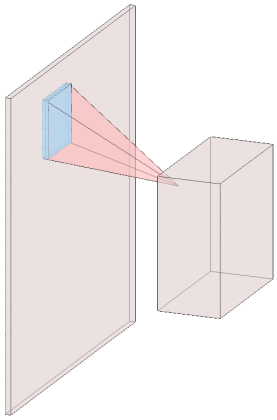
6.2.2 Agrupamento global

Na saída do módulo de convolução, os dados são encaminhados para uma camada de agrupamento global, onde passam por uma redução de dimensionalidade essencial para a execução da tarefa de regressão no módulo subsequente. A Figura 67 ilustra esse conceito. A estratégia escolhida é o agrupamento pelo valor máximo, conforme definido pela Equação 6.2.

$$Agrupamento_{medio}(R) = \frac{1}{|R|} \sum_{i \in R} x_i \tag{6.2}$$

Onde:
mean pooling(R) é o valor de saída após a aplicação da média de pooling na região R.
$|R|$ é o número de elementos na região R.
x_i são os valores dos elementos na região R.

Figura 67: Redução de dimensionalidade por agrupamento

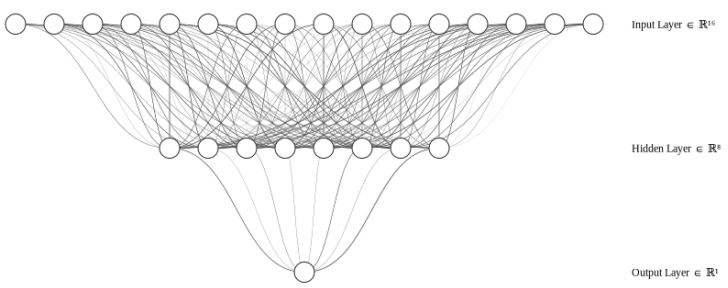


Fonte: elaborado pelo autor

6.2.3 Módulo de fusão e Módulo de regressão

O módulo de fusão recebe o volume removido como entrada antes de encaminhar os dados para uma série de camadas totalmente conectadas. A saída dessa camada é um tensor unidimensional que corresponde à previsão do tempo de processo. A regressão é realizada por meio de camadas totalmente conectadas, como ilustrado na figura 68

Figura 68: Camadas totalmente conectadas



Fonte: elaborado pelo autor

6.3 Definição dos parâmetros

Foi realizado um experimento abrangente para explorar uma ampla gama de parâmetros, visando identificar um ponto de partida para ajustes mais refinados. No total, foram avaliados 180 modelos, variando a quantidade de camadas convolucionais, o grau do polinômio de Chebyshev (K), o tamanho do lote de treinamento e a quantidade de unidades de processamento (também conhecidas como 'neurônios'). Algumas predefinições foram estabelecidas:

- As convoluções seguem o padrão estabelecido na seção 6.2.1, ou seja, cada camada de convolução é seguida por uma camada de normalização em lote, uma função de ativação ReLU e uma camada de *Dropout* com 30%.
- O modelo foi testado com 3, 2 ou 1 camadas de convolução, aumentando o número de canais em ordens de 2 para cada camada de convolução adicional, garantindo um aumento progressivo na dimensionalidade nessa etapa.
- Os valores possíveis para K são 8, 4, 2 ou 1. Essa decisão foi baseada na distribuição da quantidade de faces de cada grafo.
- Os valores do tamanho do lote de treinamento podem ser 8, 16 ou 32.
- o número de camadas totalmente conectadas bem como a quantidade de neurônios foi ajustada dependendo da saída do módulo de agrupamento, para que nunca houvesse um aumento de dimensionalidade nessa etapa.
- A taxa de aprendizado (*learning rate*) foi fixada em 0.001

7 RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados da etapa de definição e refinamento dos parâmetros. As condições de treinamento foram semelhantes em ambas as etapas. Foram utilizadas 200 peças, geradas sinteticamente durante o experimento, com uma divisão de dados de 80% para treinamento e 20% para validação e teste. O treinamento foi conduzido com a estratégia de *early stopping*, limitando-o a 20 iterações. Isso significa que, se não houvesse melhoria na saída do modelo dentro desse intervalo, o treinamento era interrompido. Além disso, **exclusivamente** para etapa de descoberta de parâmeros, foi adotada a estratégia de validação cruzada *K-fold* com 5 dobras. Os resultados registrados para essa análise são as médias das iterações. Após essa etapa, os resultados apresentados são as melhores configurações encontradas do modelo em consecutivas *single runs*. A função de perda selecionada foi o Erro Quadrático Médio (MSE) como é comum para esse tipo de tarefa. Todos os experimentos foram conduzidos em um computador Linux, utilizando as bibliotecas PyTorch [146] e PyG [149], com a GPU NVIDIA GeForce GTX 1650 e CUDA versão 12.2.

Para avaliar o desempenho do modelo, foram considerados os indicadores de perda média durante e erro absoluto médio durante a etapa de validação e o coeficiente de determinação R^2 no conjunto de dados de teste. Os resultados obtidos estão apresentados na Tabela 6. Com base na análise dos resultados, a arquitetura final da rede consiste em uma única camada de convolução, com ordem polinomial igual a 8, 64 canais e um tamanho de lote de treinamento de 32.

Após a definição do modelo, os resultados são apresentados da seguinte maneira. O primeiro gráfico (canto superior esquerdo) exibe a curva de perda durante as fases de treinamento e validação. O segundo gráfico (canto superior direito) mostra o Erro Médio Absoluto (MAE) ao longo das épocas para os dados de validação. O terceiro gráfico (canto inferior esquerdo) apresenta o coeficiente de determinação, indicando a qualidade das previsões no conjunto de testes. O quarto gráfico é um gráfico QQ que avalia a normalidade dos resíduos em relação a uma distribuição normal. O resultado do modelo

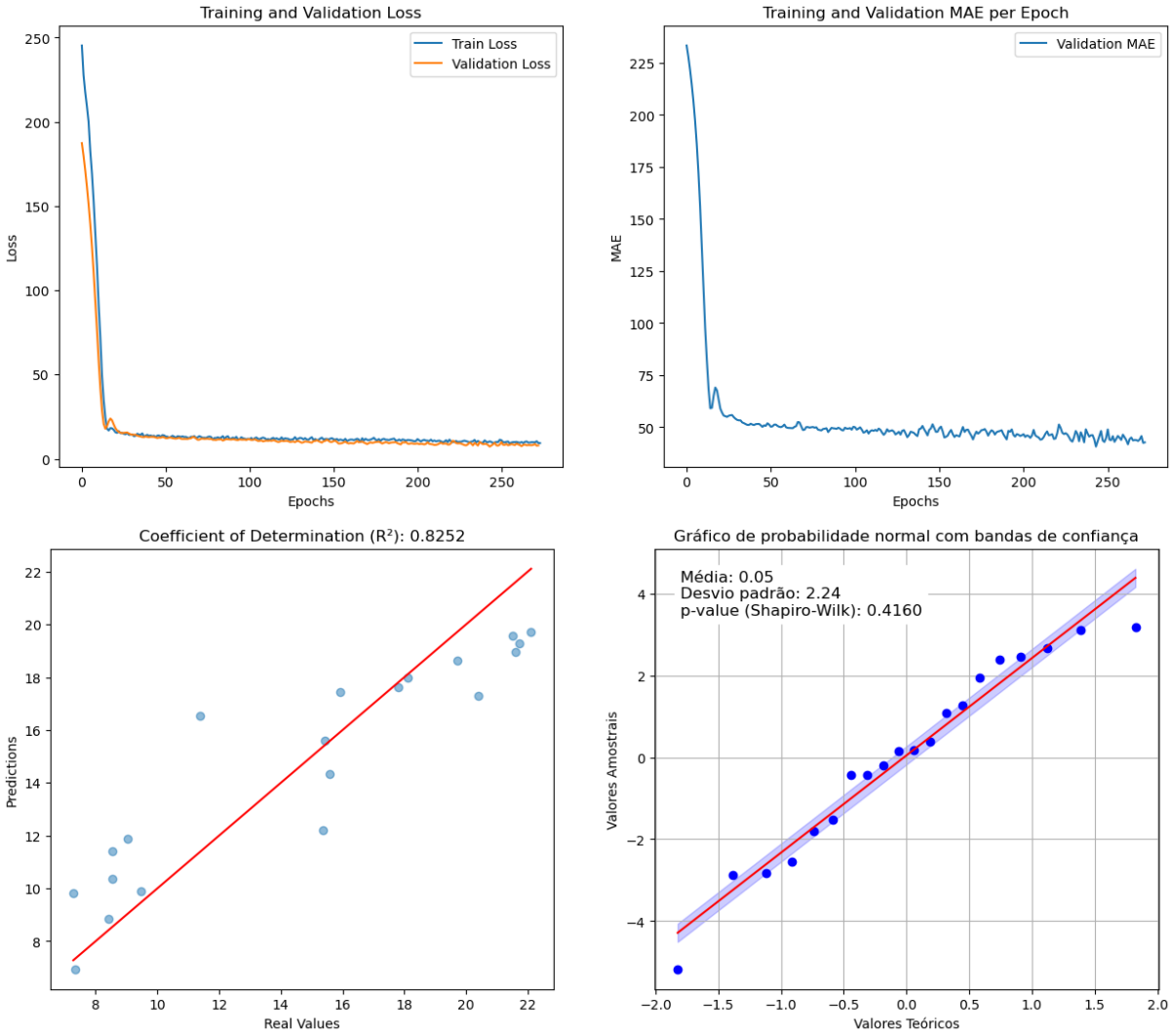
Tabela 6: Resultados parciais da etapa de descoberta de parâmetros

Perda média validação	Erro absoluto médio validação	Número de épocas médio	R2	P-value	Num_Units	K_Value	Num_Conv	Batch_Size
17.083	55.6599	81.7	0.2128	0.4261	32	4	1	32
21.6651	51.5018	48.9778	0.061	0.4306	16	2	3	8
16.7538	41.9576	30.9636	0.2701	0.4555	256	1	3	16
12.5907	84.7683	198.4	0.4455	0.3782	32	8	1	32
...	...	...	...	...	...	...	...	...
15.1603	41.7693	97.2667	0.3797	0.5224	128	8	1	8
16.338	48.5743	71.56	0.2422	0.4022	32	4	1	16
16.5766	46.365	29.58	0.2142	0.4503	64	1	3	32
20.9171	54.6564	37.12	0.1599	0.3977	16	4	2	16
18.6504	51.4008	35.04	0.0907	0.3864	64	4	3	16
...	...	...	...	...	...	...	...	...
19.8831	46.2725	34.85	0.142	0.4404	32	1	3	8
14.7401	40.3547	42.325	0.3002	0.473	256	2	2	16
15.9279	39.8621	60.2667	0.2809	0.4812	64	4	1	8

Fonte: elaborado pelo autor

final é apresentado na 69

Figura 69: Desempenho do modelo final



Fonte: elaborado pelo autor

A análise dos gráficos revela que, apesar de algum ruído, o modelo apresenta uma

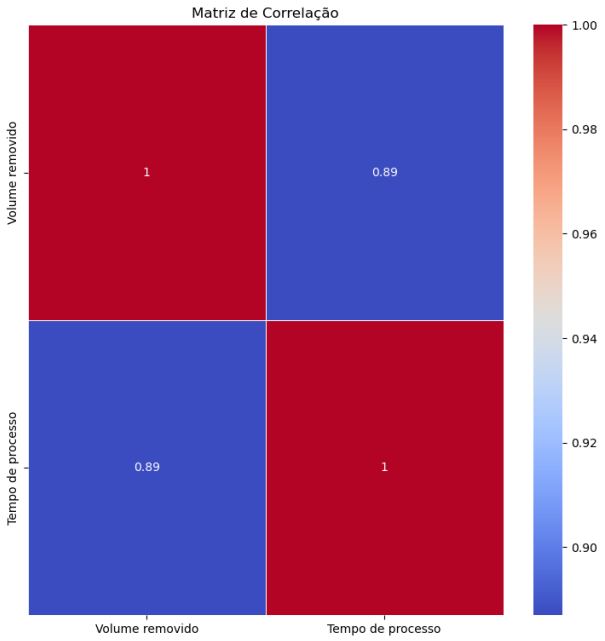
convergência clara. Tanto a perda na fase de validação quanto o Erro Médio Absoluto (MAE) se encontram em níveis aceitáveis. O coeficiente de determinação (R^2) oferece uma explicação razoável dos dados, sugerindo que o modelo conseguiu capturar *insights* significativos. Além disso, o gráfico de probabilidade de normalidade dos resíduos (QQ plot) não refuta a hipótese nula, com um valor de p acima de 0.05.

7.0.1 Discussão

A primeira coisa que chama atenção é o coeficiente de determinação, que embora indique uma explicação razoável dos dados, ainda está distante de um valor útil para que o modelo seja usado como preditor. Além disso, ao analisar o gráfico, é possível notar uma dispersão maior dos valores nas extremidades, o que **poderia** sugerir a presença de *outliers* no conjunto de dados. É importante ressaltar que a detecção de *outliers* neste domínio não é trivial, uma vez que a relação entre as variáveis independentes e o rótulo não é necessariamente conhecida (uma peça pode ter um alto tempo de processo por conta de uma estrutura topológica complexa).

Buscando entender um pouco melhor os dados, podemos analisar o coeficiente de *Pearson* que indica uma correlação **linear** muito forte do volume removido com a variável dependente, como mostra a figura 70

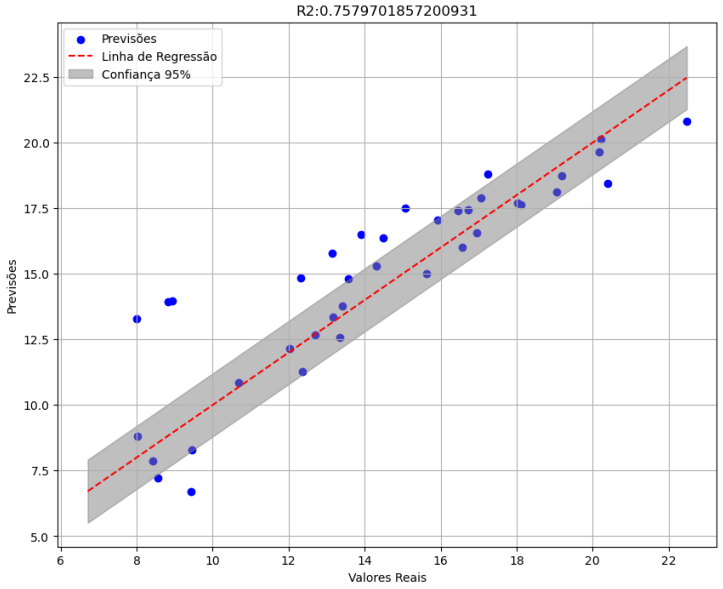
Figura 70: Coeficiente de correlação



Fonte: elaborado pelo autor

Isso nos leva a acreditar que existe uma relação linear muito bem definida entre o tempo de processo e o volume total removido dos sólidos 3D gerados sinteticamente. A partir disso, duas observações tornam-se plausíveis. Primeiro, é provável que uma simples regressão linear seja capaz de explicar os dados com alta precisão, considerando apenas o volume removido como variável independente, possivelmente superando o desempenho do modelo proposto. Segundo, ainda que menos óbvio, é razoável intuir que essa relação se torna mais linear à medida que a geometria da peça se torna mais simples. A Figura 71 apresenta o resultado da regressão linear entre o tempo de processo e o volume removido para o mesmo conjunto de dados.

Figura 71: Modelo de regressão linear para a estimativa do tempo de processo



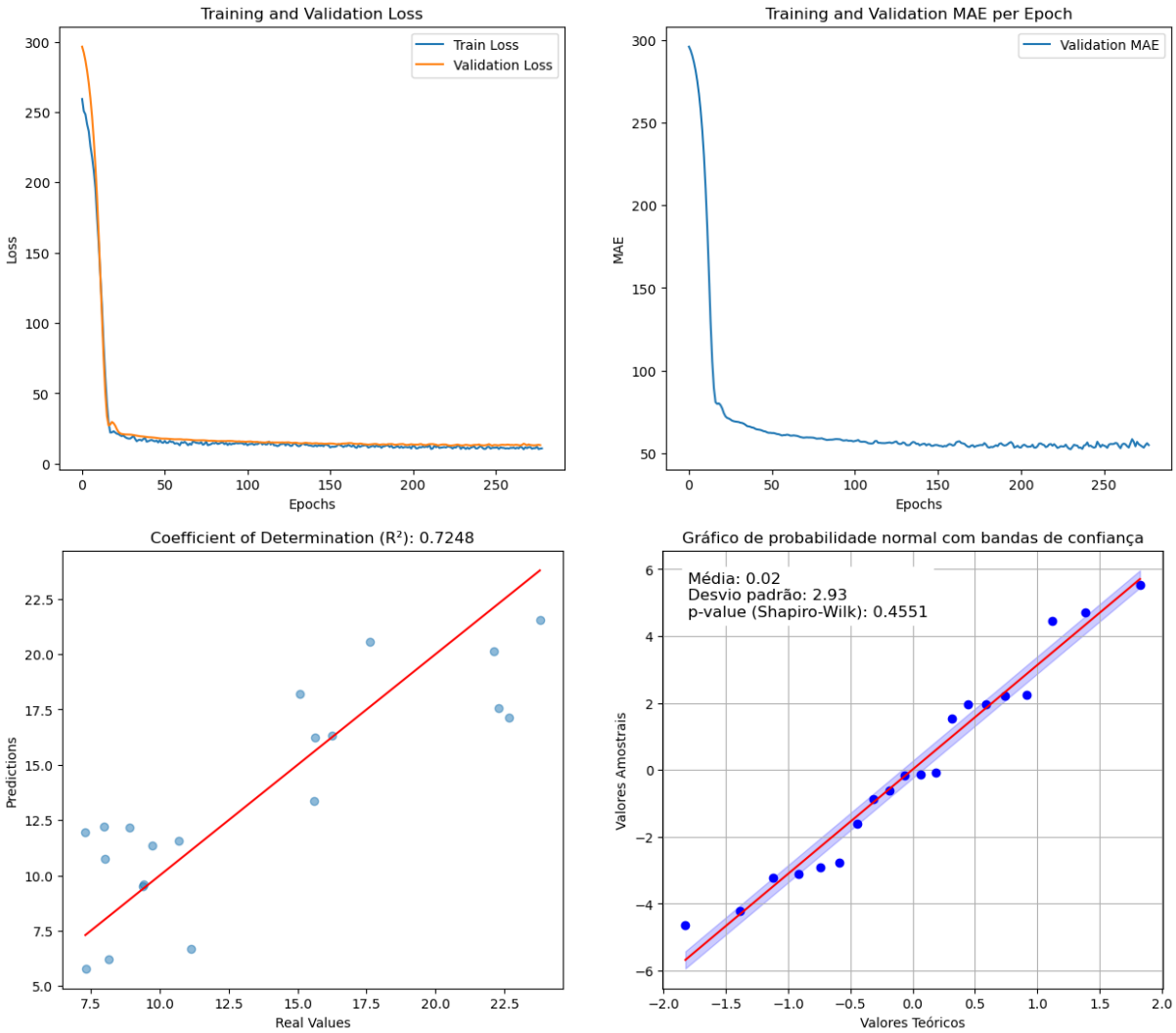
Fonte: elaborado pelo autor

Ao que tudo indica a regressão linear é capaz de explicar os dados quase tão bem quanto a rede neural gráfica. No entanto nesse momento propomos um exercício. Vamos tentar sustentar a hipótese de que, mesmo para peças de geometria simples, existe uma relação intrínseca com a geometria e topologia que a matriz de adjacência com atributos é capaz de codificar. E que além disso, que a rede neural convolucional seja capaz de reconhecer padrões complexos e criar abstrações a partir dessas informações, o que aconteceria se a informação de volume removido não tivesse disponível?

Primeiramente, direcionamos nossa atenção para a rede neural, considerando que há informações disponíveis para processamento. Neste momento, desabilitamos temporariamente o módulo de fusão e concentramos nossos esforços exclusivamente no grafo. A Figura 72 apresenta o resultado da regressão pela rede neural gráfica sem a informação

de volume removido.

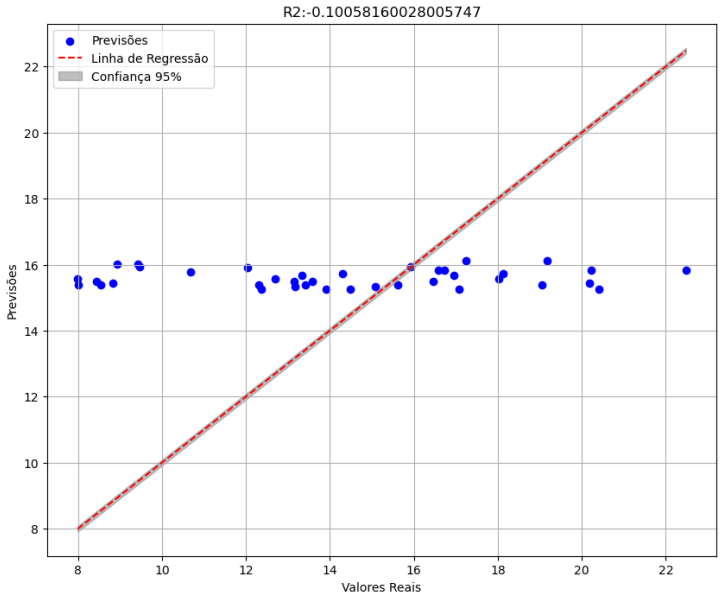
Figura 72: Desempenho do modelo final sem a informação de volume



Fonte: elaborado pelo autor

Como era esperado, observamos uma redução significativa na capacidade do modelo de explicar os dados. No entanto, ainda é importante reconhecer que ele está capturando algum padrão. Agora, para completar o exercício, vamos tentar alimentar a regressão linear com os dados que a rede neural gráfica está processando. Obviamente, não é possível alimentar diretamente o grafo em uma regressão linear, a intenção também não é criar uma representação topológica, já que as redes neurais gráficas já desempenham essa função. No entanto, podemos considerar a quantidade de faces como uma variável relevante. Essa informação parece estar intrinsecamente disponível para processamento pela rede neural. A regressão linear apenas com a quantidade de faces é apresentada na figura 73

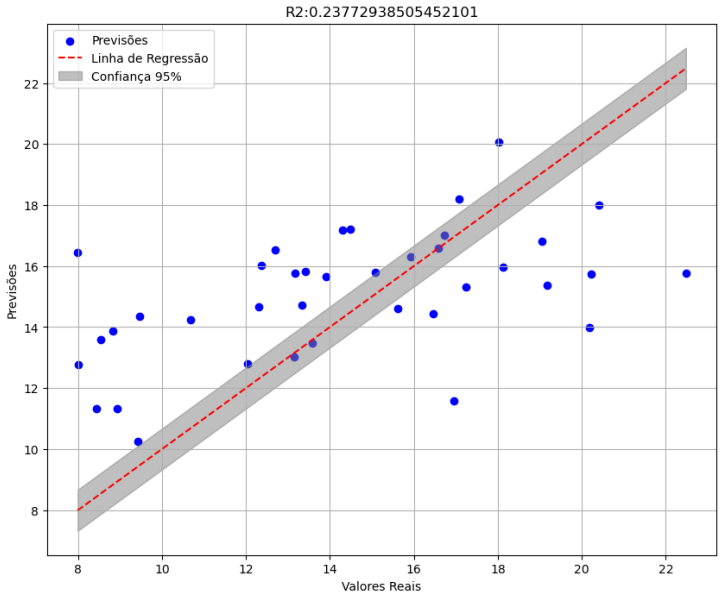
Figura 73: Regressão linear sobre o número de faces



Fonte: elaborado pelo autor

Isso não parece ajudar muito. Além disso, podemos considerar que a rede neural gráfica tem acesso aos valores das áreas de cada uma das faces. Sabemos que cada peça possui uma quantidade diferente de faces, e para manter o exemplo simples, podemos pensar em uma função de agregação. A Figura 74 ilustra a regressão linear operando com a quantidade de faces e a soma total das áreas.

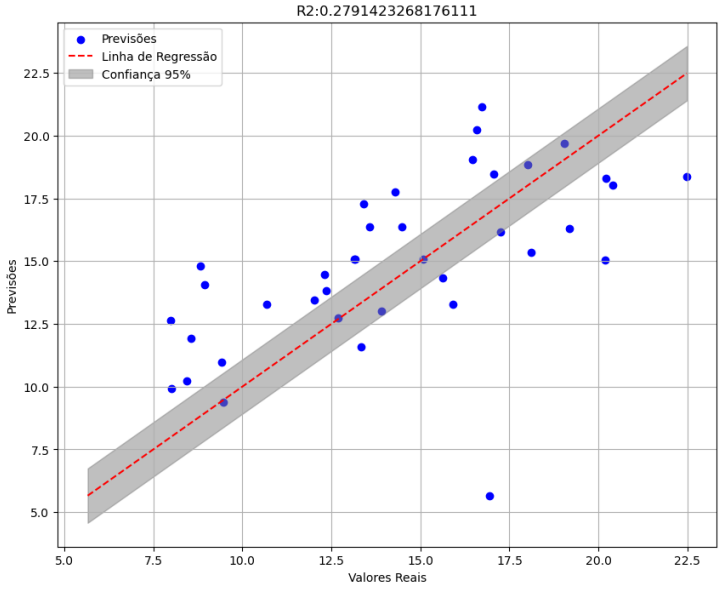
Figura 74: Regressão linear sobre o número de faces e área



Fonte: elaborado pelo autor

Para finalizar, vamos adicionar uma última informação que possivelmente está sendo capturada pela rede neural. No último estágio da convolução, antes do módulo de fusão (quando ainda usávamos o volume removido), acrescentamos uma camada de agrupamento global médio. É razoável imaginar que a média das áreas de alguma forma está sendo processada pela rede neural gráfica. A Figura 75 apresenta o resultado da regressão linear com as três variáveis discutidas.

Figura 75: Regressão linear sobre o número de faces, soma das área e média das áreas



Fonte: elaborado pelo autor

Chegamos ao final do exercício com uma melhora significativa, embora ainda estejamos distantes do resultado obtido pela rede neural gráfica. Este exercício foi conduzido com o objetivo de explorar e hipotetizar sobre o potencial adicional que a rede neural gráfica é capaz de capturar em comparação algo que seja modelável sobre o ponto de vista Euclidiano.

8 CONCLUSÃO

Neste trabalho, foi apresentado um modelo de rede neural gráfica convolucional baseado em abordagens espectrais e polinômios de Chebyshev para o processamento de modelos CAD. O principal objetivo deste arcabouço é estimar o tempo de usinagem. Ele inclui um módulo para a geração de modelos sintéticos, uma estratégia robusta de codificação de faces usando matrizes de adjacência com atributos e um laboratório dedicado à descoberta de parâmetros otimizados para redes neurais gráficas. Isso proporciona um ambiente de experimentação fértil para algoritmos dessa natureza.

Embora os resultados objetivos obtidos não tenham demonstrado robustez suficiente para a aplicação prática da ferramenta, ficou claro o potencial das redes neurais gráficas no processamento de modelos 3D. As oportunidades de melhoria do modelo podem estar relacionadas tanto à arquitetura quanto à qualidade e quantidade dos dados de treinamento.

Embora o objetivo principal fosse a estimativa do tempo de processo, é válido considerar que es‘sa abordagem poderia ser aplicada a outras finalidades. Se confirmada a hipótese de que a rede neural gráfica é capaz de capturar *insights* sobre a topologia da peça, diversas aplicações poderiam se beneficiar dessa abordagem, como a análise de manufaturabilidade, por exemplo.

Além disso o estudo contou com uma revisão tanto dos conceitos fundamentais quanto dos avanços recentes de metodologias de aprendizado de máquina que possuem aplicabilidade na indústria de manufatura. Especificamente nas áreas de reconhecimento automático de características e na estimativa de custos de usinagem foi abordado temas como formatos de representação, modelos base, algoritmos dedicados e estratégias integrativas como as redes neurais *multi-task*.

8.1 Trabalhos futuros

Seguindo a lógica do cálculo do tempo de processo com base em características, uma estratégia interessante seria incorporar um módulo de reconhecimento de características antes da entrada no modelo. Codificar o rótulo das características juntamente com o grafo de adjacência provavelmente resultaria em uma representação bastante robusta, com o potencial de exponenciar a precisão do modelo.

Do ponto de vista de melhorias incrementais, seria interessante explorar a codificação de informações dos vértices para que também pudessem ser processadas pela rede neural, como, por exemplo, características relacionadas à convexidade. Isso provavelmente resultaria na necessidade de investigar outras arquiteturas, possivelmente abordagens espaciais, que lidam de maneira mais eficaz com informações localizadas.

Além disso, estratégias de funções de perda personalizadas têm sido bem-sucedidas em tarefas de classificação, mas há uma escassez de alternativas para tarefas de regressão. Investigar o desenvolvimento de funções de perda adaptadas especificamente para a tarefa de regressão também pode ser um caminho promissor para futuras melhorias no modelo.

Como ponto de melhoria mais tático, o módulo de geração de modelos sintéticos deveria ser aprimorado para incorporar validações de manufaturabilidade.

REFERÊNCIAS

[1] LIU, C. et al. A feature-based method for NC machining time estimation. *Robot. Comput. Integr. Manuf.*, v. 29, n. 4, p. 8–14, ago. 2013. ISSN 0736-5845.

[2] NIAZI, A. et al. Product cost estimation: Technique classification and methodology review. *J. Manuf. Sci. Eng.*, American Society of Mechanical Engineers Digital Collection, v. 128, n. 2, p. 563–575, set. 2005.

[3] SOHLENIUS, G. Concurrent engineering. *CIRP Annals*, v. 41, n. 2, p. 645–655, 1992. ISSN 0007-8506. Disponível em: <https://www.sciencedirect.com/science/article/pii/S000785060763251X>.

[4] SHEHAB, E. M.; ABDALLA, H. S. Manufacturing cost modelling for concurrent product development. *Robot. Comput. Integr. Manuf.*, v. 17, n. 4, p. 341–353, ago. 2001.

[5] OU-YANG, C.; LIN, T. S. Developing an integrated framework for feature-based early manufacturing cost estimation. *Int. J. Adv. Manuf. Technol.*, Springer Nature, v. 13, n. 9, p. 618–629, set. 1997.

[6] JUNG, J.-Y. Manufacturing cost estimation for machined parts based on manufacturing features. *Journal of intelligent manufacturing*, Springer, v. 13, p. 227–238, 2002.

[7] SHI, Y. et al. A critical review of feature recognition techniques. *Computer-Aided Design and Applications*, v. 17, p. 861–899, 01 2020.

[8] JOSHI, S.; CHANG, T. Graph-based heuristics for recognition of machined features from a 3d solid model. *Computer-Aided Design*, v. 20, n. 2, p. 58–66, 1988. ISSN 0010-4485. Disponível em: <https://www.sciencedirect.com/science/article/pii/0010448588900504>.

[9] LAAKKO, T.; MÄNTYLÄ, M. Feature modelling by incremental feature recognition. *Comput. Aided Des. Appl.*, v. 25, n. 8, p. 479–492, ago. 1993. ISSN 1686-4360, 0010-4485.

[10] GAO, S.; SHAH, J. J. Automatic recognition of interacting machining features based on minimal condition subgraph. *Comput. Aided Des.*, Elsevier BV, v. 30, n. 9, p. 727–739, ago. 1998. ISSN 0010-4485, 1879-2685.

[11] MATURANA, D.; SCHERER, S. VoxNet: A 3D convolutional neural network for real-time object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.]: IEEE, 2015. p. 922–928. ISBN 9781479999941, 9781479999934.

[12] QI, C. R. et al. PointNet: Deep learning on point sets for 3D classification and segmentation. dez. 2016.

[13] HANOCKA, R. et al. MeshCNN: a network with an edge. *ACM Trans. Graph.*, n. 4, p. 1–12, jul. 2019.

[14] LAMBOURNE, J. G. et al. BRepNet: A topological message passing system for solid models. *IEEE Xplore Digit. Libr.*, 2021. ISSN 2473-2001.

[15] BRUNA, J. et al. Spectral networks and locally connected networks on graphs. dez. 2013.

[16] DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. jun. 2016.

[17] CAO, W. et al. Graph representation of 3D CAD models for machining feature recognition with deep learning. *ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers Digital Collection, p. V11AT11A003, nov. 2020.

[18] VERMA, N.; BOYER, E.; VERBEEK, J. FeaStNet: Feature-steered graph convolutions for 3D shape analysis. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE, 2018. ISBN 9781538664209.

[19] ARMILLOTTA, A. On the role of complexity in machining time estimation. *J. Intell. Manuf.*, v. 32, n. 8, p. 2281–2299, dez. 2021. ISSN 0956-5515, 1572-8145.

[20] ATIA, M. R. A.; MOKHTAR, M.; KHALIL, J. An ANN parametric approach for the estimation of total production operation time. *Ain Shams Engineering Journal*, v. 13, n. 2, p. 101579, mar. 2022. ISSN 2090-4479.

[21] BOOTHROYD, G. *Fundamentals of Metal Machining and Machine Tools, Third Edition*. [S.l.]: CRC Press, 1988. ISBN 9780824778521.

[22] BORING Tool. [S.l.]: Florida Center for Instructional Technology (FCIT). `https://etc.usf.edu/clipart/72700/72745/72745_boring_tool.htm`. Accessed: 2024-1-26.

[23] GROOVER, M. P. *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems*. [S.l.]: John Wiley & Sons, 2007.

[24] COROMANT, S. *Training Handbook: Metal Cutting Technology*. Sandviken, 2017.

[25] RAO, P. N. *Manufacturing Technology: Metal Cutting and Machine Tools*. [S.l.]: Tata McGraw-Hill Education, 2013.

[26] BABIĆ, B. R.; NEŠIĆ, N.; MILJKOVIĆ, Z. Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems. *Artif. Intell. Eng. Des. Anal. Manuf.*, Cambridge University Press (CUP), v. 25, n. 3, p. 289–304, ago. 2011. ISSN 0890-0604, 1469-1760.

[27] FU, M. et al. An approach to identify design and manufacturing features from a data exchanged part model. *Computer-Aided Design*, v. 35, n. 11, p. 979–993, 2003. ISSN 0010-4485. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010448502001604>.

[28] NASR, E. S. A.; KAMRANI, A. K. A new methodology for extracting manufacturing features from cad system. *Computers & Industrial Engineering*, Elsevier, 2006.

[29] NAGARAJAN, S.; REDDY, N. V. Step-based automatic system for recognising design and manufacturing features. *International Journal of Production Research*, Taylor & Francis, v. 48, n. 1, p. 117–144, 2010.

[30] NARANG, R. An application-independent methodology of feature recognition with explicit representation of feature interaction. *Journal of Intelligent Manufacturing*, v. 7, n. 6, p. 479–486, 1996.

[31] DEWHURST, P.; BOOTHROYD, G. Early cost estimating in product design. *Journal of Manufacturing Systems*, v. 7, n. 3, p. 183–191, jan. 1988. ISSN 0278-6125.

[32] PIMAPUNSRI, K. Conception intégrée de meubles réalisés en panneaux de fibres ou de particules. 01 2007.

[33] GARCÍA-CRESPO, Á. et al. A review of conventional and knowledge based systems for machining price quotation. *J. Intell. Manuf.*, v. 22, n. 6, p. 823–841, dez. 2011.

[34] CASTELAIN, P. D.; J. Cost estimation during design step: parametric method versus case based reasoning method. *Int J Adv Manuf Technol*, 1999.

[35] LOPEZ-CUADRADO, J. L. et al. Development of a web-based quotation expert system for machined parts. *Int. J. Comput. Appl. Technol.*, Inderscience Publishers, v. 37, n. 2, p. 87–100, jan. 2010.

[36] GAYRETLI, A.; ABDALLA, H. S. An object-oriented constraints-based system for concurrent product development. *Robot. Comput. Integr. Manuf.*, v. 15, n. 2, p. 133–144, abr. 1999.

[37] SHEHAB, E. M.; ABDALLA, H. S. A design to cost system for innovative product development. *Proc. Inst. Mech. Eng. Pt. B: J. Eng. Manuf.*, SAGE Publications, v. 216, n. 7, p. 999–1019, jul. 2002.

[38] AREZOO, B.; RIDGWAY, K.; AL-AHMARI, A. M. A. Selection of cutting tools and conditions of machining operations using an expert system. *Comput. Ind.*, v. 42, n. 1, p. 43–58, jun. 2000.

[39] BIDANDA, B.; KADIDAL, M.; BILLO, R. E. Development of an intelligent castability and cost estimation system. *Int. J. Prod. Res.*, Taylor & Francis, v. 36, n. 2, p. 547–568, fev. 1998. ISSN 0020-7543.

[40] COS, J. de et al. Rapid cost estimation of metallic components for the aerospace industry. *Int. J. Prod. Econ.*, v. 112, n. 1, p. 470–482, mar. 2008. ISSN 0925-5273.

[41] LOYER, J.-L. et al. Comparison of machine learning methods applied to the estimation of manufacturing cost of jet engine components. *Int. J. Prod. Econ.*, v. 178, p. 109–119, ago. 2016. ISSN 0925-5273.

[42] ADEROBAL, A. A generalised cost-estimation model for job shops. *ELSEVIER Int. J. Production Economics*, 1997.

[43] BEN-ARIEH, D.; QIAN, L. Activity-based cost management for design and development stage. *Int. J. Prod. Econ.*, v. 83, n. 2, p. 169–183, fev. 2003. ISSN 0925-5273.

[44] SFANTSIKOPOULOS, M. M.; DIPLARIS, S. C.; PAPAZOGLOU, P. N. Concurrent dimensioning for accuracy and cost. *Int. J. Adv. Manuf. Technol.*, Springer, v. 10, p. 263–268, 1995. ISSN 0268-3768.

[45] SON, Y. K. A cost estimation model for advanced manufacturing systems. *Int. J. Prod. Res.*, Taylor & Francis, v. 29, n. 3, p. 441–452, mar. 1991. ISSN 0020-7543.

[46] SEQUEIRA, S.; LOPES, E. Simple method proposal for cost estimation from work breakdown structure. *Procedia Comput. Sci.*, v. 64, p. 537–544, jan. 2015. ISSN 1877-0509.

[47] STOCKTON, D.; WANG, Q. Developing cost models by advanced modelling technology. *Proc. Inst. Mech. Eng. Pt. B: J. Eng. Manuf.*, IMECHE, v. 218, n. 2, p. 213–224, fev. 2004. ISSN 0954-4054.

[48] BOOTHROYD, G.; REYNOLDS, C. Approximate cost estimates for typical turned parts. *Journal of Manufacturing Systems*, v. 8, n. 3, p. 185–193, jan. 1989. ISSN 0278-6125.

[49] MUTER, S. *Cost comparison of alternate designs: an information based model*. Tese (Doutorado) — Massachusetts Institute of Technology, 1993.

[50] BOOTHROYD, G.; DEWHURST, P.; KNIGHT, W. A. *Product design for manufacture and assembly*. [S.l.]: CRC press, 2010.

[51] SARIC, T. et al. Estimation of machining time for CNC manufacturing using neural computing. *International Journal of*, ijsimm.com, 2016.

[52] Robert Creese (Author)_ M. Adithan (Author) -. *Estimating and Costing for the Metal Manufacturing Industries.pdf*. [S.l.]: CRC Press.

[53] OTHMANI, R.; BOUZID, W.; HBAIEB, M. Machining time in rough milling. *Materials Technology*, Taylor & Francis, v. 23, n. 3, p. 169–173, set. 2008. ISSN 1066-7857.

[54] MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, v. 5, n. 4, p. 115–133, dez. 1943. ISSN 0007-4985, 1522-9602.

[55] HEBB, D. O. *The Organization of Behavior: A Neuropsychological Theory*. [S.l.]: Psychology Press, 2005. ISBN 9781135631918.

[56] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, v. 65, n. 6, p. 386–408, nov. 1958. ISSN 0033-295X.

[57] BOS, S. Modelling the human brain: Feed-forward neural networks. 2018.

[58] NIELSEN, M. *Neural networks and deep learning*. [S.l.]: ise.ncsu.edu, 2015. `https://www.ise.ncsu.edu/fuzzy-neural/wp-content/uploads/sites/9/2022/08/neuralnetworksanddeeplearning.pdf`. Accessed: 2023-12-18.

[59] Wikipedia contributors. *Heaviside step function*. 2023. https://en.wikipedia.org/w/index.php?title=Heaviside_step_function&oldid=1182939366. Accessed: 2023-11-1.

[60] CICALA, F. *Perceptrons, Logical Functions, and the XOR problem*. 2018. https://towardsdatascience.com/perceptrons-logical-functions-and-the-xor-problem-37ca5025790a. Accessed: 2024-1-5.

[61] AMINI, A. *MIT Introduction to Deep Learning*. [S.l.]: Youtube, 2023.

[62] LECUN, Y. A. et al. Efficient BackProp. In: MONTAVON, G.; ORR, G. B.; MÜLLER, K.-R. (Ed.). *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 9–48. ISBN 9783642352898.

[63] KELLEY, H. J. Gradient theory of optimal flight paths. *ARS Journal*, American Institute of Aeronautics and Astronautics, v. 30, n. 10, p. 947–954, out. 1960.

[64] MASTER, A. *Please Stop Drawing Neural Networks Wrong*. 2023. https://tinyurl.com/bdezupbx. Accessed: 2024-1-10.

[65] CHAI, T.; DRAXLER, R. R. Root mean square error (RMSE) or mean absolute error (MAE)? *Geosci. Model Dev. Discuss.*, Copernicus GmbH, v. 7, n. 1, p. 1525–1534, fev. 2014. ISSN 1991-962X.

[66] Wikipedia contributors. *Partial derivative*. 2024. https://en.wikipedia.org/w/index.php?title=Partial_derivative&oldid=1194106824. Accessed: 2023-11-1.

[67] Wikipedia contributors. *Chain rule*. 2024. https://en.wikipedia.org/w/index.php?title=Chain_rule&oldid=1215462697. Accessed: 2023-11-1.

[68] Wikipedia contributors. *Gradient*. 2024. https://en.wikipedia.org/w/index.php?title=Gradient&oldid=1193102905. Accessed: 2023-11-1.

[69] UNDERSTANDING Pythagorean Distance and the Gradient. https://tinyurl.com/4tep54wc. Accessed: 2023-11-1.

[70] POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, v. 4, n. 5, p. 1–17, jan. 1964. ISSN 0041-5553.

[71] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. dez. 2014.

[72] GUPTA, A. *A Comprehensive Guide on Optimizers in Deep Learning*. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/. Accessed: 2024-1-13.

[73] SRIVASTAVA, N. et al. *Dropout: A simple way to prevent neural networks from overfitting*. [S.l.]: jmlr.org, 2014. https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer,. Accessed: 2023-12-18.

[74] FUKUSHIMA, K. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, v. 36, n. 4, p. 193–202, 1980. ISSN 0340-1200.

[75] LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, MIT Press, v. 1, n. 4, p. 541–551, dez. 1989. ISSN 0899-7667.

[76] DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. mar. 2016.

[77] HE, K. et al. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2016. ISBN 9781467388511.

[78] HUANG, G. et al. Densely connected convolutional networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2017. ISBN 9781538604571.

[79] LI, G. et al. DeepGCNs: Making GCNs go as deep as CNNs. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 45, n. 6, p. 6923–6939, jun. 2021. ISSN 0162-8828.

[80] IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. fev. 2015.

[81] AHMED, E. et al. A survey on deep learning advances on different 3D data representations. ago. 2018.

[82] WANG, Y. et al. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 5, p. 1–12, out. 2019. ISSN 0730-0301.

[83] SUCH, F. P. et al. Robust spatial filtering with graph convolutional neural networks. *IEEE J. Sel. Top. Signal Process.*, IEEE, v. 11, n. 6, p. 884–896, set. 2017. ISSN 1932-4553, 1941-0484.

[84] RAHMAN, M. S. *Basic Graph Theory*. [S.l.]: Springer International Publishing, 2017.

[85] SPIELMAN, D. Spectral graph theory. In: *Spectral Graph Theory*. [S.l.]: cs.yale.edu, 2012.

[86] LIN, H.; SUN, Y. EleGNN: Electrical-Model-Guided graph neural networks for power distribution system state estimation. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. [S.l.]: IEEE, 2022. p. 5292–5298. ISBN 9781665435406, 9781665435413.

[87] ZHANG, Q. et al. Kernel-Weighted graph convolutional network: A deep learning approach for traffic forecasting. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. [S.l.]: IEEE, 2018. p. 1018–1023. ISBN 9781538637883, 9781538637876. ISSN 1051-4651.

[88] FRANCESCO, A. G. D. et al. GATSY: Graph attention network for music artist similarity. nov. 2023.

[89] GORI, M.; MONFARDINI, G.; SCARSELLI, F. A new model for learning in graph domains. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* [S.l.]: IEEE, 2005. v. 2, p. 729–734 vol. 2. ISBN 9780780390485. ISSN 2161-4393, 2161-4407.

[90] SCARSELLI, F. et al. The graph neural network model. *IEEE Trans. Neural Netw.*, v. 20, n. 1, p. 61–80, jan. 2009. ISSN 1045-9227, 1941-0093.

[91] Wikipedia contributors. *Teorema do ponto fixo de Banach*. `https://pt.wikipedia.org/w/index.php?title=Teorema_do_ponto_fixo_de_Banach&oldid=66106792`. Accessed: 2023-11-1.

[92] Wikipedia contributors. *Teorema da contração*. `https://pt.wikipedia.org/w/index.php?title=Teorema_da_contra%C3%A7%C3%A3o&oldid=57331704`. Accessed: 2023-11-1.

[93] WU, Z. et al. A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst*, v. 32, n. 1, p. 4–24, jan. 2021. ISSN 1045-9227, 2162-237X.

[94] GALLICCHIO, C.; MICHELI, A. Graph echo state networks. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2010. p. 1–8. ISBN 9781424469185, 9781424469161. ISSN 2161-4407, 2161-4393.

[95] LI, Y. et al. Gated graph sequence neural networks. nov. 2015.

[96] DAI, H. et al. Learning Steady-States of iterative algorithms over graphs. In: DY, J.; KRAUSE, A. (Ed.). *Proceedings of the 35th International Conference on Machine Learning*. [S.l.]: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 1106–1114.

[97] SANCHEZ-LENGELING, B. et al. A gentle introduction to graph neural networks. *Distill*, Distill Working Group, v. 6, n. 8, ago. 2021. ISSN 2476-0757.

[98] WOO, Y.; SAKURAI, H. Recognition of maximal features by volume decomposition. *Comput. Aided Des.*, Elsevier BV, v. 34, n. 3, p. 195–207, mar. 2002. ISSN 0010-4485, 1879-2685.

[99] MICHELI, A. Neural network for graphs: a contextual constructive approach. *IEEE Trans. Neural Netw.*, v. 20, n. 3, p. 498–511, mar. 2009. ISSN 1045-9227, 1941-0093.

[100] ATWOOD, J.; TOWSLEY, D. Diffusion-convolutional neural networks. In: LEE, D. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016. v. 29. Disponível em: <`https://proceedings.neurips.cc/paper_files/paper/2016/file/390e982518a50e280d8e2b535462ec1f-Paper.pdf`>.

[101] GILMER, J. et al. Neural message passing for quantum chemistry. abr. 2017.

[102] COLLIGAN, A. R. et al. Hierarchical CADNet: Learning from B-Reps for machining feature recognition. *Comput. Aided Des. Appl.*, v. 147, p. 103226, jun. 2022. ISSN 1686-4360.

[103] NIEPERT, M.; AHMED, M.; KUTZKOV, K. Learning convolutional neural networks for graphs. *arXiv [cs.LG]*, maio 2016.

[104] CHUNG, F. R. K. Eigenvalues and the laplacian of a graph. In: *Spectral Graph Theory*. [S.l.]: American Mathematical Soc., 1997. cap. 1. ISBN 9780821803158.

[105] Wikipedia contributors. *Teorema da convolução*. `https://pt.wikipedia.org/w/index.php?title=Teorema_da_convolu%C3%A7%C3%A3o&oldid=34801251`. Accessed: 2023-11-1.

[106] Wikipedia contributors. *Principal component analysis*. 2023. `https://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=1192620451`. Accessed: 2023-11-1.

[107] KIPF, T. N.; WELLING, M. Semi-Supervised classification with graph convolutional networks. set. 2016.

[108] Wikipedia contributors. *Chebyshev polynomials*. 2023. `https://en.wikipedia.org/w/index.php?title=Chebyshev_polynomials&oldid=1190135028`. Accessed: 2023-11-1.

[109] ZHANG, H. et al. A novel method based on a convolutional graph neural network for manufacturing cost estimation. *Journal of Manufacturing Systems*, v. 65, p. 837–852, out. 2022. ISSN 0278-6125.

[110] RAI, R. et al. Machine learning in manufacturing and industry 4.0 applications. *Int. J. Prod. Res.*, Taylor & Francis, v. 59, n. 16, p. 4773–4778, ago. 2021. ISSN 0020-7543.

[111] WONG, T.; LAM, S. Automatic recognition of machining features from computer aided design part models. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, v. 214, n. 6, p. 515–520, 2000.

[112] MARCHETTA, M.; FORRADELLAS, R. An artificial intelligence planning approach to manufacturing feature recognition. *Computer-Aided Design*, v. 42, n. 3, p. 248–256, 2010.

[113] YILDIZ, A. et al. Integrated optimal topology design and shape optimization using neural networks. *Structural and Multidisciplinary Optimization*, v. 25, n. 4, p. 251–260, 2003.

[114] SOMASHEKAR, S. a.; MICHAEL, W. u. An overview of automatic feature recognition techniques for computer-aided process planning. *ELSEVIER Computers in Industry*, 1995.

[115] DONALDSON, I. A.; CORNEY, J. R. Rule-based feature recognition for 2.5d machined components. *Int. J. Comput. Integr. Manuf.*, Taylor & Francis, v. 6, n. 1-2, p. 51–64, jan. 1993. ISSN 0951-192X.

[116] VOSNIAKOS, G. C.; DAVIES, B. J. A shape feature recognition framework and its application to holes in prismatic parts. *Int. J. Adv. Manuf. Technol.*, v. 8, n. 6, p. 345–351, nov. 1993. ISSN 0268-3768, 1433-3015.

[117] BHANDARKAR, M. P.; NAGI, R. STEP-based feature extraction from STEP geometry for agile manufacturing. *Comput. Ind.*, Elsevier BV, v. 41, n. 1, p. 3–24, jan. 2000. ISSN 0166-3615, 1872-6194.

[118] NING, F. et al. Manufacturing cost estimation based on the machining process and deep-learning method. *Journal of Manufacturing Systems*, v. 56, p. 11–22, jul. 2020. ISSN 0278-6125.

[119] DONG, X.; WOZNY, M. J. A METHOD FOR GENERATING VOLUMETRIC FEATURES FROM SURFACE FEATURES. *Int. J. Comput. Geom. Appl.*, World Scientific Publishing Co., v. 01, n. 03, p. 281–297, set. 1991. ISSN 0218-1959.

[120] KIM, Y. S. CHAPTER 3 - volumetric feature recognition using convex decomposition. In: SHAH, J. J.; MÄNTYLÄ, M.; NAU, D. S. (Ed.). *Manufacturing Research and Technology*. [S.l.]: Elsevier, 1994. v. 20, p. 39–63.

[121] SAKURAI, H. Volume decomposition and feature recognition: part 1—polyhedral objects. *Comput. Aided Des. Appl.*, v. 27, n. 11, p. 833–843, nov. 1995. ISSN 1686-4360, 0010-4485.

[122] SAKURAI, H.; DAVE, P. Volume decomposition and feature recognition, part II: curved objects. *Comput. Aided Des. Appl.*, v. 28, n. 6, p. 519–537, jun. 1996. ISSN 1686-4360, 0010-4485.

[123] ZHANG, Z.; JAISWAL, P.; RAI, R. Featurenet: Machining feature recognition based on 3d convolution neural network. *Computer-Aided Design*, v. 101, p. 12–22, 2018. ISSN 0010-4485. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010448518301349>.

[124] Wikipedia contributors. *Watershed (image processing)*. 2024. `https://en.wikipedia.org/w/index.php?title=Watershed_(image_processing)&oldid=1216016400`. Accessed: 2023-11-1.

[125] SHI, P. et al. A novel learning-based feature recognition method using multiple sectional view representation. *J. Intell. Manuf.*, v. 31, n. 5, p. 1291–1309, jun. 2020. ISSN 0956-5515.

[126] Wikipedia contributors. *Transfer learning*. 2024. `https://en.wikipedia.org/w/index.php?title=Transfer_learning&oldid=1214495796`. Accessed: 2023-11-1.

[127] SHI, P. et al. Intersecting machining feature localization and recognition via single shot multibox detector. *IEEE Trans. Ind. Inf.*, IEEE, v. 17, n. 5, p. 3292–3302, maio 2021. ISSN 1551-3203, 1941-0050.

[128] LIU, W. et al. SSD: Single shot MultiBox detector. dez. 2015.

[129] MA, Y.; ZHANG, Y.; LUO, X. Automatic recognition of machining features based on point cloud data using convolution neural networks. In: *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*. New York, NY, USA: Association for Computing Machinery, 2019. (AICS 2019), p. 229–235. ISBN 9781450371506.

[130] JIA, J.-L. et al. Machining feature recognition method based on improved mesh neural network. *Iranian Journal of Science and Technology, Transactions of Mechanical Engineering*, v. 47, n. 4, p. 2045–2058, dez. 2023. ISSN 2364-1835.

[131] REN, S. et al. Faster R-CNN: Towards Real-Time object detection with region proposal networks. jun. 2015.

[132] JAYARAMAN, P. K. et al. UV-Net: Learning from boundary representations. jun. 2020.

[133] ZHANG, H. et al. Machining feature recognition based on a novel multi-task deep learning network. *Robot. Comput. Integr. Manuf.*, v. 77, p. 102369, out. 2022. ISSN 0736-5845.

[134] COMANICIU, D.; MEER, P. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE, v. 24, n. 5, p. 603–619, maio 2002. ISSN 0162-8828, 2160-9292.

[135] NING, F. et al. Manufacturing cost estimation based on a deep-learning method. *Journal of Manufacturing Systems*, v. 54, p. 186–195, jan. 2020. ISSN 0278-6125.

[136] TRAINBR. `https://www.mathworks.com/help/deeplearning/ref/trainbr.html`. Accessed: 2024-1-19.

[137] TRAINLM. `https://www.mathworks.com/help/deeplearning/ref/trainlm.html`. Accessed: 2024-1-19.

[138] FLASI?SKI, M. SYNTACTIC PATTERN RECOGNITION: PARADIGM ISSUES AND OPEN PROBLEMS. In: *Handbook of Pattern Recognition and Computer Vision*. [S.l.]: WORLD SCIENTIFIC, 2015. p. 3–25. ISBN 9789814656528.

[139] NING, F. et al. Feature-Based and Process-Based manufacturing cost estimation. *Machines MDPI*, 2022.

[140] YOO, S.; KANG, N. Explainable artificial intelligence for manufacturing cost estimation and machining feature visualization. *Expert Syst. Appl.*, Elsevier BV, v. 183, n. 115430, p. 115430, nov. 2021. ISSN 0957-4174, 1873-6793.

[141] MISUMI: Custom Part Configurations, Factory Automation Components, Industrial Supplies, Tools, & More. `https://us.misumi-ec.com/`. Accessed: 2024-1-17.

[142] FreeCAD: Your own 3D parametric modeler. `https://www.freecad.org/index.php`. Accessed: 2024-1-17.

[143] FEED, F. C.; FOLLOWING, M.; PREFERENCES, C. *Mesh voxelisation - File Exchange - MATLAB CentralFile Exchange - MATLAB Central*. 2013. `https://www.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation`. Accessed: 2024-1-17.

[144] IGLESIA, D. de la. *3D point cloud generation from 3D triangular mesh - David de la Iglesia*. 2017. `https://medium.com/@daviddelaiglesiacastro/3d-point-cloud-generation-from-3d-triangular-mesh-bbb602ecf238`. Accessed: 2024-1-17.

[145] SELVARAJU, R. R. et al. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. [S.l.]: IEEE, 2017. ISBN 9781538610329.

[146] PAVIOT, T. *pythonocc-core: Python package for 3D CAD/BIM/PLM/CAM.*

[147] BOX, G. E. P.; COX, D. R. An analysis of transformations. *J. R. Stat. Soc.*, Wiley, v. 26, n. 2, p. 211–243, jul. 1964. ISSN 0035-9246, 1467-9868.

[148] OCCT: Open CASCADE Technology (OCCT) is an open-source software development platform for 3D CAD, CAM, CAE. This is a clone of the official repository located on https://dev.opencascade.org/. Please use official development portal for registering issues and providing patches.

[149] FEY, M.; LENSSEN, J. E. *PyG: Geometric Deep Learning Extension Library for PyTorch*. 2019. GitHub repository. Disponível em: <`https://github.com/rusty1s/pytorch_geometric`>.