Universidade Federal do Rio Grande - FURG
Centro de Ciências Computacionais
Programa de Pós-Graduação em Computação
Mestrado em Engenharia de Computação

Dissertação de Mestrado

Alterando o mecanismo de atenção na arquitetura Transformer aplicando funções de (pré-)agregação

Joelson Sartori Junior

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para obtenção do grau de Mestre em Engenharia de Computação.

Orientadora: Prof. Dra. Helida Santos Salles

Co-orientadores: Prof. Dra. Graçaliz Pereira Dimuro

Prof. Dr. Giancarlo Lucca

Rio Grande-RS, Março de 2025

Ficha Catalográfica

S251a Sartori Junior, Joelson.
Alterando o mecanismo de atenção na arquitetura *Transformer* aplicando funções de (pré-)agregação / Joelson Sartori Junior. – 2025.
95 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2025.
Orientadora: Dra. Helida Santos Salles.
Coorientadora: Dra. Graçaliz Pereira Dimuro.
Coorientador: Dr. Giancarlo Lucca.

1. Redes Neurais 2. *Transformers* 3. Funções de agregação 4. Integral de *Choquet* 5. Integral de Sugeno I. Salles, Helida Santos II. Dimuro, Graçaliz Pereira III. Lucca, Giancarlo IV. Título.

CDU 004

Catalogação na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

Alterando o mecanismo de atenção na arquitetura Transformer aplicando funções de (pré-)agregação

Joelson Sartori Junior

Banca examinadora:

Documento assinado digitalmente
GIANCARLO LUCCA
Data: 18/06/2025 12:09:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Giancarlo Lucca – UCPEL

Documento assinado digitalmente
BENJAMIN RENE CALLEJAS BEDREGAL
Data: 24/06/2025 21:07:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Benjamin Bedregal – UFRN

Documento assinado digitalmente
RODRIGO ANDRADE DE BEM
Data: 23/06/2025 20:28:58-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rodrigo De Bem – FURG

Documento assinado digitalmente
GRACALIZ PEREIRA DIMURO
Data: 15/06/2025 21:27:38-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Graçaliz Dimuro – FURG

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

BERT Pre-training of Deep Bidirectional Transformers for Language Understanding. 28–31, 38

CALTECH-101 California Institute of Technology Dataset. vi, 57, 58

Caltech101 California Institute of Technology 101 Dataset. iv, vi, 1, 2, 63, 68–70, 74

CIFAR Canadian Institute For Advanced Research. 55

CIFAR-10 Canadian Institute For Advanced Research 10 Dataset. iv, vi, 1, 2, 55–57, 63–66, 72, 74

CIFAR-100 Canadian Institute For Advanced Research 100 Dataset. iv, vi, 1, 2, 56, 57, 63, 66–68, 74

COCO Common Objects in Context. iv, vi, 1, 2, 58, 59, 63, 70–72, 74

dCC dCC-integral. iv, 1, 2, 40, 50–54

DETR Detection Transformer. 34, 39

DINO Self-Distillation with No Labels. 34, 35, 39

DL Deep Learning. 79

FG Functional Generalization. 2, 56, 58, 59, 62, 77

FSTA Federation of Software and Advanced Technology. 78

GINFO Grupo de Informática. 78

GLUE General Language Understanding Evaluation. 30, 31

Google Colab Google Colaboratory. 61

GPT Generative Pre-trained Transformer. 29

GPT-2 Generative Pre-trained Transformer 2. 32, 33

GPU Graphics Processing Unit. 42, 45–48, 61

IA Inteligência Artificial. 3, 40, 54, 62, 79

MLM Masked Language Modeling. 29

NSP Next Sentence Prediction. 29

PLN Processamento de Linguagem Natural. 5, 27–33, 38, 62, 78

Point Transformer Point Transformer. 36

PyTorch Python Torch Framework. 61

PyTorch Python Torch Framework. 10, 12

RDF Restricted Dissimilarity Function. 50–53

RNA Rede Neural Artificial. 3

RNC Rede Neural Convolucional. 3, 27, 28, 31, 32

RNR Rede Neural Recorrente. 3, 27

SIBGRAPI Brazilian Symposium on Computer Graphics and Image Processing. 78

SQuAD Stanford Question Answering Dataset. 30, 31

Transformer-XL Transformer with Extra Long Context. 31

ViT Vision Transformer. iv, vii, 27, 28, 31, 32, 35, 38, 60, 61, 63, 64, 70, 72, 73, 75, 78

XLNet Generalized Autoregressive Pretraining for Language Understanding. 30, 31, 38

Resumo

JOELSON SARTORI JUNIOR, **Alterando o mecanismo de atenção na arquitetura Transformer aplicando funções de (pré-)agregação**. Março de 2025. 87 f. Biblioteca Central – FURG, Rio Grande-RS.

Esta dissertação propõe aprimoramentos nos modelos Transformers e Visual Transformer ao incorporar *funções de (pré-)agregação* baseadas nas **integrais de Choquet**, **Integral de Sugeno** e dCC-integrals no mecanismo de autoatenção. Essas *funções*, fundamentadas no conceito de *capacidade fuzzy*, substituem a *agregação matricial* tradicional por *operações* mais sofisticadas, ampliando a capacidade dos modelos de capturar *dependências complexas* em *dados de alta dimensionalidade*. Três *abordagens* foram desenvolvidas: uma *formulação adaptada* da integral de Choquet, *generalizações* da **Integral de Sugeno** pelas funções FG-*funcionais* e pelas dCC-integrais, que combinam *cópulas* e *funções de dissimilaridade*. Os *experimentos*, realizados em *conjuntos de dados* como CIFAR-10, CIFAR-100, Caltech101 e COCO, demonstraram que as propostas alcançaram *desempenho* comparável ao *mecanismo tradicional*, com *melhorias significativas* em *generalização* e *robustez*. Este *estudo* contribui para o *avanço teórico* e *prático* de *modelos de atenção*, oferecendo uma *base sólida* para o *desenvolvimento* de *arquiteturas* mais adaptáveis e eficientes no *aprendizado de máquina*.

Palavras-chave: Redes Neurais, Transformers, Funções de agregação, Integral de Choquet, Integral de Sugeno.

Abstract

JOELSON SARTORI JUNIOR, **Changing the Attention Mechanism in the Transformer Architecture Using (Pre-)Aggregation Functions**. Março de 2025. 87 f. Biblioteca Central – FURG, Rio Grande-RS.

This dissertation proposes enhancements to Transformer and Visual Transformer models by incorporating **(pre-)aggregation functions** based on **Choquet integrals**, **Sugeno integral**, and dCC-integrals into the self-attention mechanism. These functions, grounded in the concept of fuzzy capacity, replace the traditional **matrix aggregation** with more sophisticated operations, increasing the models' ability to capture complex dependencies in high-dimensional data. Three approaches were developed: an adapted formulation of the Choquet integral, generalizations of the **Sugeno integral** through Functional Generalization (FG)-functionals, and dCC-integrals, which combine copulas and **dissimilarity functions**. The experiments, conducted on datasets such as CIFAR-10, CIFAR-100, Caltech101, and COCO, demonstrated that the proposed methods achieved performance comparable to the traditional mechanism, with significant improvements in generalization and robustness. This study contributes to the theoretical and practical advancement of **attention models**, providing a solid foundation for the development of more adaptable and efficient architectures in machine learning.

Keywords: Neural Networks, Transformers, Aggregation Functions, Choquet Integral, Sugeno Integral.

1 Introdução

A Inteligência Artificial (IA) (Russell 2010) tem se consolidado como uma das áreas mais dinâmicas e inovadoras da ciência moderna. Ela gera mudanças profundas na forma como a sociedade interage com a tecnologia e compreende o mundo ao seu redor. Desde suas origens, o campo da IA tem buscado replicar a capacidade humana de raciocinar e aprender. Esse objetivo tornou-se progressivamente mais realizável graças aos contínuos avanços em algoritmos e ao aumento do poder computacional.

Essas inovações tecnológicas possuem o potencial de transformar setores inteiros da sociedade. Elas abrangem desde a automação industrial (J. Zhang e Man 1998), passando pelo diagnóstico médico em tempo real (Liu et al. 2021), até sistemas educacionais personalizados que se ajustam ao ritmo de aprendizado dos estudantes (Ouyang e Jiao 2021).

Nesse contexto, a Rede Neural Artificial (RNA) (Abdi, Valentin e Edelman 1999) emergiu como uma das tecnologias mais influentes e revolucionárias. Inspiradas pela estrutura dos neurônios biológicos, as RNAs são projetadas para aprender e identificar padrões complexos a partir de grandes volumes de dados, permitindo avanços significativos em áreas como reconhecimento de fala (G. Hinton et al. 2012), tradução automática (Di Gangi, Negri e Turchi 2019) e visão computacional (Jing et al. 2019).

O sucesso dessas redes incentivou o desenvolvimento de arquiteturas mais avançadas, tais como a Rede Neural Convolucional (RNC) (LeCun et al. 1998) e a Rede Neural Recorrente (RNR) (Rumelhart, G. E. Hinton e Williams 1986), cada uma proje-

tada para lidar com diferentes tipos de dados e tarefas específicas.

As RNCs, em específico, foram desenvolvidas para o processamento de dados visuais, como imagens e vídeos. Essas redes utilizam operações de convolução que atuam como filtros, capazes de analisar imagens de diferentes maneiras e destacar características cruciais, como bordas, texturas e formas. A medida que uma imagem passa por múltiplas camadas dessa rede, essas características são combinadas e refinadas, resultando em uma representação mais detalhada e complexa da imagem original. Por exemplo, na análise de uma fotografia de um rosto, uma RNC pode identificar inicialmente contornos e bordas, depois reconhecer olhos, nariz e boca, e finalmente identificar o rosto como um todo.

Essa capacidade torna as RNCs extremamente eficazes para tarefas de visão computacional (Kshatri e D. Singh 2023), incluindo reconhecimento de objetos, detecção de rostos e análise de imagens médicas. Um exemplo prático do uso de RNCs é na análise de radiografias para identificar anomalias, auxiliando no diagnóstico médico (LeCun et al. 1998). A habilidade das RNCs em extrair e interpretar padrões visuais complexos torna-as essenciais para muitas aplicações que exigem análise detalhada de imagens.

Enquanto as RNCs são especializadas em dados visuais, as RNRs são projetadas para lidar com dados sequenciais, como textos e séries temporais. As RNRs têm a capacidade de manter informações ao longo do tempo, permitindo que retenham contextos passados ao processar novos dados. Essa característica é fundamental em tarefas nas quais a ordem dos dados é relevante, como na tradução de idiomas, onde o significado de uma palavra pode depender de palavras anteriores, ou na previsão de séries temporais, onde eventos passados influenciam previsões futuras (Rumelhart, G. E. Hinton e Williams 1986).

Um uso comum das RNRs é em sistemas de tradução automática, que precisam compreender não apenas o significado de palavras individuais, mas também como essas palavras relacionam-se em uma frase completa (Alahmadi, Wali e Alzahrani 2022). Por

exemplo, ao traduzir a frase “Eu vou para a escola amanhã”, a RNR deve manter o contexto de “vou” para produzir uma tradução precisa. Essa capacidade de manter uma “memória de curto prazo” para sequências de dados permite que as RNRs sejam altamente eficazes em aplicações que envolvem processamento de linguagem natural (Jelodar et al. 2020) e previsão de séries temporais (Hewamalage, Bergmeir e Bandara 2021), como a análise de preços de ações (Zhao et al. 2021) ou a interpretação de dados de sensores (Golestani e Moghaddam 2020).

Um dos avanços mais na evolução das redes neurais foi a introdução da arquitetura Transformer (Vaswani et al. 2017), que redefiniu o estado da arte em diversas tarefas de Processamento de Linguagem Natural (PLN). Diferente das RNRs, que processam dados de maneira sequencial, os Transformers utilizam um mecanismo de atenção que permite o processamento paralelo, resultando em maior eficiência e precisão. A capacidade dos Transformers de focar em partes específicas dos dados de entrada e capturar relações de longo alcance e contextos complexos de maneira eficiente revolucionou aplicações como: tradução automática (Vydana et al. 2021), geração de texto (Goyal, Kumar e V. Singh 2024), sumarização de documentos (Khandelwal et al. 2019) e o reconhecimento de imagens (Hong et al. 2021).

O elemento central do sucesso dos Transformers é o seu mecanismo de atenção. Essa abordagem utiliza a combinação de diversos valores em um só, um processo conhecido como agregação de dados. Por exemplo, ao multiplicar matrizes ou usar funções que normalizam pesos de atenção, como a função *softmax*. Essa agregação atua como um filtro, priorizando informações relevantes e minimizando o impacto de ruídos, o que é essencial em conjuntos de dados de alta dimensionalidade e complexidade. A abordagem tradicional do mecanismo de atenção, embora eficaz, apresenta limitações ao lidar com relações não lineares e incertezas, aspectos comuns em dados do mundo real. Esta observação levanta uma questão de pesquisa fundamental: **Como podemos aprimorar**

o desempenho dos Transformers modificando seu mecanismo de atenção com novas funções de agregação?

Explorar novas funções de agregação (Beliakov, Pradera, Calvo et al. 2007) oferece uma oportunidade de potencializar a arquitetura Transformer em diversos aspectos:

- **Melhoria da capacidade de generalização:** Funções de agregação diversas das comumente utilizadas podem habilitar os Transformers a identificar padrões mais complexos em dados heterogêneos, ampliando sua capacidade de generalizar conjuntos de treinamento para contextos não vistos anteriormente.
- **Redução da necessidade de grandes volumes de dados:** Otimizando o mecanismo de atenção, é possível reduzir a dependência de grandes volumes de dados rotulados, uma vez que o modelo torna-se mais eficiente em aprender a partir de uma menor quantidade de exemplos.
- **Aumento da robustez do modelo:** A introdução de novas funções de agregação pode fortalecer os modelos contra ruídos e variações nos dados, resultando em um desempenho mais consistente, mesmo em condições adversas.
- **Otimização do uso de recursos computacionais:** Novas funções de agregação que simplifiquem os cálculos do mecanismo de atenção, sem comprometer a eficácia, podem reduzir o custo computacional associado ao treinamento e à inferência, tornando os Transformers mais acessíveis para uma ampla gama de aplicações.

Com base nessas considerações, este trabalho propõe uma exploração de novas funções de agregação aplicadas ao mecanismo de atenção dos modelos Transformers, com o objetivo de aumentar sua eficácia e desempenho na análise de conjuntos de dados complexos. Além disso, a pesquisa visa avaliar a aplicabilidade dessas funções em contextos específicos, contribuindo para avanços na compreensão e no desenvolvimento de soluções inovadoras em IA. Este estudo não apenas enriquecerá a literatura acadêmica sobre a

arquitetura Transformer, mas também abrirá novas possibilidades para aplicações práticas em áreas críticas, como análise de imagens médicas, processamento de linguagem natural e detecção de fraudes em tempo real.

1.1 Questão de Pesquisa

Qual é o impacto de diferentes funções de (pré)-agregação no mecanismo de atenção dos Transformers, considerando sua influência no desempenho e na eficácia na análise de conjuntos de dados complexos?

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral desta pesquisa é investigar e analisar o impacto de novas funções de (pré)-agregação no mecanismo de atenção dos modelos Transformers, visando avaliar como essas modificações podem aprimorar o desempenho dos modelos em diferentes contextos e tarefas de aprendizado de máquina. Pretende-se compreender se a introdução de funções de agregação alternativas pode aumentar a capacidade dos Transformers de capturar dependências complexas e sutilezas nos dados, resultando em modelos mais eficientes, precisos e robustos.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, estabeleceram-se as seguintes metas:

1. Revisar e categorizar funções de (pré)-agregação (Bustince et al. 2016) baseadas em integrais fuzzy, como as integrais de Choquet e Sugeno (Choquet 1954), descritas na literatura.

2. Analisar estudos de caso que utilizam modelos Transformers, incluindo *Vision Transformers* em reconhecimento de padrões visuais.

3. Desenvolver teoricamente a adaptação das funções de (pré)-agregação baseadas nas integrais de Choquet e Sugeno ao contexto de matrizes, visando modificar o mecanismo de atenção dos Transformer (Vaswani et al. 2017).

4. Implementar e testar diferentes funções de (pré)-agregação baseadas em integrais fuzzy nos modelos Transformers e *Vision Transformers* (Dosovitskiy et al. 2020).

5. Avaliar empiricamente o desempenho dessas funções, realizando uma análise detalhada dos resultados obtidos com os novos mecanismos de atenção.

1.3 Metodologia

Para atingir o objetivo proposto, esta dissertação envolve a modificação sistemática das funções de agregação utilizadas no mecanismo de atenção dos Transformers, substituindo as operações padrão por alternativas baseadas em funções de agregação avançadas, como as integrais de Choquet e Sugeno e suas generalizações. O foco central é analisar como essas novas funções influenciam o processamento de informações pelo modelo, afetando sua capacidade de aprendizado e generalização em diferentes tarefas.

A metodologia adotada inclui a implementação dessas funções de (pré)-agregação no mecanismo de atenção dos Transformers e a realização de experimentos empíricos utilizando conjuntos de dados de referência em diversos domínios, tais como classificação de imagens, reconhecimento de linguagem gestual e outras tarefas relevantes. Foram conduzidas análises comparativas entre os modelos modificados e os modelos tradicionais, focadas em métricas de desempenho como acurácia e taxa de erro.

1.3.1 Conjuntos de Dados

Os experimentos foram realizados utilizando quatro conjuntos de dados amplamente reconhecidos na comunidade de aprendizado de máquina, cada um com características distintas que permitem avaliar o desempenho dos modelos em diferentes contextos:

- **CIFAR-10** (Krizhevsky, G. Hinton et al. 2009): Composto por 60.000 imagens coloridas de resolução 32×32 pixels distribuídas em dez classes diferentes, o CIFAR-10 é um conjunto de dados padrão para tarefas de classificação de imagens. Ele oferece um bom equilíbrio entre simplicidade e complexidade, sendo ideal para avaliar a capacidade dos modelos de reconhecer padrões básicos em imagens de baixa resolução.

- **CIFAR-100** (Krizhevsky, G. Hinton et al. 2009): Similar ao CIFAR-10, o CIFAR-100 contém 60.000 imagens coloridas de resolução 32×32 pixels, mas distribuídas em cem classes distintas. A maior diversidade de classes torna o CIFAR-100 um desafio maior para os modelos, exigindo uma capacidade de generalização mais robusta e uma discriminação mais fina entre categorias similares.

- **Caltech101** (F.-F. Li et al. 2022): Este conjunto de dados consiste em aproximadamente 9.146 imagens de objetos distribuídas em 101 categorias, com variações significativas em termos de iluminação, orientação e fundo. O Caltech101 é amplamente utilizado para avaliar a capacidade dos modelos em reconhecer objetos em condições de alta variabilidade visual, tornando-o ideal para testar a robustez das funções de agregação propostas.

- **COCO** (Common Objects in Context) (Lin et al. 2014): Com mais de 330.000 imagens e 80 categorias de objetos, o COCO é um dos conjuntos de dados mais desafiadores e abrangentes para tarefas de visão computacional. Ele inclui imagens

com múltiplos objetos em diferentes contextos, exigindo que os modelos capturem interações complexas e generalizem bem em condições altamente variáveis.

A seleção desses conjuntos de dados permite uma avaliação abrangente dos modelos Transformers modificados, cobrindo desde tarefas de classificação simples com baixo número de classes até cenários complexos com alta variabilidade e múltiplos objetos. Essa diversidade assegura que os resultados obtidos reflitam a eficácia das novas funções de agregação em uma ampla gama de aplicações de aprendizado de máquina.

Além disso, para cada conjunto de dados, foram realizadas etapas de pré-processamento específicas, como normalização das imagens, aumento de dados para melhorar a robustez do modelo e divisão adequada entre conjuntos de treino, validação e teste. Essas práticas garantem que os modelos sejam treinados e avaliados de forma consistente, permitindo comparações justas entre diferentes abordagens de agregação.

1.3.2 Implementação das Funções de Agregação

As funções de agregação baseadas em integrais fuzzy foram implementadas no mecanismo de atenção dos Transformers, substituindo as operações padrão. Isso envolveu a adaptação das integrais de Choquet e Sugeno para o contexto de matrizes, permitindo que o mecanismo de atenção capture interações não lineares e dependências complexas entre os elementos de entrada. A implementação foi realizada utilizando o *framework* Python Torch Framework (PyTorch) (**py**), que oferece flexibilidade e eficiência para a construção e treinamento de modelos de aprendizado profundo.

1.3.3 Configuração dos Experimentos

Os experimentos foram configurados de maneira a garantir a reprodutibilidade e a consistência dos resultados. Cada modelo foi treinado utilizando os mesmos hiperparâmetros básicos, como taxa de aprendizado, tamanho do *batch* e número de épocas,

permitindo uma comparação justa entre as diferentes funções de agregação. Além disso, múltiplas execuções foram realizadas para cada configuração, e os resultados foram agregados para obter métricas estatísticas robustas, como média e desvio padrão.

1.3.4 Análise de Desempenho

O desempenho dos modelos foi avaliado utilizando métricas padrão de aprendizado de máquina, incluindo acurácia e taxa de erro. Além disso, foram analisadas as curvas de aprendizado para entender a convergência dos modelos durante o treinamento e a capacidade de generalização nos conjuntos de validação.

A abordagem metodológica adotada nesta pesquisa permite uma análise abrangente e detalhada do impacto das funções de agregação no mecanismo de atenção dos Transformers, contribuindo para a compreensão e o aprimoramento das capacidades desses modelos em diferentes tarefas de aprendizado de máquina.

1.3.5 Procedimentos Experimentais

Os procedimentos experimentais seguiram as seguintes etapas:

1. **Revisão e Implementação das Funções de Agregação:** Após a revisão bibliográfica, as funções de agregação baseadas em integrais fuzzy foram implementadas no mecanismo de atenção dos Transformers. Isso incluiu a adaptação das integrais de Choquet e Sugeno para operar eficientemente com matrizes de atenção.

2. **Preparação dos Conjuntos de Dados:** Cada conjunto de dados foi pré-processado conforme as necessidades específicas de cada tarefa. Isso incluiu a normalização das imagens, aplicação de técnicas de aumento de dados para aumentar a diversidade das amostras de treinamento e divisão dos dados em conjuntos de treino, validação e teste.

3. **Treinamento dos Modelos:** Os modelos Transformers e *Vision Transformers* modificados foram treinados em cada um dos conjuntos de dados utilizando os hiperparâmetros previamente estabelecidos. O treinamento foi monitorado através de métricas de desempenho em tempo real para garantir a convergência adequada.

4. **Avaliação e Comparação:** Após o treinamento, os modelos foram avaliados nos conjuntos de validação e teste. As métricas de desempenho foram calculadas e comparadas entre os modelos modificados e os modelos tradicionais, permitindo a identificação das melhorias proporcionadas pelas novas funções de agregação.

5. **Análise de Resultados:** Os resultados obtidos foram analisados quantitativamente e qualitativamente. A análise quantitativa envolveu a comparação das métricas de desempenho, enquanto a análise qualitativa incluiu a inspeção de exemplos específicos para entender como as novas funções de agregação influenciaram o comportamento dos modelos.

1.3.6 Ferramentas e Tecnologias Utilizadas

Para a implementação e execução dos experimentos, foram utilizadas as seguintes ferramentas e tecnologias:

- **Python**[1]: Linguagem de programação escolhida devido à sua versatilidade e vasta biblioteca de ferramentas para aprendizado de máquina.
- **PyTorch**[2]: Framework de aprendizado profundo utilizado para a construção e treinamento dos modelos Transformers modificados.
- **Google Colab**[3]: Ambiente de desenvolvimento que fornece recursos computacionais, incluindo GPUs, necessários para o treinamento eficiente dos modelos.

[1] https://www.python.org/
[2] https://pytorch.org/
[3] https://colab.google/

- **Bibliotecas Complementares**: Incluem *NumPy*[4], *Pandas*[5] para manipulação de dados, *Matplotlib*[6] e *Seaborn*[7] para visualização de resultados.

1.3.7 Validação e Reprodutibilidade

Para assegurar a validade dos resultados e a reprodutibilidade dos experimentos, adotaram-se as seguintes práticas:

- **Configuração de Seed Fixa:** Utilização de uma **seed** fixa (1601) para a inicialização dos pesos dos modelos, reduzindo a variabilidade entre execuções e garantindo a comparabilidade dos resultados.
- **Documentação Detalhada:** Cada passo dos experimentos foi cuidadosamente documentado, incluindo as configurações dos hiperparâmetros, as versões das bibliotecas utilizadas e os procedimentos de pré-processamento dos dados.
- **Repositório de Código:** Todo o código implementado foi organizado em um repositório público (por exemplo, GitHub[8]), permitindo que outros pesquisadores reproduzam os experimentos e validem os resultados obtidos.

1.3.8 Considerações Éticas e de Uso de Dados

Todos os conjuntos de dados utilizados nos experimentos são públicos e amplamente utilizados na comunidade de aprendizado de máquina, respeitando as políticas de uso e privacidade associadas. Nenhum dado sensível ou privado foi utilizado, garantindo a conformidade com as normas éticas de pesquisa.

[4] https://numpy.org/
[5] https://pandas.pydata.org/
[6] https://matplotlib.org/stable/
[7] https://seaborn.pydata.org/
[8] https://github.com/JoelsonSartoriJr

1.4 Principal Contribuição

Espera-se que os resultados desta pesquisa contribuam para o avanço do estado da arte em modelos Transformers, oferecendo *insights* valiosos sobre como aprimorar o mecanismo de atenção por meio de funções de agregação alternativas. Tal contribuição pode abrir novas perspectivas para o desenvolvimento de modelos mais robustos e eficientes em diversas aplicações da inteligência artificial, beneficiando áreas como processamento de linguagem natural, visão computacional e reconhecimento de padrões.

1.5 Organização do Trabalho

Esta dissertação está estruturada em seis capítulos, conforme descrito a seguir:

Capítulo 1 - Introdução apresenta o contexto e a motivação do estudo, formulando a questão de pesquisa e estabelecendo os objetivos gerais e específicos. Também descreve a metodologia adotada, incluindo os procedimentos experimentais, as ferramentas utilizadas e as considerações éticas envolvidas. Ademais, discute-se o papel das integrais de Choquet, Sugeno e d-CC no aprimoramento do mecanismo de atenção em redes Transformer, ressaltando a importância dessas funções de agregação em aplicações de Processamento de Linguagem Natural e Visão Computacional.

Capítulo 2 - Fundamentação Teórica aborda os principais conceitos teóricos que embasam o trabalho. Inicia-se com uma descrição sobre funções de (pré)-agregação e integrais fuzzy, enfatizando as integrais de Choquet e Sugeno e suas propriedades. Em seguida, são discutidos conceitos de redes neurais, incluindo arquiteturas como Redes Neurais Convolucionais e Recorrentes. Por fim, apresenta-se a arquitetura Transformer, explorando seus componentes e o funcionamento do mecanismo de autoatenção.

Capítulo 3 - Novos Métodos de Self-Attention Baseados em Integrais Fuzzy introduz as propostas de alteração do mecanismo de atenção no Transformer.

São investigadas, de maneira aprofundada, três abordagens distintas:

- Mecanismo de Atenção Baseado na Integral de Choquet;
- Mecanismo de Atenção Baseado na Integral de Sugeno;
- Mecanismo de Atenção Baseado na d-CC-integral.

São abordados o desenvolvimento teórico de cada método, seus aspectos de implementação e suas possíveis vantagens na captura de interações não lineares e complexas nos dados.

Capítulo 4 - Conjuntos de Dados descreve detalhadamente os conjuntos de dados utilizados nos experimentos, com ênfase nas características do CIFAR-10, CIFAR-100, Caltech-101 e COCO. Discute-se como esses datasets foram escolhidos para avaliar a eficácia das novas funções de agregação propostas, bem como os desafios e particularidades de cada um.

Capítulo 5 - Experimentos e Resultados apresenta a metodologia experimental, detalhando a configuração dos testes, os parâmetros de treinamento e as métricas de avaliação. Em seguida, são expostos os resultados obtidos, com comparações entre o modelo Transformer padrão e as variantes que utilizam integrais de Choquet, Sugeno e d-CC. Essa análise possibilita avaliar o impacto das novas funções de agregação no desempenho dos modelos em tarefas de classificação e detecção.

Capítulo 6 - Conclusão sintetiza as principais contribuições e resultados alcançados ao longo da pesquisa. Discorre-se sobre a relevância das integrais de Choquet, Sugeno e d-CC para o aprimoramento do mecanismo de atenção em Transformers, realçando os ganhos de desempenho e robustez observados. Por fim, são sugeridas direções para trabalhos futuros, apontando oportunidades de investigação tanto em aspectos teóricos quanto em aplicações práticas no âmbito da inteligência artificial.

2 Fundamentação Teórica

Neste capítulo, apresentamos os conceitos fundamentais que sustentam as análises e experimentos realizados nesta dissertação. Inicialmente, abordamos as noções essenciais de funções de agregação (M. Grabisch et al. 2009), e pré-agregação (Bustince et al. 2016) além de introduzir conceitos fundamentais relacionados à função *Softmax* (Gao e Pavel 2017) e redes neurais (LeCun et al. 1998). Esses conceitos servem como alicerces para a construção das metodologias propostas e para a interpretação dos resultados obtidos.

Em seguida, aprofundamos a discussão em torno das funções que têm maior relevância para este estudo, destacando-se a integral de Choquet (Choquet 1954) e a integral de Sugeno (Murofushi e M. Sugeno 1991). A integral de Choquet, uma ferramenta amplamente aplicada na teoria de conjuntos *fuzzy* (Zadeh 1965), será examinada com foco em suas propriedades e aplicações práticas. Similarmente, a integral de Sugeno, que desempenha um papel crucial na tomada de decisões sob incerteza (S.-L. Wu et al. 2017), será explorada para proporcionar uma compreensão abrangente de suas características e importância.

2.1 Funções de (Pré)-Agregação e Integrais Fuzzy

As funções de agregação desempenham um papel essencial na análise e processamento de dados em diversas áreas do aprendizado de máquina (Wilkin e Beliakov 2015).

Elas permitem combinar múltiplas entradas em uma única saída, sintetizando informações provenientes de diferentes fontes. Esse processo é particularmente relevante em modelos de aprendizado profundo, onde a capacidade de integrar e interpretar grandes volumes de dados é crucial para alcançar resultados precisos e eficientes. Para que uma função seja classificada como função de agregação, ela deve atender a duas propriedades principais: crescimento (*monotonicidade*) e condições de contorno.

Definição 1 *Uma função* $A\colon [0,1]^n \to [0,1]$ *é classificada como uma função de agregação se satisfizer as seguintes condições:(Beliakov, Pradera, Calvo et al. 2007; M. Grabisch et al. 2009)*

(A1) ***Crescimento:*** *A função* A *é crescente em cada argumento. Isto é, para cada* $i \in \{1, \ldots, n\}$*, se* $x_i \leq y$*, então* $A(x_1, \ldots, x_n) \leq A(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$*.*

(A2) ***Condições de contorno:*** $A(0, \ldots, 0) = 0$ *e* $A(1, \ldots, 1) = 1$*.*

Um exemplo simples de função de agregação é o operador **mínimo**. Considere $A(x, y) = \min(x, y)$. Aplicando os valores $x = 0.4$ e $y = 0.7$, temos:

$$A(x, y) = \min(0.4, 0.7) = 0.4. \tag{2.1}$$

Neste caso, a função preserva a propriedade de crescimento, pois ao aumentar x ou y, o valor da função também aumenta.

2.1.1 Integrais Fuzzy

As integrais *fuzzy* estendem o conceito clássico de integrais para lidar com cenários de incerteza e dependências não lineares entre variáveis. Elas são amplamente utilizadas em tarefas que envolvem agregação ponderada, como em sistemas de decisão multicritério ou aprendizado de máquina. A definição dessas integrais está diretamente ligada ao conceito de medida *fuzzy*.

Definição 2 *Seja N um conjunto finito e não vazio, e 2^N o conjunto das partes de N. Uma função $\mathfrak{m}\colon 2^N \to [0,1]$ é uma medida fuzzy se satisfizer:(Michel Grabisch 2016)*

($\mathfrak{m}$**1**) ***Crescimento:*** *Se $X \subseteq Y$, então $\mathfrak{m}(X) \leq \mathfrak{m}(Y)$.*

($\mathfrak{m}$**2**) ***Condições de contorno:*** $\mathfrak{m}(\emptyset) = 0$ *e* $\mathfrak{m}(N) = 1$.

Integral Discreta de Sugeno

A integral Discreta de Sugeno é uma ferramenta amplamente utilizada para integrar dados em relação a uma medida *fuzzy*, ponderando as entradas com base na sua relevância relativa.

Definição 3 *Seja $\mu\colon 2^N \to [0,1]$ uma medida fuzzy. A Integral discreta de Sugeno, denotada por Su_μ, é definida como:(Michio Sugeno 1974; M. Grabisch et al. 2009)*

$$Su_\mu(\vec{x}) = \bigvee_{i=1}^{n} \left(x_{(i)} \wedge \mu(A_{(i)})\right), \tag{2.2}$$

onde $(x_{(1)}, \ldots, x_{(n)})$ é uma permutação crescente de $\vec{x} = (x_1, \ldots, x_n)$ tal que $x_{(i-1)} \leq x_{(i)}$, $x_0 = 0$, e $A_{(i)} = \{(i), \ldots, (n)\}$.

Exemplo 1: Considere três variáveis

$$\vec{x} = (x_1, x_2, x_3) = (0.3, 0.7, 0.5)$$

e a medida $\mu(A) = \dfrac{|\,A\,|}{3}$. Ordenando em ordem crescente, $(x_{(1)}, x_{(2)}, x_{(3)}) = (0.3, 0.5, 0.7)$, obtemos:

$$\mathcal{S}_\mu(\vec{x}) = \bigvee_{i=1}^{3} \big[x_{(i)} \wedge \mu\big(A_{(i)}\big)\big]$$
$$= \bigvee\{\, 0.3 \wedge 1,\ 0.5 \wedge \tfrac{2}{3},\ 0.7 \wedge \tfrac{1}{3}\,\}$$
$$= \bigvee\{0.3,\ 0.5,\ 0.33\} = 0.5,$$

$$A_{(1)} = \{1,2,3\}, \quad A_{(2)} = \{2,3\}, \quad A_{(3)} = \{3\}.$$

Exemplo 2: Considere agora

$$\vec{x} = (x_1, x_2, x_3) = (0.6, 0.2, 0.6)$$

com a mesma medida $\mu(A) =\mid A \mid /3$. Como $x_1 = x_3$, temos duas permutações ordenantes; em ambos os casos o resultado é igual:

(i) $\pi_1 = (2,1,3)$, $\quad (x_{(1)}, x_{(2)}, x_{(3)}) = (0.2, 0.6, 0.6)$,

$$A_{(1)} = \{2,1,3\}, \quad A_{(2)} = \{1,3\}, \quad A_{(3)} = \{3\},$$

$$\mathcal{S}_\mu(\vec{x}) = \bigvee\{\, 0.2 \wedge 1,\ 0.6 \wedge \tfrac{2}{3},\ 0.6 \wedge \tfrac{1}{3}\,\}$$
$$= \bigvee\{0.2,\ 0.6,\ 0.33\} = 0.6.$$

(ii) $\pi_2 = (2,3,1)$, $(x_{(1)}, x_{(2)}, x_{(3)}) = (0.2, 0.6, 0.6)$,

$$A_{(1)} = \{2,3,1\}, \quad A_{(2)} = \{3,1\}, \quad A_{(3)} = \{1\},$$

$$\begin{aligned}\mathcal{S}_\mu(\vec{x}) &= \bigvee\{\, 0.2 \wedge 1,\ 0.6 \wedge \tfrac{2}{3},\ 0.6 \wedge \tfrac{1}{3}\,\} \\ &= \bigvee\{0.2,\ 0.6,\ 0.33\} = 0.6.\end{aligned}$$

Em ambos os casos, apesar das permutações diferenciarem a ordem dos índices, como $\mu(A_{(2)}) = \frac{2}{3}$ em $\{1,3\}$ ou $\{3,1\}$, o valor final do integral de Sugeno é o mesmo (0.6), mostrando a definição independente da permutação.

Integral Discreta de Choquet

Na origem, Choquet introduziu o conceito de *capacidade* (capacity), ou seja, uma função monotônica

$$v\colon 2^N \to \mathbb{R}, \quad v(\emptyset) = 0.$$

Quando tais capacidades são adaptadas para assumirem valores no intervalo $[0,1]$ e normalizadas de modo que $v(N) = 1$, obtêm-se exatamente as *medidas fuzzy* $\mu\colon 2^N \to [0,1]$.

A integral discreta de Choquet é uma alternativa à integral discreta de Sugeno, permitindo modelar interações entre variáveis por meio de uma agregação ponderada baseada em medidas *fuzzy*.

Definição 4 *Seja* $\mu\colon 2^N \to [0,1]$ *uma medida fuzzy (capacidade normalizada). Ordene as componentes de* $\vec{x} = (x_1, \ldots, x_n)$ *de forma não decrescente, obtendo* $\{x_{(1)}, \ldots, x_{(n)}\}$

com

$$x_{(0)} = 0, \qquad x_{(1)} \leqslant \cdots \leqslant x_{(n)},$$

e defina

$$A_{(i)} = \{(i), (i+1), \ldots, (n)\}, \quad i = 1, \ldots, n.$$

A integral discreta de Choquet de $\vec{x}$ relativa a μ é então dada por(Choquet 1954)

$$\mathfrak{C}_\mu(\vec{x}) = \sum_{i=1}^{n} \big(x_{(i)} - x_{(i-1)}\big)\ \mu\big(A_{(i)}\big). \tag{2.3}$$

Exemplo 1: Usando $\vec{x} = (0.3, 0.7, 0.5)$ e $\mu(A) = \frac{|A|}{3}$, ordene $(x_{(1)}, x_{(2)}, x_{(3)}) = (0.3, 0.5, 0.7)$. Então:

$$\begin{aligned}
\mathfrak{C}_\mu(\vec{x}) &= (x_{(1)} - x_{(0)})\ \mu(A_{(1)}) + (x_{(2)} - x_{(1)})\ \mu(A_{(2)}) + (x_{(3)} - x_{(2)})\ \mu(A_{(3)}) \\
&= (0.3 - 0) \cdot 1 + (0.5 - 0.3) \cdot \tfrac{2}{3} + (0.7 - 0.5) \cdot \tfrac{1}{3} \\
&= 0.3 + 0.134 + 0.066 \\
&= 0.5,
\end{aligned}$$

onde

$$A_{(1)} = \{1, 2, 3\}, \quad A_{(2)} = \{2, 3\}, \quad A_{(3)} = \{3\}.$$

Exemplo 2: Com $\vec{x} = (0.6, 0.2, 0.6)$ e $\mu(A) = \frac{|A|}{3}$, há duas permutações:

$$
\begin{aligned}
\text{(i) } \pi_1 &= (2,1,3),\\
(x_{(1)}, x_{(2)}, x_{(3)}) &= (0.2, 0.6, 0.6),\\
A_{(1)} &= \{2,1,3\}, \quad A_{(2)} = \{1,3\}, \quad A_{(3)} = \{3\},\\
\mathfrak{C}_\mu(\vec{x}) &= (0.2-0)\cdot 1 + (0.6-0.2)\cdot \tfrac{2}{3} + (0.6-0.6)\cdot \tfrac{1}{3}\\
&= 0.2 + 0.2667 + 0\\
&= 0.4667;
\end{aligned}
$$

$$
\begin{aligned}
\text{(ii) } \pi_2 &= (2,3,1),\\
(x_{(1)}, x_{(2)}, x_{(3)}) &= (0.2, 0.6, 0.6),\\
A_{(1)} &= \{2,3,1\}, \quad A_{(2)} = \{3,1\}, \quad A_{(3)} = \{1\},\\
\mathfrak{C}_\mu(\vec{x}) &= (0.2-0)\cdot 1 + (0.6-0.2)\cdot \tfrac{2}{3} + (0.6-0.6)\cdot \tfrac{1}{3}\\
&= 0.2 + 0.2667 + 0\\
&= 0.4667.
\end{aligned}
$$

2.1.2 Generalizações

A integral de Choquet e de Sugeno podem ser generalizadas para incorporar funções mais complexas. Um exemplo é a FG-funcional, que substitui os operadores de soma e máximo por funções F e G mais gerais.

Definição 5 *Seja* $\mu\colon 2^N \to [0,1]$ *uma medida fuzzy simétrica,* $F\colon [0,\infty[\times[0,1] \to [0,\infty]$ *uma função binária, e* $G\colon [0,\infty]^n \to [0,\infty]$ *uma função n-ária. A FG-funcional é definida*

como:(Bardozzo et al. 2021; Murofushi e M. Sugeno 1991)

$$Su_{\mu}^{F,G}(\vec{x}) = G\left(F\left(x_{(1)}, \mu(A_{(1)})\right), \ldots, F\left(x_{(n)}, \mu(A_{(n)})\right)\right), \tag{2.4}$$

onde $\vec{x} = (x_1, \ldots, x_n)$ *e* $A_{(i)} = \{(i), \ldots, (n)\}$.

A FG-funcional permite flexibilidade adicional no cálculo da agregação, sendo especialmente útil em problemas que envolvem dependências complexas entre variáveis.

Para aprofundar a compreensão sobre as funções de agregação no contexto dos Transformers, aplicamos tanto a integral de *Sugeno* padrão quanto as funções *FG-funcionais*, baseando-se nas funções descritas na Tabela 2.1.

Tabela 2.1: Fórmulas e funções G e F associadas às principais integrais *fuzzy*.

Integral	Fórmula	$G(\vec{x})$	$F(x, y)$
Sugeno	$Su_{\mu}(\vec{x}) = \bigvee_{i=1}^{n} \left(x_{(i)} \wedge \mu(A_{(i)})\right)$	$Max(\vec{x}) = \max_{i=1}^{n} x_i$	$Min(x, y) = \min(x, y)$
FG-funcional	$Su_{\mu}^{F,G}(\vec{x}) = G\left(F(x_{(1)}, \mu(A_{(1)})), \ldots, F(x_{(n)}, \mu(A_{(n)}))\right)$	$Max(\vec{x}) = \max_{i=1}^{n} x_i$	$Prod(x, y) = x \cdot y$
		$Sum(\vec{x}) = \sum_{i=1}^{n} x_i$	$Min(x, y) = \min(x, y)$
		$\prod_{i=1}^{n} x_i$	
Choquet	$\mathfrak{C}_{\mu}(\vec{x}) = \sum_{i=1}^{n} \left(x_{(i)} - x_{(i-1)}\right) \mu(A_{(i)})$	$Sum(\vec{x}) = \sum_{i=1}^{n} x_i$	$Prod(x, y) = x \cdot y$

Esses conceitos fundamentais precedem o aprofundamento do estudo de aplicar as funções de agregação em modelos Transformers, conforme explorado nas seções subsequentes.

2.2 Redes Neurais

As redes neurais (Ghosh-Dastidar e Adeli 2009) são sistemas de computação inspirados no funcionamento do cérebro humano, compostos por neurônios artificiais

dispostos em camadas. Elas são capazes de aprender e realizar tarefas complexas, como reconhecimento de padrões, classificação, previsão e processamento de dados. Tipicamente, as redes neurais são estruturadas em três camadas principais—entrada, camadas ocultas e saída—porém podem incluir múltiplas camadas ocultas, como ilustrado na Figura 2.1. Os neurônios de uma camada estão interconectados com os neurônios das camadas adjacentes através de conexões ponderadas. Esses pesos são ajustados durante o treinamento da rede, permitindo que ela aprenda a identificar padrões nas entradas e a produzir as saídas desejadas.

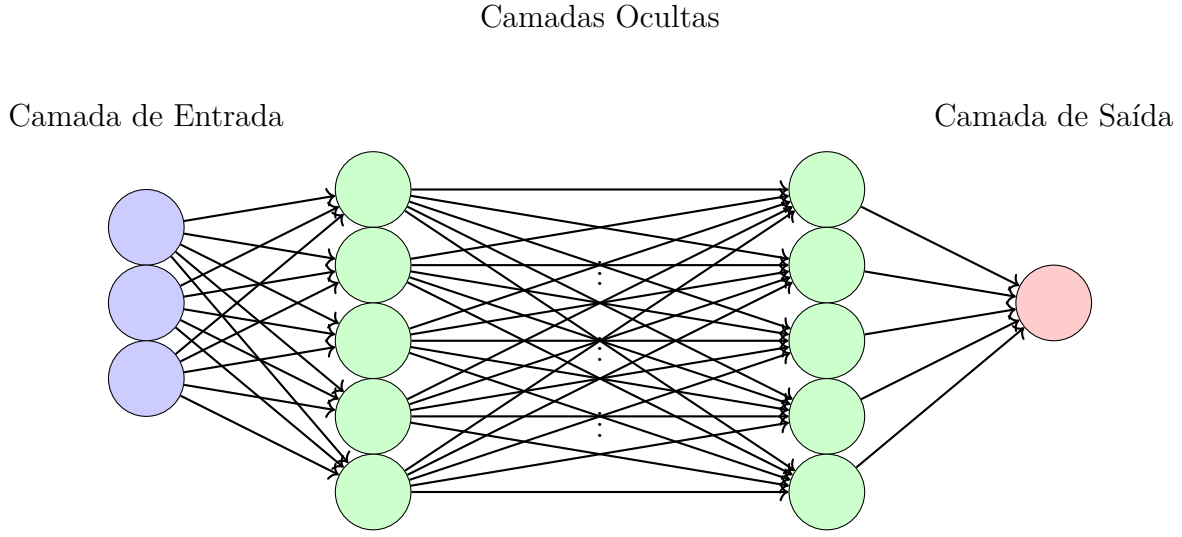

Figura 2.1: Representação de uma rede neural com duas camadas ocultas

O aprendizado em redes neurais pode ser supervisionado, com entradas acompanhadas por rótulos que orientam o aprendizado, ou não supervisionado, onde a rede deve encontrar padrões nos dados de maneira autônoma. A informação flui pela rede através das camadas, onde cada neurônio recebe entradas ponderadas, aplica uma função de ativação e envia o resultado para os neurônios da camada seguinte.

Para cada entrada x_i na rede, duas operações principais são realizadas. Primeiro, cada entrada é ponderada por um fator específico, denominado peso w_i. Esse peso ajusta a importância relativa de cada entrada para a ativação do neurônio. Em seguida, todas

as entradas ponderadas são somadas, e um termo de viés b é adicionado, permitindo ajustes finos na saída do neurônio. O resultado é o valor z, que representa a entrada para a função de ativação do neurônio. Essa sequência de operações é descrita pela Equação (2.5).

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b. \tag{2.5}$$

A função de ativação transforma a entrada z em uma saída y utilizando uma função não linear f, conforme descrito na Equação 2.6.

$$y = f(z). \tag{2.6}$$

Dentre as funções de ativação mais comuns estão a função sigmoide (Cybenko 1989), a tangente hiperbólica (Ganea, Bécigneul e Hofmann 2018) e a $ReLU$ (acrônimo em inglês do termo *Rectified Linear Unit*), esta última definida na Equação 2.7. A função $ReLU$ é amplamente utilizada devido à sua simplicidade e eficácia:

$$f(x) = \max(0, x). \tag{2.7}$$

Um exemplo da aplicação da função $ReLU$ ilustra como ela ativa valores positivos e inibe valores negativos:

Dadas as entradas $[-2, -1, 0, 1, 2]$, a aplicação da função $ReLU$ tem os seguintes resultados:

- Para -2 e -1, a saída é 0 (valores negativos são inibidos).
- Para 0, a saída é 0 (limiar da função).
- Para 1 e 2, as saídas são 1 e 2, respectivamente (valores positivos são mantidos).

Assim, a saída da $ReLU$ é $[0, 0, 0, 1, 2]$.

Em problemas de classificação multiclasse, a função de ativação *Softmax* é frequentemente utilizada na camada de saída. Ela converte as saídas dos neurônios em uma distribuição de probabilidade sobre várias classes, assegurando que a soma das probabilidades seja igual a 1. A função *Softmax* é descrita pela Equação (2.8):

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}. \tag{2.8}$$

Onde e é a base do logaritmo natural, z_i é a saída do neurônio i, e k é o número total de neurônios na camada de saída. A função exponencial torna os valores positivos, e a normalização garante que os resultados estejam na faixa de 0 a 1, permitindo sua interpretação como probabilidades.

Exemplo da aplicação da função *Softmax*:

Considerando as entradas $[1, 2, 3]$, os valores exponenciais são aproximadamente $[2.718, 7.389, 20.085]$. A soma total desses valores é 30.192. Aplicando a *Softmax* a cada entrada, tem-se, para o valor de entrada igual a:

- 1: $\frac{2.718}{30.192} \approx 0.090$;
- 2: $\frac{7.389}{30.192} \approx 0.245$;
- 3: $\frac{20.085}{30.192} \approx 0.665$.

Portanto, a saída da *Softmax* é aproximadamente $[0.090, 0.245, 0.665]$, representando as probabilidades de cada classe.

Esses exemplos demonstram o papel fundamental das funções de ativação na formação da saída de uma rede neural, auxiliando na modelagem de relações complexas e não lineares entre as entradas.

2.3 A Arquitetura Transformer

Os Transformers tiveram um impacto significativo na área de inteligência artificial ao oferecer uma nova abordagem para processar e compreender dados sequenciais. Antes do surgimento dessa arquitetura, modelos baseados em RNRs e RNCs eram amplamente utilizados para lidar com sequências e dados estruturados, como texto e áudio. No entanto, essas abordagens enfrentavam desafios significativos, especialmente em capturar dependências de longo alcance e lidar com o problema do desvanecimento de gradiente em sequências muito longas.

A arquitetura Transformer, introduzida por Vaswani et al. (2017) no artigo intitulado *Attention is All You Need*, propôs uma solução inovadora ao basear-se exclusivamente no mecanismo de *atenção*. Diferentemente das RNRs, que processam sequências de forma recursiva, os Transformers permitem processamento paralelo das entradas, resultando em maior eficiência computacional e capacidade de capturar relacionamentos distantes entre elementos da sequência.

O mecanismo de *atenção* funciona atribuindo pesos diferentes a cada parte da entrada, permitindo que o modelo concentre-se nas informações mais relevantes para a tarefa. Essa capacidade de focar em partes específicas dos dados é análoga à forma como os humanos prestam *atenção* em informações importantes ao ler um texto ou observar uma cena.

Desde sua concepção, os Transformers têm sido aplicados com sucesso em diversas áreas, notadamente em PLN, impulsionando avanços significativos em tarefas como tradução automática, resumo de textos e respostas a perguntas. A versatilidade da arquitetura também levou à sua adaptação para o domínio da visão computacional, resultando em modelos como o ViT (Dosovitskiy et al. 2020), que estende os princípios dos Transformers para a classificação e análise de imagens.

A Figura 2.2 apresenta uma linha do tempo dos principais avanços relacionados

à arquitetura Transformer, destacando marcos importantes que culminaram no desenvolvimento do *Vision Transformer*.

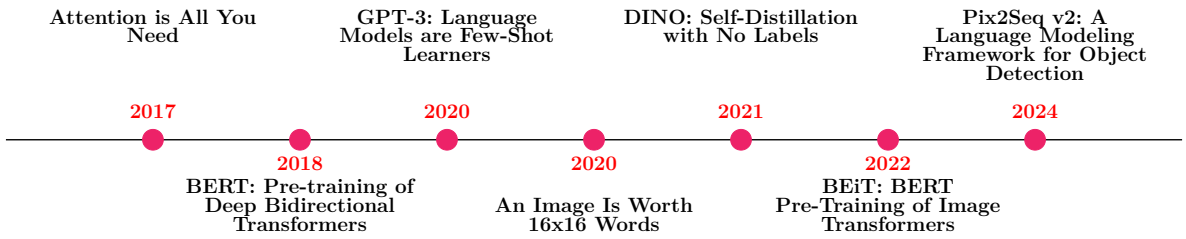

Figura 2.2: Linha do tempo com marcos importantes de artigos sobre a arquitetura Transformer (Junior et al. 2023).

A adoção dos Transformers na visão computacional marca uma evolução significativa, uma vez que tarefas como reconhecimento de imagens e detecção de objetos tradicionalmente dependiam de RNCs. O ViT demonstra que modelos baseados em *atenção* podem ser altamente eficazes também em dados visuais, abrindo novas perspectivas para pesquisas e aplicações práticas.

2.3.1 *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

O Pre-training of Deep Bidirectional Transformers for Language Understanding (BERT) (Devlin et al. 2018) revolucionou o campo do PLN ao introduzir uma nova forma de aprender representações contextuais de palavras. Diferentemente dos modelos unidirecionais que consideram apenas o contexto à esquerda ou à direita de uma palavra, o BERT é bidirecional, analisando uma palavra levando em conta simultaneamente tanto o contexto anterior quanto o posterior. Essa abordagem permite que o modelo capture um entendimento mais profundo e preciso das relações entre as palavras em uma frase.

O principal diferencial do BERT está em seu pré-treinamento bidirecional. Durante esse processo, o modelo é exposto a grandes quantidades de texto não rotulado e aprende a prever palavras que são intencionalmente ocultadas no meio de uma frase, através da técnica de Masked Language Modeling (MLM). Nesta técnica, uma porcentagem das palavras em uma sequência de texto é substituída por um token especial [MASK], e o objetivo do BERT é prever as palavras originais com base no contexto em ambos os lados. Isso permite que o modelo entenda como as palavras interagem em uma sentença completa, fornecendo uma visão mais rica do significado contextual.

Ao contrário de modelos autoregressivos como o Generative Pre-trained Transformer (GPT), que preveem a próxima palavra com base no contexto anterior, o BERT utiliza informações de todo o contexto, tanto à esquerda quanto à direita da palavra oculta. Isso faz do BERT uma escolha poderosa para tarefas que requerem uma compreensão profunda do texto, como perguntas e respostas, onde é crucial entender todo o contexto para encontrar a resposta correta.

O pré-treinamento do BERT é composto por duas tarefas principais: o MLM e a Next Sentence Prediction (NSP). Enquanto o MLM ensina o modelo a prever palavras ocultas em uma frase, o NSP ajuda o BERT a entender as relações entre sentenças. No NSP, o modelo recebe duas sentenças consecutivas e deve prever se a segunda realmente segue a primeira ou se é uma frase aleatória retirada de outra parte do texto.

Essa combinação de tarefas permite que o BERT aprenda não apenas o significado das palavras no nível da frase, mas também as relações entre frases inteiras, essencial para tarefas como resumo de textos, geração de títulos e análise de coesão textual.

Uma das características mais poderosas do BERT é que, uma vez pré-treinado, ele pode ser ajustado (*fine-tuning*) para tarefas específicas de PLN com relativa facilidade. Durante o ajuste fino, o BERT é alimentado com dados rotulados para uma tarefa

específica, como classificação de sentimentos, perguntas e respostas ou reconhecimento de entidades nomeadas. O ajuste fino requer apenas uma quantidade relativamente pequena de dados e ajustes, pois o modelo já possui uma compreensão robusta da linguagem adquirida durante o pré-treinamento.

Desde sua introdução, o BERT alcançou resultados de ponta em uma série de benchmarks de PLN, como o General Language Understanding Evaluation (GLUE) e o Stanford Question Answering Dataset (SQuAD), superando muitos dos modelos existentes. Seu impacto foi particularmente significativo em tarefas que envolvem a compreensão de contexto, como a resposta a perguntas e a análise semântica de frases.

Em aplicações práticas, o BERT é amplamente utilizado em assistentes virtuais, motores de busca, análise de sentimentos e sistemas de suporte ao cliente. A capacidade do BERT de entender nuances linguísticas e fornecer respostas baseadas no contexto completo tornou-o uma ferramenta indispensável em diversas áreas que requerem inteligência artificial avançada em linguagem natural.

2.3.2 *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

O Generalized Autoregressive Pretraining for Language Understanding (XLNet) (Yang et al. 2019) é um modelo avançado de pré-treinamento para tarefas de PLN que combina o melhor de duas abordagens: o aprendizado bidirecional, popularizado pelo BERT, e uma técnica autoregressiva para modelar dependências de longo alcance. Ao incorporar as vantagens de ambas as abordagens, o XLNet supera as limitações de modelos anteriores e melhora a captura de dependências contextuais ao longo de sequências mais longas.

O diferencial do XLNet está em sua abordagem autoregressiva generalizada. Enquanto modelos como o BERT são bidirecionais, o XLNet adota uma estratégia au-

toregressiva que preserva as propriedades de aprendizado bidirecional sem mascarar palavras durante o treinamento. Isso permite que o modelo aprenda dependências de longo prazo sem perder a ordem natural das palavras, capturando o contexto de maneira mais eficiente.

O XLNet utiliza a técnica de *Permuted Language Modeling*, onde diferentes permutações da sequência de palavras são usadas no treinamento. Essa abordagem preserva a natureza autoregressiva, permitindo que o modelo trate a sequência completa de maneira mais flexível, sem a necessidade de mascarar partes dela.

Além disso, o XLNet incorpora o Transformer with Extra Long Context (Transformer-XL) (Dai et al. 2019), uma versão aprimorada do Transformer tradicional que permite ao modelo reter informações de sequências anteriores por um período mais longo. Isso é crucial para tarefas como tradução automática, onde a compreensão precisa de frases e ideias ao longo de passagens longas é essencial para produzir traduções de alta qualidade.

O XLNet demonstrou desempenho superior em uma série de benchmarks de PLN, como o GLUE e o SQuAD, superando os resultados alcançados por modelos anteriores, incluindo o BERT. Sua capacidade de capturar dependências de longo alcance e entender o contexto em passagens longas faz com que seja particularmente eficaz em tarefas complexas, como tradução automática, geração de texto e sumarização.

2.3.3 *An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale*

Inspirados pelo sucesso dos Transformers no PLN, pesquisadores adaptaram essa arquitetura para a visão computacional, resultando no ViT (Dosovitskiy et al. 2020). Esse modelo inovador aproveita o mecanismo de *atenção* dos Transformers para processar imagens de forma eficiente, substituindo as RNCs tradicionalmente utilizadas em tarefas de visão computacional.

Em vez de processar a imagem inteira de uma só vez, o ViT divide a imagem em pequenos blocos chamados *patches*, geralmente de 16×16 pixels. Esses *patches* são linearizados (transformados em vetores) e tratados como *tokens*, de forma similar ao que é feito com palavras em modelos de PLN. A Figura **??** ilustra esse processo.

Cada *patch* é combinado com uma codificação de posição, que informa ao modelo onde esse *patch* está localizado na imagem original. Essa codificação é essencial para que o modelo entenda a estrutura espacial da imagem.

Após a conversão em uma sequência de *tokens* com codificações de posição, a imagem é processada por camadas de Transformer, assim como em tarefas de PLN. O mecanismo de autoatenção permite que o ViT capture relações globais e locais entre os diferentes *patches* da imagem, aprendendo quais partes são mais relevantes para identificar objetos ou entender cenas, sem depender de operações convolucionais.

A principal vantagem do ViT sobre as RNCs é sua capacidade de lidar com relações de maneira paralela e global. Enquanto as RNCs extraem características locais usando janelas deslizantes, o ViT é capaz de "ver"a imagem inteira desde o início, aprendendo padrões globais e contextuais ao mesmo tempo em que processa características locais.

O ViT provou ser uma alternativa eficaz aos métodos tradicionais baseados em RNCs para tarefas de visão computacional, especialmente em cenários com grandes volumes de dados disponíveis. Sua escalabilidade e simplicidade arquitetural abrem novas possibilidades para a análise visual em grande escala.

2.4 GPT-2: Generative Pre-trained Transformer 2

O Generative Pre-trained Transformer 2 (GPT-2) (Radford et al. 2019) representa um avanço significativo na aplicação dos conceitos de Transformers, especialmente na área de geração de texto de alta qualidade. Baseando-se na arquitetura de Transfor-

mers, o GPT-2 foi projetado para modelar dependências de longo alcance em sequências de texto, permitindo gerar textos que não apenas são gramaticalmente corretos, mas também altamente coerentes e contextualmente relevantes.

Utilizando uma arquitetura unidirecional focada no mecanismo de *autoatenção*, o GPT-2 prevê a próxima palavra em uma sequência considerando apenas o contexto anterior. Isso permite capturar uma ampla gama de padrões linguísticos e estruturas de frases ao longo de grandes volumes de texto.

Treinado de forma não supervisionada em um enorme corpus de dados coletados da internet, o GPT-2 aprende diretamente a partir de exemplos reais, capturando nuances da linguagem natural em diversos estilos e tópicos. Com até 1,5 bilhão de parâmetros na maior de suas versões, o modelo possui uma capacidade substancial para armazenar informações linguísticas e contextuais.

A escalabilidade do GPT-2 permite sua adaptação para uma ampla gama de tarefas de PLN, desde tradução automática até redação de e-mails e assistência em diálogos interativos. Sua habilidade de manter o contexto ao longo de várias frases e parágrafos é essencial para a geração de narrativas e explicações detalhadas.

Apesar de suas capacidades impressionantes, o GPT-2 não está isento de limitações. Uma preocupação central é a tendência de gerar informações imprecisas ou enganosas, especialmente sobre tópicos nos quais não foi diretamente treinado. Além disso, há considerações éticas sobre seu uso para criar conteúdo malicioso ou desinformação.

O GPT-2 destaca-se como um marco no desenvolvimento de modelos de geração de linguagem natural, demonstrando como os Transformers podem ser utilizados para criar textos de alta qualidade em diversos contextos. No entanto, o uso responsável dessas tecnologias é essencial para maximizar seus benefícios enquanto minimiza os riscos associados.

2.4.1 DETR: End-to-End Object Detection with Transformers

O Detection Transformer (DETR) (Detection Transformer) (Carion et al. 2020) introduziu uma abordagem inovadora para a detecção de objetos em imagens, aplicando a arquitetura de Transformer para identificar e classificar objetos de forma direta e eficiente. Diferentemente dos métodos tradicionais, que envolvem várias etapas de pré e pós-processamento, o DETR unifica todo o processo de detecção em um único modelo de aprendizado de ponta a ponta.

Combinando um *backbone* convolucional para extração de características e um Transformer que atua como *encoder-decoder*, o DETR utiliza o mecanismo de *autoatenção* global para considerar informações de todas as partes da imagem simultaneamente. Isso elimina a necessidade de componentes adicionais, como geradores de propostas e mecanismos de supressão de não-máximos.

Durante o treinamento, o DETR utiliza uma técnica de correspondência bipartida baseada no algoritmo húngaro para associar previsões de objetos com as anotações de verdade-terra. A perda de treinamento inclui erros de classificação e de localização, garantindo previsões precisas tanto em termos de classe quanto de posição.

Uma das principais vantagens do DETR é a simplificação do pipeline de detecção de objetos, resultando em uma arquitetura mais direta e fácil de adaptar para diferentes conjuntos de dados e tarefas. No entanto, o DETR requer grandes volumes de dados e tempo de treinamento relativamente longo para alcançar desempenho de estado da arte.

2.4.2 DINO: Self-Distillation with No Labels

A abordagem Self-Distillation with No Labels (DINO) (Self-Distillation with No Labels) (Caron et al. 2021) introduz um método inovador de auto-supervisão para Transformer na visão computacional, permitindo que o modelo aprenda representações visuais robustas sem a necessidade de rótulos explícitos. Essa técnica é particularmente

útil em cenários onde grandes volumes de dados não rotulados estão disponíveis.

A auto-destilação é o princípio central do DINO. O modelo atua tanto como professor quanto como aluno, ajustando suas próprias previsões para orientar o aprendizado. O DINO utiliza um Transformer como base, geralmente um ViT, para processar imagens e gerar representações visuais.

Expondo o modelo a diferentes vistas da mesma imagem, as versões aluno e professor são alimentadas com variações da entrada, promovendo a consistência entre as representações e permitindo que o modelo aprenda de maneira mais robusta.

Uma grande vantagem do DINO é a capacidade de aprender sem rótulos explícitos, o que é crucial em aplicações onde a obtenção de rótulos é dispendiosa. Apesar dos desafios, como a necessidade de grandes volumes de dados não rotulados e infraestrutura computacional significativa, o DINO exemplifica o poder dos Transformers no aprendizado auto-supervisionado em visão computacional.

2.4.3 Conformer: Convolution-augmented Transformer for Speech Recognition

O *Conformer: Convolution-augmented Transformer for Speech Recognition* (Gulati et al. 2020) combina convoluções com Transformer para reconhecimento de fala, integrando a capacidade de extração de características locais das convoluções com o alcance global da *atenção* dos Transformers. Essa combinação mostra como técnicas complementares podem ser usadas para melhorar o desempenho de modelos em tarefas de sequência de áudio, oferecendo *insights* sobre a potencial integração de funções de agregação para melhorar a robustez do mecanismo de *atenção*.

2.4.4 Point Transformer

Em Point Transformer (Point Transformer) (Engel, Belagiannis e Dietmayer 2021), a atenção é aplicada ao contexto de nuvens de pontos tridimensionais. Este trabalho propõe uma adaptação do mecanismo de *atenção* para lidar com dados não estruturados, como nuvens de pontos, o que requer uma abordagem que considere a topologia e a distribuição espacial dos pontos. A relevância desse estudo reside na exploração de como a *atenção* pode ser ajustada para diferentes tipos de dados, oferecendo *insights* valiosos sobre a flexibilidade das funções de agregação no processamento de informações espaciais.

2.4.5 Multi-gate Attention Network for Image Captioning

Multi-gate Attention Network for Image Captioning (Jiang et al. 2021) apresenta uma rede de *atenção* que utiliza múltiplos *gates* para atribuir pesos diferenciados às regiões de uma imagem, melhorando a geração de legendas automáticas. A aplicação de múltiplos *gates* permite um controle mais detalhado sobre quais partes da imagem devem ser enfatizadas durante o processo de descrição. Esse método está alinhado com o objetivo de explorar como funções de agregação podem ajustar a forma como a *atenção* é distribuída, otimizando a representação de características visuais.

2.4.6 Sparse Self-Attention Transformer for Image Inpainting

O *Sparse Self-Attention Transformer for Image Inpainting* (Huang et al. 2024) explora uma versão esparsa da *autoatenção*, focando em melhorar a eficiência do processamento em tarefas de restauração de imagens. A *atenção* esparsa busca concentrar os recursos computacionais nas áreas mais relevantes da imagem, reduzindo a carga computacional em áreas menos significativas. Esse conceito de esparsidade é diretamente relevante para o estudo das funções de agregação, já que ambos os enfoques visam oti-

mizar a alocação de recursos computacionais para melhorar o desempenho do modelo em tarefas específicas.

2.4.7 Attention is All You Need in Speech Separation

Attention is All You Need in Speech Separation (Subakan et al. 2021) explora a separação de fala, utilizando o mecanismo de *atenção* para identificar e separar diferentes fontes sonoras em um sinal de áudio. Esse trabalho demonstra a eficácia dos Transformers em lidar com tarefas de separação de sinais complexos, onde a identificação precisa de padrões é crucial. A aplicação de mecanismos de *atenção* em um contexto de dados temporais oferece uma perspectiva complementar ao estudo, que explora o uso de funções de agregação para melhorar a precisão em diferentes domínios.

2.4.8 Transformer Tracking with Cyclic Shifting Window Attention

Transformer Tracking with Cyclic Shifting Window Attention (Song et al. 2022) propõe um método de rastreamento de objetos utilizando uma *atenção* em *janelas deslizantes cíclicas*. Essa abordagem permite que o modelo se concentre em diferentes partes de uma sequência de imagens ao longo do tempo, otimizando o desempenho em tarefas de rastreamento contínuo. A ideia de *atenção* em *janelas deslizantes* complementa o estudo das funções de agregação, pois ambas as técnicas buscam melhorar a eficiência do modelo ao ajustar a distribuição do foco de *atenção*.

2.4.9 TransCAM: Transformer Attention-based Cam Refinement for Weakly Supervised Semantic Segmentation

TransCAM: Transformer Attention-based Cam Refinement for Weakly Supervised Semantic Segmentation (R. Li et al. 2023) investiga o refinamento da *atenção* em tarefas de segmentação semântica, utilizando um modelo Transformer para melhorar a precisão na identificação de objetos em imagens. A abordagem proposta destaca a importância de um mecanismo de *atenção* ajustado para melhorar a qualidade da segmentação sem a necessidade de anotações detalhadas. A relação entre esse estudo e o uso de funções de agregação reside na busca por melhorar a seletividade e a precisão do foco de *atenção* em tarefas visuais complexas.

2.5 Resumo da Seção

Nesta seção, exploramos os principais avanços relacionados à aplicação da arquitetura Transformer, focando em como diferentes abordagens têm otimizado o mecanismo de *atenção* para atender a uma variedade de desafios em aprendizado de máquina. Desde aplicações em PLN até visão computacional, os artigos analisados refletem a versatilidade e o impacto dos Transformers em domínios diversos.

Começamos revisitando o conceito introdutório do mecanismo de *atenção* apresentado em *Attention is All You Need*, que revolucionou a arquitetura de modelos ao substituir redes convolucionais e recorrentes. Em seguida, destacamos avanços no PLN, como o BERT e XLNet, que introduziram abordagens bidirecionais e autoregressivas para capturar contextos mais ricos.

No contexto da visão computacional, o ViT marcou um divisor de águas ao demonstrar que Transformers podem superar redes convolucionais em tarefas de classifi-

cação e análise visual. Trabalhos como o DETR e DINO mostraram como a arquitetura pode ser adaptada para detecção de objetos e aprendizado auto-supervisionado, enquanto estudos como o *Refiner* e *TransCAM* investigaram ajustes específicos no mecanismo de *atenção* para melhorar a eficiência e precisão em tarefas visuais.

Trabalhos inovadores que combinaram Transformers com outras técnicas, como o *Conformer*, que integrou convoluções para enriquecer a análise de sequências temporais. Abordagens especializadas, como o *Point Transformer* e *Sparse Self-Attention*, exploraram variantes do mecanismo de *atenção* para lidar com dados tridimensionais e esparsos, ampliando ainda mais a aplicabilidade dos Transformers.

Esses avanços fornecem uma base sólida para explorar como a introdução de funções de agregação pode aprimorar o mecanismo de *atenção* dos Transformers. Ao integrar conceitos de (pré-)agregação e funções *fuzzy*, o estudo busca adaptar o foco de *atenção* de forma mais seletiva e eficiente, possibilitando ganhos em tarefas complexas que demandam maior adaptabilidade.

Esse levantamento embasa a nossa proposta, demonstrando que o campo possui lacunas e oportunidades de melhoria que podem ser exploradas através da introdução de novas funções de agregação nos modelos Transformers.

Este panorama destaca não apenas a evolução da arquitetura Transformer, mas também as lacunas que podem ser preenchidas ao incorporar novos mecanismos de *atenção*. Essa análise fundamenta a proposta deste trabalho, que visa expandir as aplicações dos Transformers por meio da introdução de funções de agregação, contribuindo para sua eficiência e precisão em cenários desafiadores.

3 Novos Métodos de *Self-Attention* Baseados em Integrais Fuzzy

A arquitetura Transformer (Subakan et al. 2021) revolucionou diversas áreas da IA ao introduzir o mecanismo de *self-attention*, uma abordagem capaz de identificar e priorizar informações relevantes em sequências de entrada. O sucesso dos Transformers está diretamente relacionado à sua habilidade de realizar agregações eficientes, como a *função softmax*, que transforma pesos em probabilidades normalizadas. No entanto, ao lidar com dados de alta dimensionalidade e padrões complexos, os métodos tradicionais de agregação apresentam limitações significativas, dificultando a captura de interações não lineares.

Para superar esses desafios, este capítulo propõe novos métodos de *self-attention* baseados em integrais *fuzzy*, especificamente utilizando as integrais de Choquet, Sugeno e a dCC-integral. Essas integrais permitem realizar agregações ponderadas e flexíveis, ampliando a capacidade de generalização e robustez dos Transformers, especialmente em tarefas complexas e de alto custo computacional. Cada método será detalhado em seções específicas, destacando suas respectivas vantagens e implementações.

3.1 Mecanismo de Atenção Baseado na Integral Discreta de Choquet

A **integral de Choquet** é uma ferramenta matemática poderosa para realizar agregações ponderadas e flexíveis, permitindo a modelagem de dependências complexas entre os elementos de entrada. Este método adapta o mecanismo de *self-attention* tradicional do Transformer, substituindo a *função de agregação* padrão pela **Integral de Choquet**.

3.1.1 Modificação do Mecanismo de Atenção

Identificamos dois momentos no cálculo do mecanismo de *self-attention* em que a **Integral Discreta de Choquet** pode substituir a função de agregação padrão:

- Na multiplicação de Q por K^T, ajustando diretamente os pesos de atenção.
- Entre a saída da *função softmax* e a matriz V, no instante de aplicação dos pesos normalizados.

Escolhemos a segunda opção, substituindo a multiplicação padrão por uma agregação *fuzzy* baseada na Integral discreta de Choquet. Essa escolha fundamenta-se na compatibilidade direta entre as probabilidades normalizadas pela *softmax*, que operam no intervalo $[0, 1]$, e o conceito de medida *fuzzy*. A nova formulação do mecanismo é descrita pela Equação 3.1:

$$\text{Attention}_{\text{Choquet}}(Q, K, V) = \mathfrak{C}_{\mu}(\text{softmax}(QK^T), V), \tag{3.1}$$

Essa substituição confere maior flexibilidade ao modelo, permitindo:

- **Captura de Interações Não Lineares:** A **Integral de Choquet** possibilita a modelagem de dependências complexas entre os elementos de entrada.

- **Robustez a Ruídos:** Reduz a sensibilidade do mecanismo a valores extremos, mantendo a consistência do modelo.
- **Melhoria da Generalização:** A flexibilidade da medida *fuzzy* permite a adaptação a diferentes padrões de dados.

3.1.2 Otimização do Algoritmo Baseado na Integral de Choquet

A introdução da **Integral de Choquet** no mecanismo de *self-attention* aumentou a complexidade computacional devido ao cálculo de agregações ponderadas. Para viabilizar seu uso prático, desenvolvemos um algoritmo otimizado que reduz o custo computacional explorando operações paralelas, aproveitando ao máximo o processamento em Graphics Processing Unit (GPU)s. A descrição detalhada desse algoritmo está apresentada no Algoritmo 13.

Antes de detalharmos a otimização, é fundamental compreender a estrutura subjacente que permite essa eficiência. Utilizamos a *matriz de produto* MV, que combina vetores de atenção estocásticos com vetores de valores, permitindo uma agregação ponderada eficiente através da **Integral de Choquet**.

Matriz de Produto MV e Suas Propriedades

Para implementar de forma eficiente a **Integral de Choquet** no mecanismo de *self-attention*, introduzimos a *matriz de produto* MV, definida conforme a Definição 6. Essa estrutura combina vetores de atenção estocásticos com vetores de valores, facilitando o cálculo da **Integral de Choquet** de maneira paralelizada.

Essa paralelização é obtida dividindo o problema em pequenos subproblemas sem dependência temporal e enviando-os simultaneamente aos núcleos de processamento

das GPUs, de modo a maximizar a utilização dos recursos de hardware e reduzir a latência de execução.

Definição 6 *Seja* $M = [m_{ij}] \in \mathcal{M}_{n\times n}([0,1])$ *uma matriz tal que cada linha é um vetor linha* i *estocástico* $\boldsymbol{m}_i = (m_{i,1}, \ldots, m_{i,n})$*, e* $V = [v_{kl}] \in \mathcal{M}_{n\times q}([0,1])$ *uma matriz onde cada coluna é um vetor coluna* l $\boldsymbol{v}^l = (v_{1,l}, \ldots, v_{n,l})^T$*, com* $l \in Q = \{1, \ldots, q\}$*. Para* M *e* V*, a capacidade da matriz produto* MV *é definida como a matriz*

$$MV = [\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}] \in \mathcal{M}_{n\times q}\left([2^N \to [0,1]]\right),$$

onde cada entrada $\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l} \colon 2^N \to [0,1]$ *é uma capacidade de vetor produto* $(\boldsymbol{m}_i, \boldsymbol{v}^l)$*-produto, para* $i \in N$ *e* $l \in Q$*.*

Propriedades da Matriz de Produto MV

A capacidade de vetor produto $(\boldsymbol{m}_i, \boldsymbol{v}^l)$ possui as seguintes propriedades, conforme a Proposição 7:

Definição 7 *A capacidade de vetor produto* $(\boldsymbol{m}_i, \boldsymbol{v}^l)$ *é:*

1. ***Aditiva****:* $\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X \cup Y) = \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X) + \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(Y)$*, para* $X \cap Y = \emptyset$*.*
2. ***Auto-conjugada****:* $\overline{\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}} = \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}$*.*
3. ***Não Simétrica****: Não satisfaz* $\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X) = \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(Y)$ *para todos* $\mid X \mid = \mid Y \mid$*.*
4. ***Não Maxitiva****:* $\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X \cup Y) \neq \max(\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X), \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(Y))$*, para* $X \cup Y \neq \emptyset$*.*
5. ***Não Minitiva****:* $\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X \cap Y) \neq \min(\mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(X), \mu_{\boldsymbol{m}_i}^{\boldsymbol{v}^l}(Y))$*, para* $X \cap Y \neq \emptyset$*.*

Essas propriedades asseguram que a *matriz de produto* MV permite uma agregação não linear e eficiente dos valores de atenção, essencial para a implementação otimizada do algoritmo.

Exemplo 1 *Considere as matrizes* $M \in \mathcal{M}_{3\times 3}([0,1])$ *e* $V \in \mathcal{M}_{3\times 4}([0,1])$ *definidas como:*

$$M = \begin{pmatrix} 0.2 & 0.2 & 0.6 \\ 0.25 & 0.25 & 0.5 \\ 0.7 & 0.1 & 0.2 \end{pmatrix}, \quad V = \begin{pmatrix} 0.8 & 0.22 & 0.45 & 0.15 \\ 0.2 & 0.15 & 0.62 & 0.9 \\ 0.75 & 0.38 & 0.9 & 0.1 \end{pmatrix}$$

Os vetores linha estocásticos são:

$$\boldsymbol{m}_1 = (0.2, 0.2, 0.6), \quad \boldsymbol{m}_2 = (0.25, 0.25, 0.5), \quad \boldsymbol{m}_3 = (0.7, 0.1, 0.2)$$

e os vetores coluna são:

$$\boldsymbol{v}^1 = (0.8, 0.2, 0.75)^T, \quad \boldsymbol{v}^2 = (0.22, 0.15, 0.38)^T,$$
$$\boldsymbol{v}^3 = (0.45, 0.62, 0.9)^T, \quad \boldsymbol{v}^4 = (0.15, 0.9, 0.1)^T.$$

A capacidade da matriz produto MV *é dada por:*

$$MV = \begin{pmatrix} \mu^{\boldsymbol{v}^1}_{\boldsymbol{m}_1} & \mu^{\boldsymbol{v}^2}_{\boldsymbol{m}_1} & \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_1} & \mu^{\boldsymbol{v}^4}_{\boldsymbol{m}_1} \\ \mu^{\boldsymbol{v}^1}_{\boldsymbol{m}_2} & \mu^{\boldsymbol{v}^2}_{\boldsymbol{m}_2} & \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2} & \mu^{\boldsymbol{v}^4}_{\boldsymbol{m}_2} \\ \mu^{\boldsymbol{v}^1}_{\boldsymbol{m}_3} & \mu^{\boldsymbol{v}^2}_{\boldsymbol{m}_3} & \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_3} & \mu^{\boldsymbol{v}^4}_{\boldsymbol{m}_3} \end{pmatrix}$$

Por exemplo, a capacidade $\mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}$ *é calculada conforme segue:*

$$\mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\emptyset) = 0, \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{1\}) = 0.25, \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{2\}) = 0.25, \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{3\}) = 0.50,$$
$$\mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{1,3\}) = 0.25 + 0.50 = 0.75, \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{1,2\}) = 0.25 + 0.25 = 0.50,$$
$$\mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{2,3\}) = 0.25 + 0.50 = 0.75, \mu^{\boldsymbol{v}^3}_{\boldsymbol{m}_2}(\{1,2,3\}) = 0.25 + 0.25 + 0.50 = 1.$$

Esses exemplos ilustram como a capacidade da *matriz produto* MV combina os pesos de atenção estocásticos com os valores correspondentes, facilitando o cálculo eficiente da **Integral de Choquet**.

3.1.3 Algoritmo Otimizado para Cálculo da Integral de Choquet

A seguir, apresentamos o Algoritmo 3.1, que implementa uma abordagem otimizada para calcular a **Integral Discreta de Choquet** baseada na *matriz* MV-produto. Este algoritmo aproveita a paralelização e as operações vetorizadas das GPUs para minimizar o tempo de execução.

Algoritmo 3.1: Cálculo da **Integral de Choquet**

Data: Matriz de saída da *Softmax* e tensor de entrada V

Result: Tensor de saída *tmp*

tmp ← tensor vazio
sorted_v, *index_v* ← Ordena V ao longo da penúltima dimensão
zeros_col ← Cria um tensor de zeros
v_with_zeros ← Concatena zeros e valores ordenados
v_difference ← Calcula diferenças consecutivas
for *i* ∈ *range(tamanho da última dimensão de v_difference)* **do**
    new_v ← Extrai valores e adiciona nova dimensão
    new_index_v ← Ajusta índices para alinhamento
    att_order ← Reorganiza pesos de atenção
    att_cumsum ← Calcula soma cumulativa invertida
    tmp ← Realiza produto escalar e concatena resultados
end
return *tmp*

Descrição do Algoritmo

O Algoritmo (13) descreve o processo otimizado para calcular a **Integral de Choquet** baseada na *matriz* MV-produto. As etapas principais incluem:

1. **Ordenação Paralela**: Os valores da *matriz* V são ordenados ao longo da penúltima dimensão de forma paralela, aproveitando a capacidade das GPUs de realizar operações simultâneas em grande escala.

2. **Cálculo de Diferenças Consecutivas**: Após a ordenação, calculamos as diferenças entre valores consecutivos para facilitar a aplicação da capacidade da *matriz produto*.

3. **Soma Cumulativa Invertida**: Realizamos a soma cumulativa invertida dos pesos de atenção, o que permite calcular a contribuição ponderada de cada valor de forma eficiente.

4. **Produto Escalar Vetorizado**: A realização do produto escalar entre os pesos de atenção e as diferenças calculadas é executada de maneira vetorizada, aproveitando as operações matriciais rápidas das GPUs.

5. **Concatenação de Resultados**: Os resultados são concatenados para formar o tensor de saída final *tmp*, representando a *matriz de atenção* baseada na **Integral de Choquet**.

Essas otimizações garantem que o cálculo da **Integral de Choquet** não se torne um gargalo no desempenho geral do modelo Transformer, mantendo a competitividade em termos de eficiência computacional.

3.1.4 Implementação e Otimizações Computacionais

A eficiência do Algoritmo (13) deriva da exploração de operações paralelas e da vetorização, características inerentes ao processamento em GPUs. As principais etapas otimizadas incluem:

1. **Ordenação Paralela**: A ordenação dos valores de V é realizada de forma paralela, aproveitando a capacidade das GPUs de lidar com grandes volumes de dados simultaneamente.

2. **Cálculo de Diferenças**: As diferenças consecutivas entre os valores ordenados são computadas de maneira vetorizada, eliminando a necessidade de iterações sequenciais.

3. **Soma Cumulativa Invertida**: A soma cumulativa é calculada de forma invertida para facilitar a aplicação da capacidade da *matriz produto* MV, permitindo que a **Integral de Choquet** seja computada eficientemente.

4. **Produto Escalar Vetorizado**: A realização do produto escalar entre os pesos de atenção e as diferenças calculadas é executada de forma vetorizada, aproveitando as operações matriciais rápidas das GPUs.

Essas otimizações asseguram que o cálculo da **Integral de Choquet** seja integrado ao mecanismo de *self-attention* sem comprometer a performance global do modelo.

3.1.5 Conclusão

O método de *self-attention* baseado na **Integral de Choquet** apresentado nesta seção destaca-se por sua capacidade de capturar interações não lineares e oferecer maior robustez em comparação aos métodos tradicionais. Através de uma formulação

matemática detalhada e da implementação de um algoritmo otimizado para GPUs, demonstramos a viabilidade prática da integração da **Integral de Choquet** no mecanismo de atenção dos Transformers. Os resultados iniciais indicam que o modelo proposto não apenas preserva a eficiência do Transformer, mas também amplia sua capacidade de generalização e robustez em tarefas complexas. A seguir, apresentamos métodos alternativos baseados em outras integrais *fuzzy*, ampliando ainda mais as possibilidades de aprimoramento dos mecanismos de *self-attention*.

3.2 Mecanismo de Atenção Baseado na Integral de Sugeno

A **Integral de Sugeno** é outra ferramenta matemática robusta para a realização de agregações *fuzzy*, destacando-se em cenários que requerem a priorização adaptativa de determinados critérios. Este método adapta o mecanismo de *self-attention* do Transformer, substituindo a *função de agregação* padrão pela **Integral de Sugeno**.

3.2.1 Modificação do Mecanismo de Atenção

Analisamos dois pontos principais no cálculo do mecanismo de *self-attention* onde a **Integral de Sugeno** pode ser aplicada para substituir a *função de agregação* padrão:

- Na multiplicação das matrizes Q e K^T, ajustando diretamente os pesos de atenção.
- Entre a saída da *função softmax* e a matriz V, no momento da aplicação dos pesos normalizados.

Optamos pela primeira abordagem, substituindo a multiplicação padrão entre Q e K^T pela **Integral de Sugeno**. Essa decisão permite incorporar diretamente a

priorização de critérios e a captura de relações não lineares entre os pesos Q e K. A formulação matemática do novo mecanismo é expressa na Equação 3.2:

$$\text{Attention}_{\text{Sugeno}}(Q, K, V) = \text{softmax}(Su^{F,G}(Q, K))\, V, \tag{3.2}$$

onde $Su^{F,G}$ representa a **Integral de Sugeno** aplicada sobre os pesos das matrizes Q e K. Essa substituição redefine o cálculo dos pesos de atenção, preservando a estrutura geral do Transformer.

A aplicação da **Integral de Sugeno** no mecanismo de *self-attention* traz benefícios específicos para cenários onde a priorização de critérios ou valores extremos é essencial. Destacamos as seguintes vantagens:

- **Priorização de Valores Significativos:** A **Integral de Sugeno** enfatiza os elementos mais relevantes no contexto, oferecendo maior eficiência para tarefas que exigem destaque de características específicas.
- **Decisões Hierárquicas e Interpretáveis:** Essa abordagem é especialmente útil para capturar a relevância relativa de múltiplos critérios, tornando os resultados mais interpretáveis, mesmo em sistemas complexos.
- **Flexibilidade e Adaptação:** A capacidade de trabalhar com critérios dinâmicos e ajustáveis torna o modelo ideal para aplicações em cenários mutáveis, como sistemas de recomendação ou detecção de anomalias.

Essas características posicionam a abordagem **Integral de Sugeno** como uma alternativa robusta ao modelo tradicional, especialmente em problemas que exigem um controle mais refinado sobre os critérios avaliados.

3.2.2 Considerações Finais

O método de *self-attention* baseado na **Integral de Sugeno** apresentado nesta seção destaca-se por sua capacidade de priorizar valores significativos e capturar relações hierárquicas de forma interpretável. A substituição da multiplicação padrão pela **Integral de Sugeno** permite que o mecanismo de atenção se adapte dinamicamente às características do dado de entrada, aumentando a flexibilidade e a robustez do modelo. Os experimentos preliminares indicam que este método é particularmente vantajoso em tarefas que requerem a priorização de valores extremos, como classificação hierárquica ou detecção de padrões específicos. A seguir, exploramos uma abordagem ainda mais inovadora baseada na dCC-integral, combinando diversas teorias *fuzzy* para aprimorar ainda mais o mecanismo de *self-attention*.

3.3 Mecanismo de Atenção Baseado na dCC-Integral

Além das **Integrais de Choquet** e **Integrais de Sugeno**, a dCC-integral representa uma abordagem avançada para a realização de agregações *fuzzy*, combinando a teoria de medidas *fuzzy*, *cópulas* e *funções de dissimilaridade restrita* (Restricted Dissimilarity Function (RDF)). Este método oferece uma solução mais adaptável e expressiva para o mecanismo de *self-attention* do Transformer.

3.3.1 Modificação do Mecanismo de Atenção com dCC-Integrals

Para implementar a dCC-integral no mecanismo de *self-attention*, realizamos uma análise detalhada do funcionamento interno dos Transformers, identificando opor-

tunidades para aprimorar o processo de agregação. Essa análise revelou que a operação tradicional de multiplicação de matrizes entre Q (*queries*) e K (*keys*) pode ser substituída por uma operação baseada na dCC-integral, com o objetivo de tornar o processo de atenção mais flexível e adaptável.

A modificação é composta por três etapas principais:

1. **Ordenação e Reordenação da *Matriz K*:** Para preservar a relação entre Q e K, a *matriz* K é reordenada com base nos índices derivados de Q. Isso garante consistência no processo de agregação.

2. **Transformação de *K* em uma Medida *Fuzzy*:** A *matriz* K é transformada em uma medida *fuzzy*, permitindo que a dCC-integral capture interações não lineares e variações de significância nos elementos de entrada.

3. **Cálculo dos Pesos de Atenção com a dCC-Integral:** Os pesos são calculados utilizando a dCC-integral, incorporando uma combinação de *cópulas* e RDFs para modelar relações complexas.

Definição da Medida *Fuzzy*

Para transformar K em uma medida *fuzzy*, aplicamos uma normalização dinâmica que ajusta os valores com base no intervalo dos elementos ordenados. A fórmula de normalização é dada por:

$$\text{FuzzyMeasure}(K) = \frac{K_{\text{sorted}} - \min(K_{\text{sorted}})}{\max(K_{\text{sorted}}) - \min(K_{\text{sorted}})}, \quad \forall\, K_{\text{sorted}} \tag{3.3}$$

Essa transformação ajusta os valores de K ao intervalo $[0, 1]$, preservando as propriedades relevantes do dado original e minimizando a influência de valores atípicos. A medida *fuzzy* resultante é então utilizada para definir o comportamento da dCC-integral.

Cálculo dos Pesos com a dCC-Integral

O cálculo dos pesos de atenção utiliza a dCC-integral, conforme descrito na Equação 3.4:

$$W = Q_{(1)} + \sum_{i=2}^{n-1} \delta\big(C(q_{(i)}, m_{(i)}), C(q_{(i-1)}, m_{(i)})\big) \tag{3.4}$$

Onde:

- W representa os pesos de atenção ajustados.
- $Q_{(1)}$ é o componente inicial do vetor de *queries*.
- δ é a RDF, que mede diferenças entre os componentes.
- C é a *função de cópula*, aplicada para capturar dependências entre Q e K.
- $m_{(i)}$ é a medida *fuzzy* derivada da *matriz* K.

Essa formulação permite que o mecanismo de atenção capture relações mais complexas e não lineares entre os elementos de entrada.

Novo Mecanismo de Atenção com a função dCC-Integral

Com base nos pesos calculados, o novo mecanismo de atenção pode ser formulado como na Equação 3.5:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(Q_{(1)} + \sum_{i=2}^{n-1} \delta\big(C(q_{(i)}, m_{(i)}), C(q_{(i-1)}, m_{(i)})\big)\right) V \tag{3.5}$$

Essa equação redefine o cálculo da **matriz de atenção** ao integrar a dCC-integral, mantendo a compatibilidade com as arquiteturas Transformers e expandindo sua capacidade de modelar interações complexas nos dados.

A introdução da dCC-integral no mecanismo de atenção traz benefícios significativos:

- **Modelagem Precisa de Relações Não Lineares:** A combinação de medidas *fuzzy*, *cópulas* e RDFs permite capturar padrões mais ricos e complexos nos dados.
- **Maior Flexibilidade:** A transformação adaptativa da *matriz* K garante que o mecanismo se ajuste dinamicamente às características do dado de entrada.
- **Redução de Sensibilidade a Outliers:** O processo de normalização minimiza a influência de valores extremos, tornando o modelo mais robusto.

3.3.2 Considerações Gerais

O método de *self-attention* baseado na dCC-integral apresentado nesta seção representa uma abordagem inovadora que combina medidas *fuzzy*, *cópulas* e *funções de dissimilaridade* para capturar relações não lineares e específicas nos dados de entrada. A modificação proposta permite que os Transformers ampliem sua capacidade de modelar padrões complexos, mantendo eficiência e escalabilidade. A integração da dCC-integral no mecanismo de atenção não apenas preserva a eficiência do Transformer, mas também aumenta sua robustez e flexibilidade em tarefas de alto nível. No próximo capítulo, realizaremos uma análise comparativa detalhada entre os mecanismos baseados nas **Integrais de Choquet**, **Integrais de Sugeno** e dCC, avaliando suas respectivas vantagens e limitações em diferentes cenários de aplicação.

Os métodos de *self-attention* baseados nas **Integrais de Choquet**, **Integrais de Sugeno** e dCC apresentam abordagens distintas para a agregação de informações, cada uma com suas próprias vantagens e aplicações específicas. A **Integral de Choquet** se destaca pela capacidade de modelar interações não lineares e robustez a ruídos, tornando-a adequada para tarefas complexas onde a generalização é crucial. A **Inte-**

gral de Sugeno, por sua vez, oferece uma priorização adaptativa de critérios, sendo ideal para cenários que requerem decisões hierárquicas e interpretáveis. Finalmente, a dCC-integral combina diversas teorias *fuzzy* para proporcionar uma solução ainda mais flexível e expressiva, capaz de capturar relações complexas e reduzir a sensibilidade a outliers.

A escolha entre esses métodos depende das características específicas da tarefa em questão e dos requisitos de desempenho e interpretabilidade. Em contextos onde a captura de interações não lineares e a robustez são prioritárias, a **Integral de Choquet** e a dCC-integral se mostram particularmente eficazes. Já em aplicações que exigem uma priorização clara de critérios ou a interpretação das decisões, a **Integral de Sugeno** oferece vantagens significativas.

A seguir, no próximo capítulo, será realizada uma análise comparativa detalhada entre esses mecanismos de *self-attention*, explorando suas respectivas vantagens, limitações e aplicações em diferentes domínios da IA.

4 Conjuntos de Dados

Neste capítulo, apresentamos os conjuntos de dados (*datasets*) utilizados para validar os modelos propostos, bem como os tratamentos aplicados a cada um deles. Os conjuntos de dados foram criteriosamente selecionados e categorizados em duas áreas distintas: um grupo voltado para a classificação de imagens e outro para a classificação de textos e movimentos. Essa divisão foi estrategicamente planejada para garantir uma validação robusta dos modelos em relação a diferentes funções de agregação, oferecendo uma avaliação mais abrangente e completa do desempenho dos métodos propostos em diversos contextos.

4.1 CIFAR-10

O CIFAR-10 (Krizhevsky, G. Hinton et al. 2009) é um dos conjuntos de dados mais populares na comunidade de aprendizado de máquina e visão computacional. Desenvolvido pelo *Canadian Institute for Advanced Research* (Canadian Institute For Advanced Research (CIFAR)), este conjunto é composto por imagens rotuladas que pertencem a dez classes distintas, incluindo aviões, automóveis, pássaros, gatos, veados, cachorros, sapos, cavalos, navios e caminhões, como ilustrado na Figura 4.1. Essas categorias foram escolhidas para representar uma variedade de objetos do cotidiano, tornando o CIFAR-10 um excelente ponto de partida para testar a capacidade dos modelos em reconhecer e classificar padrões visuais comuns.

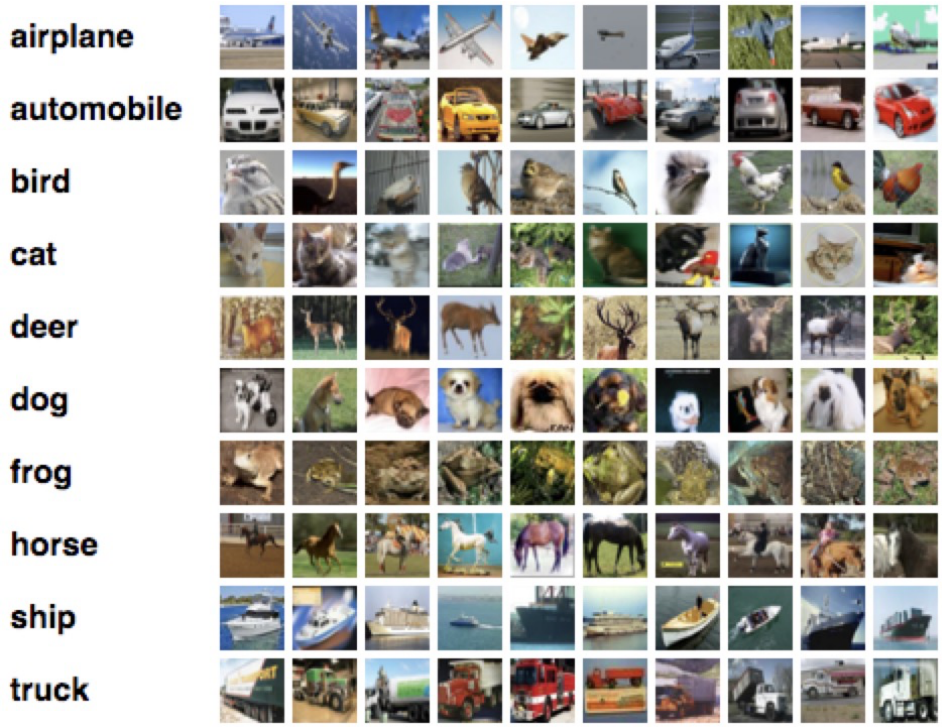

Figura 4.1: Imagem representando as classes do conjunto de dados CIFAR-10, conforme descrito em (Krizhevsky, G. Hinton et al. 2009).

Com um total de 60.000 imagens coloridas de 32×32 pixels, o CIFAR-10 é distribuído igualmente entre as dez classes, resultando em 6.000 imagens por classe. Esse conjunto é dividido em um subconjunto de treinamento com 50.000 imagens e um subconjunto de teste com 10.000 imagens. A simplicidade e a estrutura organizada do CIFAR-10 o tornam ideal para experimentar e validar novos métodos de classificação, como aqueles propostos por modelos baseados em funções de agregação FG-funcionais. Além disso, devido à sua ampla adoção em pesquisas anteriores, ele fornece uma base sólida para comparação de desempenho.

4.2 CIFAR-100

O CIFAR-100 (Krizhevsky, G. Hinton et al. 2009), também criado pelo *Canadian Institute for Advanced Research*, é uma versão mais complexa e desafiadora do

CIFAR-10. Ele contém 100 classes diferentes, cada uma composta por 600 imagens, totalizando também 60.000 imagens de 32×32 pixels. Essa maior variedade de classes exige dos modelos uma capacidade aumentada de distinguir entre um maior número de categorias, o que torna o CIFAR-100 uma excelente ferramenta para avaliar a capacidade dos modelos em lidar com problemas de classificação em cenários mais complexos e diversificados.

As classes no CIFAR-100 são agrupadas em duas superclasses principais: "coisas"e "animais". A categoria "coisas"abrange objetos inanimados e diversos fenômenos, enquanto "animais"engloba seres vivos. Cada uma dessas superclasses contém 20 subclasses, como flores, árvores, insetos, mamíferos e peixes. Esse nível de detalhamento permite avaliar a eficácia dos modelos em capturar nuances e diferenças sutis entre categorias semelhantes, um aspecto fundamental para validar o uso de funções de agregação avançadas.

4.3 CALTECH-101

O CALTECH-101 (F.-F. Li et al. 2022) é outro conjunto de dados essencial em pesquisas de visão computacional e aprendizado de máquina, conhecido por sua capacidade de desafiar os modelos de classificação de objetos. Este conjunto é composto por 9.146 imagens coloridas, distribuídas em 101 categorias distintas, que incluem uma ampla gama de objetos, como animais, veículos, utensílios domésticos e elementos naturais. A Figura 4.2 ilustra a diversidade presente neste conjunto de dados.

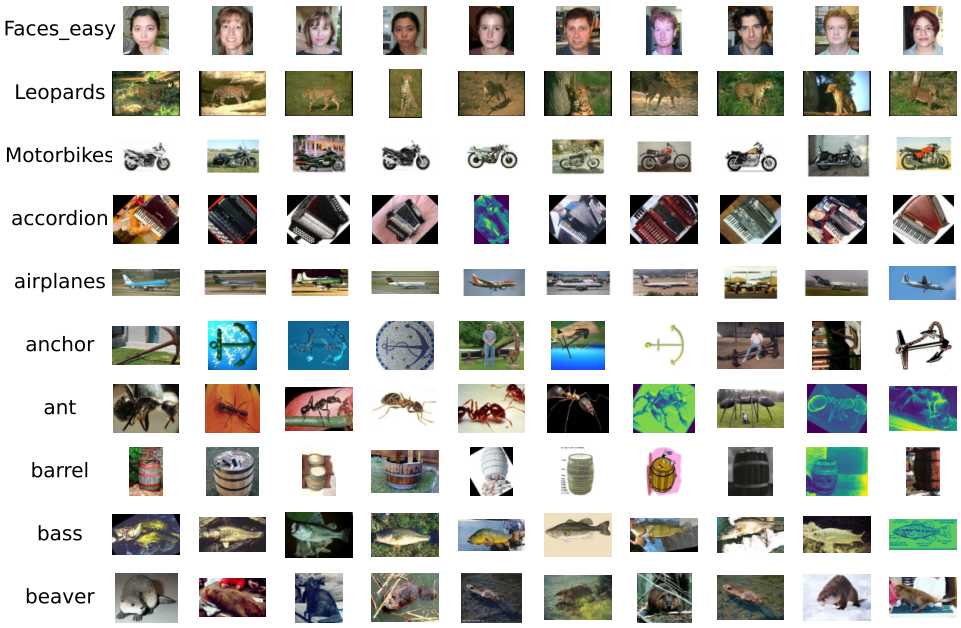

Figura 4.2: Imagem representando as classes do conjunto de dados CALTECH-101. (F.-F. Li et al. 2022)

O CALTECH-101 é especialmente valioso para avaliar a robustez dos modelos em condições variáveis, como diferentes iluminações, escalas, fundos e poses dos objetos. Isso o torna um desafio significativo para qualquer modelo de aprendizado de máquina. A utilização deste conjunto de dados é essencial para testar a capacidade dos modelos baseados em FG-funcionais de generalizar para ambientes visuais dinâmicos e realistas, indo além das condições controladas dos conjuntos de dados mais simples.

4.4 COCO

O COCO (Lin et al. 2014) é um dos conjuntos de dados mais complexos e amplamente utilizados em tarefas de visão computacional, como detecção de objetos, segmentação de imagens e geração de legendas automáticas. Desenvolvido pela Microsoft Research, o COCO contém mais de 200.000 imagens rotuladas manualmente, abrangendo mais de 80 classes de objetos. Essas classes incluem desde pessoas e animais até veículos e eletrodomésticos, conforme ilustrado na Figura 4.3. A complexidade e a vari-

edade do COCO o tornam ideal para treinar e testar modelos em ambientes que exigem reconhecimento detalhado de múltiplos objetos em uma única imagem.

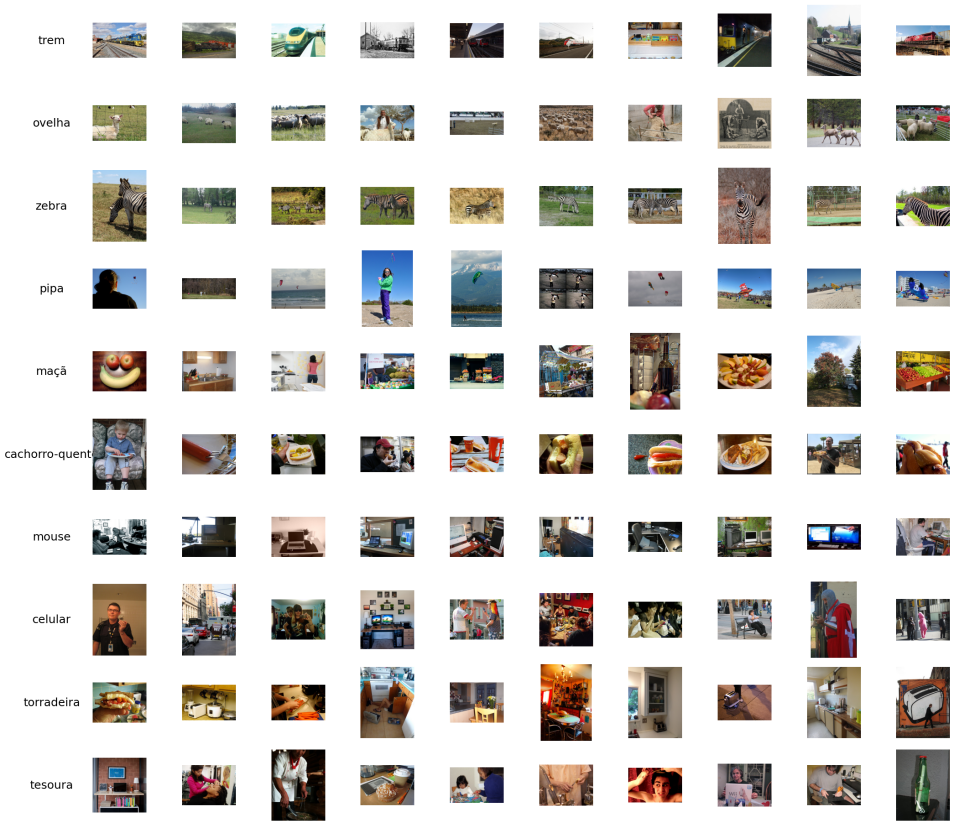

Figura 4.3: Exemplos de imagens do COCO Dataset, mostrando a variedade de classes e a complexidade dos cenários. (Lin et al. 2014)

O COCO se destaca por fornecer múltiplas anotações por imagem, incluindo caixas delimitadoras, máscaras de segmentação, pontos de articulação para poses humanas e descrições textuais. Esse nível de detalhamento é fundamental para avaliar a capacidade dos modelos de capturar interações complexas entre objetos e entender contextos diversificados. Ao usar o COCO, podemos testar a eficácia dos novos mecanismos de *self-attention* baseados em funções FG-funcionais para lidar com tarefas que envolvem a detecção e classificação de múltiplos objetos simultaneamente, proporcionando insights valiosos sobre a generalização e aplicabilidade dos modelos em cenários do mundo real.

5 Experimentos e Resultados

Neste capítulo, apresentamos uma análise detalhada dos experimentos realizados com o novo conjunto de mecanismos de autoatenção introduzidos no Capítulo 3, aplicados aos modelos Transformers e ViT. O objetivo principal desses experimentos foi avaliar como diferentes combinações de *funções de agregação* influenciam o desempenho dos modelos em tarefas de classificação, tanto no domínio temporal quanto no espacial.

Para os modelos Transformers, selecionamos a tarefa de reconhecimento de linguagem gestual, utilizando séries temporais de pontos de referência da postura corporal. Essa tarefa permite explorar a capacidade dos modelos em capturar as sutilezas dos movimentos humanos, que envolvem sequências complexas e dinâmicas. Já para os modelos ViT, focamos na tarefa de classificação de imagens, buscando avaliar o impacto das novas *funções de agregação* na capacidade de reconhecimento visual em cenários de diferentes complexidades e variações contextuais. Com essa abordagem, foi possível analisar o efeito das *funções de agregação* nos mecanismos de autoatenção em contextos distintos, evidenciando a versatilidade e a eficácia dos métodos propostos.

Utilizamos as arquiteturas padrão dos modelos Transformers (Vaswani et al. 2017) e ViT (Dosovitskiy et al. 2020) como referências para comparação. A partir delas, desenvolvemos versões modificadas dessas arquiteturas, nas quais incorporamos diferentes *funções de agregação* aos mecanismos de autoatenção, buscando aprimorar a representação das informações e a capacidade de aprendizado dos modelos.

Para assegurar consistência e reprodutibilidade nos experimentos, adotamos

uma configuração de hiperparâmetros fixa, conforme apresentada na Tabela 5.1. Os hiperparâmetros foram cuidadosamente selecionados para garantir o melhor desempenho possível dos modelos em ambas as tarefas. O *embedding dropout* foi configurado em 0,1 para ambos os modelos, auxiliando na prevenção de *overfitting* ao desativar aleatoriamente uma fração das unidades de *embedding* durante o treinamento. O número de **épocas** foi estabelecido em 100 para os Transformers e 50 para os ViT, refletindo as diferentes complexidades e tempos de convergência das tarefas. Ajustamos o **batch size** para 128 nos Transformers e 256 nos ViT, otimizando o uso de memória e a eficiência computacional durante o treinamento.

As taxas de aprendizado (*learning rates*) foram definidas como 5×10^{-4} para Transformer e 1×10^{-3} para ViT, valores que equilibram a velocidade de convergência e a estabilidade dos modelos durante o treinamento. O número de cabeças de atenção (**número de cabeças**) foi configurado para 4 nos Transformers e 8 nos ViT, permitindo que os ViT capturem representações mais diversas, o que é especialmente útil para a análise de imagens complexas. A dimensão das cabeças de atenção (**dimensão das cabeças**) foi ajustada para 256 nos Transformers e 64 nos ViT, considerando a necessidade de maior capacidade de processamento temporal nos Transformers.

Além disso, utilizamos uma **seed** fixa (1601) para a inicialização dos pesos, o que ajuda a reduzir a variabilidade entre execuções e garante a comparabilidade dos resultados. Todos os experimentos foram conduzidos no Google Colaboratory (Google Colab) (Google Colab 2024), utilizando o framework Python Torch Framework (PyTorch) (Paszke et al. 2019) e aproveitando a infraestrutura de GPUs para acelerar o processo de treinamento, o que foi crucial para lidar com a complexidade computacional das redes neurais profundas.

Tabela 5.1: Hiperparâmetros utilizados nos experimentos

Hiperparâmetros	**Transformers**	**ViT**
Embedding Dropout	0.1	0.1
Épocas	100	50
Batch Size	128	256
Taxa de Aprendizado	5×10^{-4}	1×10^{-3}
Número de Cabeças	4	8
Seed	1601	1601
Dimensão das Cabeças	256	64

5.1 Experimentos e Avaliação de Desempenho

Para validar as novas configurações de autoatenção propostas, realizamos uma série de experimentos utilizando diferentes conjuntos de dados. Nosso foco foi avaliar a eficácia das *funções de agregação* propostas em tarefas de classificação em domínios variados, tanto espaciais quanto temporais. Essa abordagem permitiu verificar a robustez e a capacidade de generalização dos modelos propostos, bem como identificar possíveis limitações e áreas para melhorias futuras.

As métricas de avaliação incluíram *acurácia* e *taxa de erro*, fornecendo uma análise abrangente do impacto das novas *funções de agregação* na melhoria das representações geradas. Além disso, analisamos as *curvas de aprendizado* para entender melhor o comportamento dos modelos em diferentes etapas do treinamento. Os resultados demonstraram o potencial das *funções de agregação* baseadas em FG-funcionais para aprimorar significativamente o desempenho de modelos de autoatenção em cenários complexos, sugerindo que essas novas abordagens podem ser vantajosas em diversas áreas da IA, como PLN, visão computacional e análise de séries temporais.

5.2 Experimentos com Vision Transformers (ViT)

Nesta etapa do trabalho, conduzimos uma análise aprofundada dos modelos ViTs aplicados a diferentes conjuntos de dados: CIFAR-10 (Krizhevsky, G. Hinton et al. 2009), CIFAR-100 (Krizhevsky, G. Hinton et al. 2009), COCO (Lin et al. 2014) e Caltech101 (F.-F. Li et al. 2022). A seleção desses *datasets* visou avaliar o desempenho dos modelos em variados níveis de complexidade, abrangendo desde cenários mais simples, como o CIFAR-10, até contextos mais desafiadores e representativos do mundo real, como o COCO.

A utilização de múltiplos conjuntos de dados permitiu uma análise comparativa que considera aspectos cruciais, como a variabilidade das imagens, o tamanho do *dataset* e a diversidade de classes. Por exemplo, o CIFAR-10 e o CIFAR-100 consistem em imagens de baixa resolução, oferecendo um ponto de partida adequado para avaliar a capacidade de generalização dos modelos em cenários controlados. Em contraste, o COCO, com sua alta variabilidade e complexidade, e o Caltech101, com classes mais balanceadas e uma representação visual mais rica, desafiam os modelos a capturar padrões mais complexos e intrincados, exigindo maior capacidade de abstração e generalização.

Durante os experimentos, consideramos cuidadosamente as limitações do poder computacional disponível, o que impactou diretamente a seleção dos hiperparâmetros e o tempo de treinamento. Optamos por um número de **épocas** que equilibrasse a necessidade de convergência do modelo com o tempo disponível para os experimentos. Os resultados obtidos e as análises realizadas fornecem insights valiosos para pesquisadores interessados na aplicação de modelos ViT em diversos contextos práticos, desde cenários de visão computacional controlados até aplicações mais complexas e realistas.

5.2.1 Resultados no CIFAR-10

Os experimentos iniciais no *dataset* CIFAR-10, composto por 60.000 imagens coloridas de 32×32 pixels distribuídas em dez classes distintas, proporcionaram insights significativos sobre o impacto das *funções de agregação* nos modelos ViT. A Figura 5.1 ilustra a evolução da *acurácia* durante o treinamento para cada *função de agregação* avaliada.

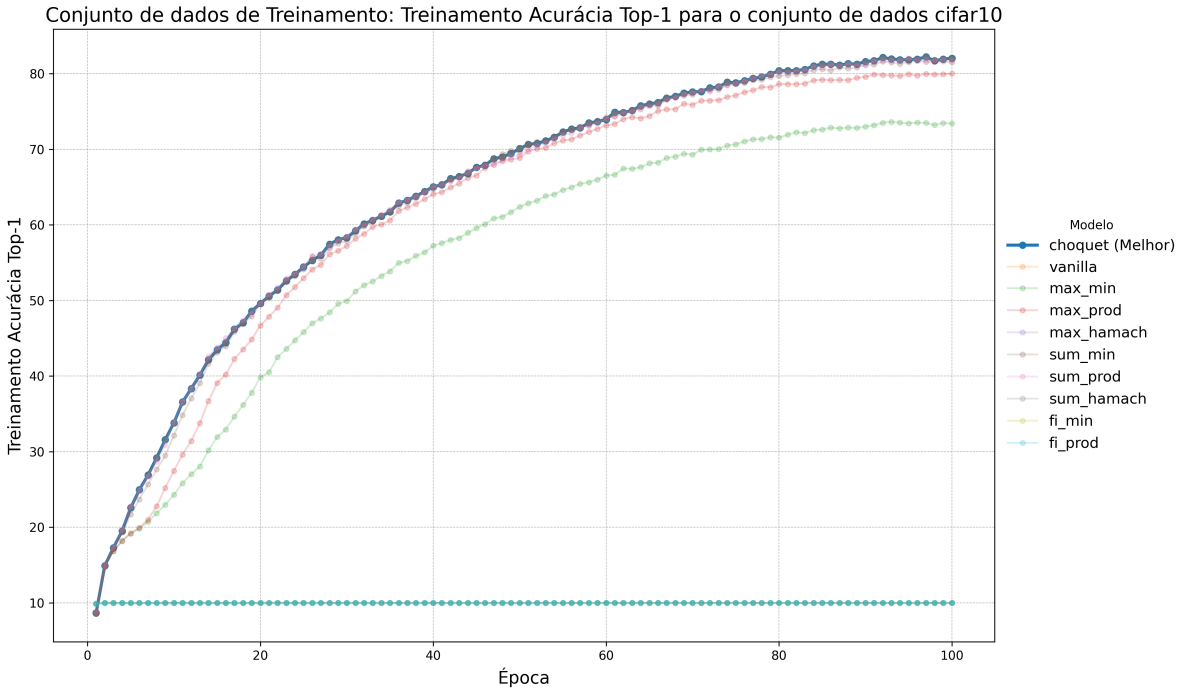

Figura 5.1: Evolução da *acurácia* no treinamento com o *dataset* CIFAR-10, conforme ilustrado na Figura 4.1.

Observou-se que a **Choquet** alcançou a maior *acurácia* no conjunto de treinamento, seguida de perto pelas funções **Vanilla** e **Max(Min)**. Esses resultados sugerem que a **Choquet** é particularmente eficaz na captura de padrões relevantes durante o processo de aprendizagem, o que é fundamental para assegurar um desempenho robusto em tarefas de classificação.

A **Choquet** distingue-se por sua capacidade de modelar interações complexas entre variáveis, permitindo um balanceamento adaptativo entre os valores agregados.

Esse comportamento é análogo à operação lógica “OR”, onde maior peso é atribuído às entradas mais significativas, potencializando a resposta do modelo às características mais relevantes dos dados. Em contraste, funções como **Max(Min)** adotam uma abordagem mais restritiva, similar à operação lógica “AND”, exigindo que todas as entradas sejam significativas para maximizar a resposta. Embora essa abordagem possa limitar a flexibilidade do modelo, ela pode conferir maior robustez em cenários onde a consistência entre as entradas é essencial.

A Figura 5.2 apresenta a *acurácia* no conjunto de validação. Destaca-se que a **Sum(Min)** obteve o melhor desempenho nesse conjunto, indicando uma excelente capacidade de generalização. A **Sum(Min)** pode ser interpretada como uma combinação flexível entre as operações lógicas “AND” e “OR”, onde a soma permite um acúmulo progressivo de informações, mas respeita os limites impostos pelos valores mínimos, assegurando que as características críticas sejam consideradas.

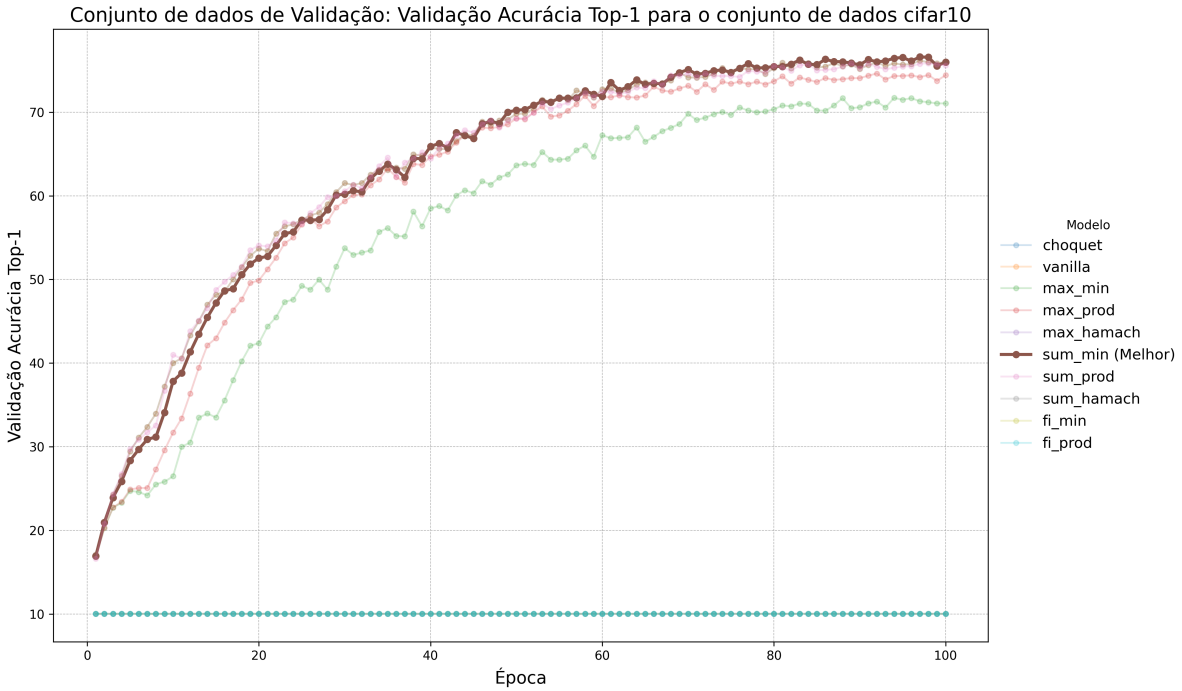

Figura 5.2: Evolução da *acurácia* na validação com o *dataset* CIFAR-10.

Um aspecto notável é o desempenho da **Fi(Prod)**, que apresentou *acurácia*

estagnada desde o início do treinamento. Esse comportamento pode ser atribuído ao fenômeno do desvanecimento de gradiente (*vanishing gradient*), no qual os gradientes calculados durante a retropropagação tornam-se muito pequenos, impedindo atualizações eficazes dos pesos. Esse problema é comum em redes profundas e em funções de ativação que comprimem os gradientes em intervalos limitados, resultando em um treinamento ineficiente.

Esses resultados ressaltam a importância da escolha adequada da *função de agregação*, que deve equilibrar a complexidade e a capacidade de generalização do modelo. *Funções* que operam de forma semelhante à lógica "AND" podem ser mais seletivas, garantindo consistência em contextos onde todas as entradas são relevantes. Por outro lado, *funções* que atuam como uma lógica "OR" podem ser mais tolerantes a variações, capturando melhor as interações complexas presentes nos dados.

5.2.2 Resultados no CIFAR-100

O conjunto de dados CIFAR-100, uma versão mais complexa do CIFAR-10 com 100 classes distintas, apresentou desafios adicionais aos modelos testados. Conforme ilustrado na Figura 5.3, a **Choquet** novamente demonstrou desempenho superior no conjunto de treinamento, embora a diferença entre as *funções* tenha sido menos pronunciada em comparação com o CIFAR-10. Isso sugere que, à medida que a complexidade do problema aumenta, as *funções de agregação* tendem a apresentar eficácia semelhante, desde que adequadamente configuradas.

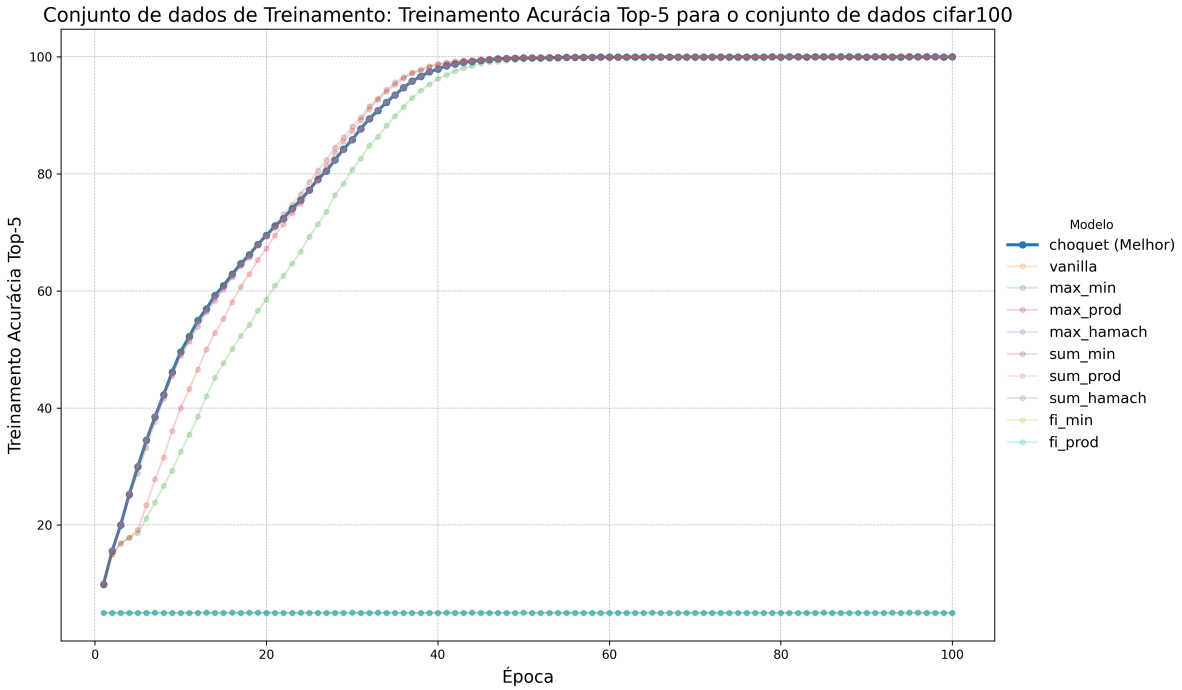

Figura 5.3: Evolução da *acurácia* no treinamento com o *dataset* CIFAR-100.

No conjunto de validação, representado na Figura 5.4, observou-se uma leve mudança no comportamento, com a **Sum(Prod)** alcançando os melhores resultados, enquanto **Choquet** e **Vanilla** mantiveram desempenhos comparáveis. A **Sum(Prod)** destaca-se por sua abordagem que combina características das lógicas "AND" e "OR", permitindo que a multiplicidade dos elementos influencie o resultado de forma acumulativa e assegurando que todas as contribuições sejam consideradas de maneira relevante.

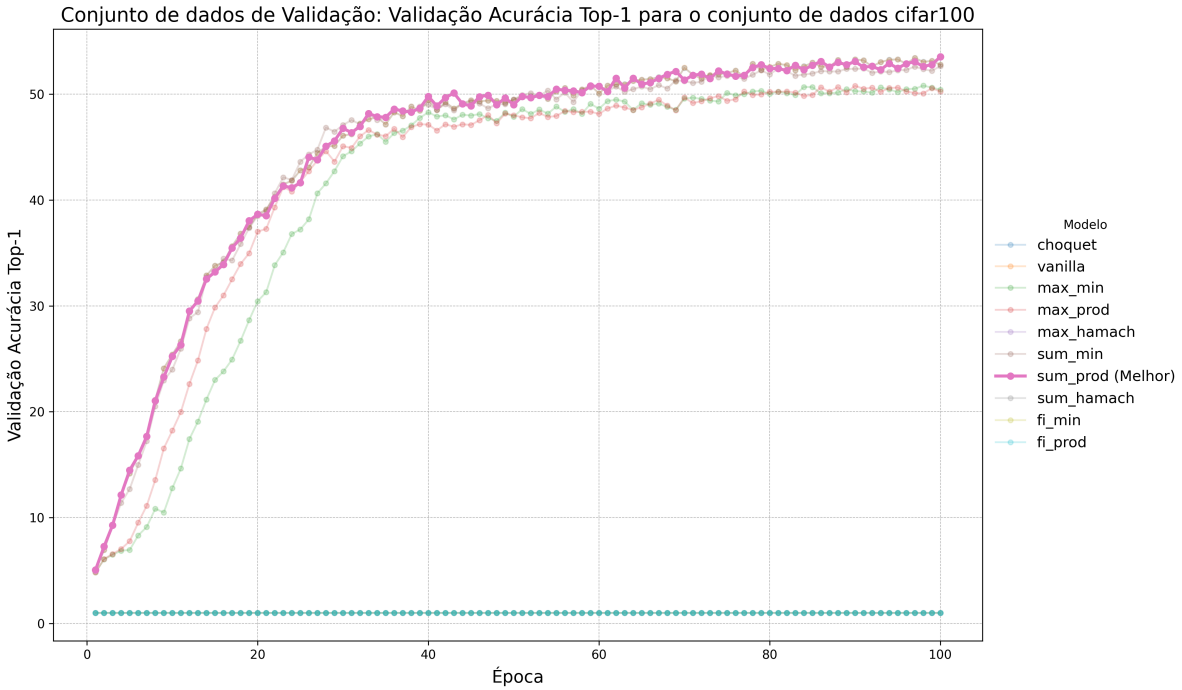

Figura 5.4: Evolução da *acurácia* na validação com o *dataset* CIFAR-100.

Esses resultados reforçam a importância de selecionar *funções de agregação* que equilibrem a capacidade de capturar interações complexas (características de uma lógica "OR") e a robustez de considerar todas as contribuições relevantes (semelhante a uma lógica "AND"). Em ambientes de maior complexidade, como o CIFAR-100, esse equilíbrio torna-se ainda mais crítico para o desempenho do modelo.

5.2.3 Resultados no Caltech101

Os experimentos realizados com o *dataset* Caltech101, que contém imagens coloridas de diferentes resoluções distribuídas em 101 classes, forneceram insights importantes sobre a influência das *funções de agregação* em cenários de alta variabilidade visual. Esse *dataset* representa um desafio significativo para os modelos, exigindo a captura de padrões detalhados e complexos.

A **Max(Min)** destacou-se ao alcançar os melhores resultados no conjunto de

treinamento, conforme ilustrado na Figura 5.5. Essa *função de agregação*, que pode ser comparada à operação lógica "AND" por exigir uma contribuição significativa de todas as entradas para maximizar a resposta, mostrou-se eficaz em cenários onde a consistência entre as variáveis é essencial.

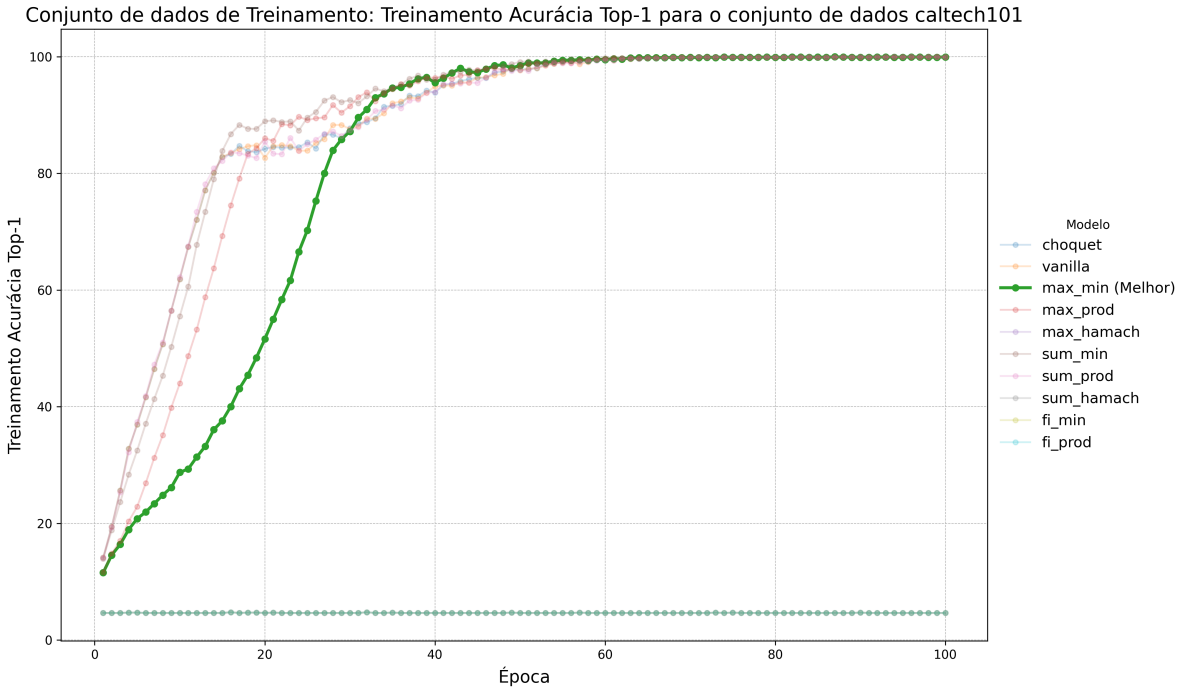

Figura 5.5: Evolução da *acurácia* no treinamento com o *dataset* Caltech101.

No conjunto de validação, a **Max(Min)** manteve seu desempenho superior, conforme evidenciado na Figura 5.6. A consistência dessa *função* durante o treinamento e a validação sugere que ela é capaz de manter a robustez em cenários complexos, ao considerar somente as entradas mais relevantes e exigir que todas contribuam significativamente para o resultado final.

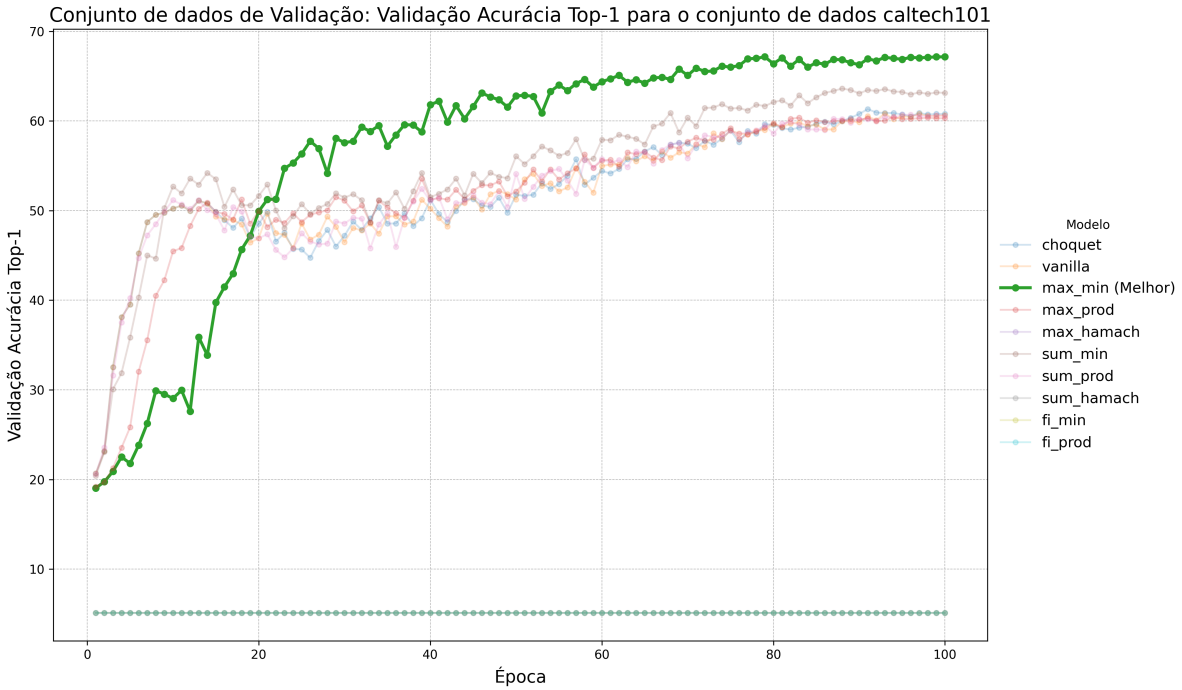

Figura 5.6: Evolução da *acurácia* na validação com o *dataset* Caltech101.

Esses resultados indicam que, em ambientes de alta complexidade como o Caltech101, *funções de agregação* que seguem uma lógica mais seletiva e restritiva, como **Max(Min)**, podem oferecer vantagens em termos de precisão e consistência. Isso ocorre porque tais *funções* garantem que somente as características mais relevantes e consistentes sejam consideradas pelo modelo, reduzindo a influência de ruídos e variações indesejadas.

5.2.4 Resultados no COCO

Os experimentos com o *dataset* COCO proporcionaram insights valiosos sobre o desempenho dos modelos ViT em um ambiente altamente complexo e desafiador. O COCO é conhecido por sua alta variabilidade e complexidade, devido à presença de múltiplas classes e objetos em uma única imagem, o que exige que os modelos capturem padrões intrincados e generalizem bem em condições adversas.

A **Sum(Min)** destacou-se com os melhores resultados no conjunto de treinamento, conforme ilustrado na Figura 5.7. Essa *função de agregação* combina características das operações lógicas "AND" e "OR", acumulando informações de forma progressiva enquanto respeita os limites impostos pelos valores mínimos. Isso assegura que o modelo mantenha uma abordagem equilibrada durante o aprendizado, considerando tanto a relevância individual das características quanto sua combinação.

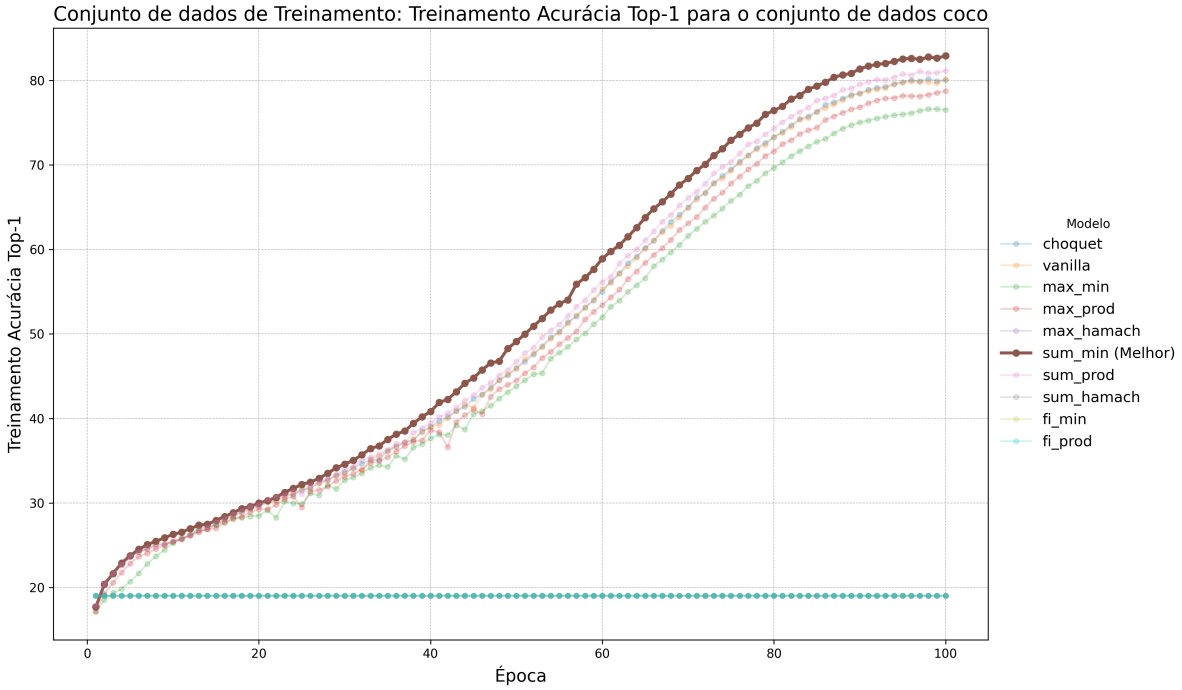

Figura 5.7: Evolução da *acurácia* no treinamento com o *dataset* COCO.

No conjunto de validação, representado na Figura 5.8, a **Choquet** apresentou desempenho superior a partir da vigésima época. Essa *função de agregação*, que modela interações complexas e permite um ajuste mais adaptativo das contribuições, é comparável a uma operação lógica "OR", favorecendo entradas com maior impacto e assegurando uma generalização eficaz em ambientes desafiadores. A consistência da **Choquet** durante o treinamento e a validação demonstra sua robustez na captura de relações complexas entre os dados.

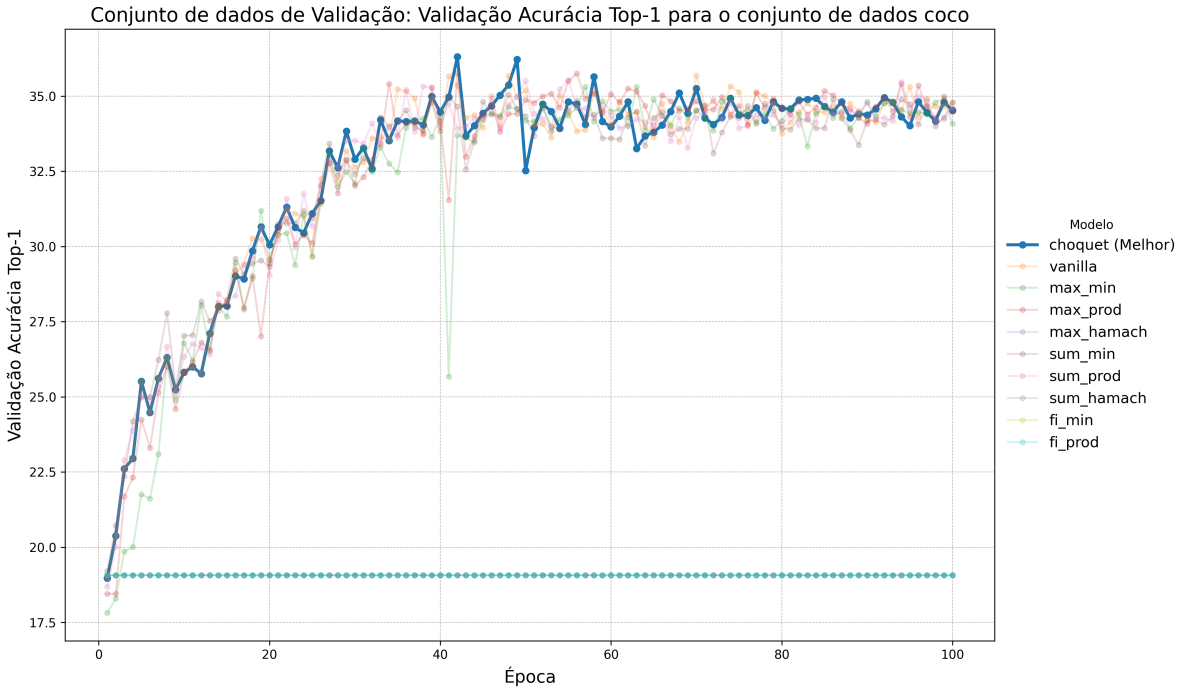

Figura 5.8: Evolução da *acurácia* na validação com o *dataset* COCO.

Esses resultados sugerem que, em ambientes de alta complexidade como o COCO, a seleção de *funções de agregação* adaptativas, como **Choquet**, pode oferecer vantagens substanciais em termos de precisão e capacidade de capturar interações de alto impacto. Em contraste, *funções* mais seletivas como **Sum(Min)** garantem uma abordagem mais rigorosa durante o treinamento, mas podem não generalizar tão bem em cenários de validação com alta variabilidade, onde a flexibilidade para capturar múltiplas características relevantes é essencial.

5.3 Análise Comparativa dos Experimentos

A Tabela 5.2 resume os melhores resultados obtidos com diferentes *funções de agregação* nos modelos ViT para cada um dos *datasets* avaliados. Observa-se que, embora a **Vanilla** tenha apresentado desempenho sólido no CIFAR-10, as diferenças em relação às demais *funções de agregação* foram pequenas em todos os *datasets*. Isso sugere que

funções alternativas podem oferecer desempenho competitivo, trazendo benefícios adicionais, como menor custo computacional ou melhor adaptação a certos tipos de dados, dependendo da aplicação.

Dataset	Função de Agregação	Última Perda (Treino)	Última Acurácia 1 (Treino)	Última Perda (Val)	Última Acurácia 1 (Val)
caltech101	**Choquet**	0.99	99.96	2.75	60.78
caltech101	**Fi_Min**	NaN	NaN	4.65	5.11
caltech101	**Fi_Prod**	NaN	NaN	4.65	5.11
caltech101	**Max_Hamach**	NaN	NaN	4.65	5.11
caltech101	**Max(Min)**	1.00	99.96	2.47	67.15
caltech101	**Max_Prod**	0.99	99.96	2.71	60.30
caltech101	**Sum_Hamach**	NaN	NaN	4.65	5.11
caltech101	**Sum(Min)**	0.98	99.95	2.61	63.13
caltech101	**Sum_Prod**	0.99	99.93	2.75	60.58
caltech101	**Vanilla**	0.99	99.96	2.74	60.63
cifar10	**Choquet**	1.36	82.04	1.55	76.01
cifar10	**Fi_Min**	NaN	NaN	10.00	10.02
cifar10	**Fi_Prod**	NaN	NaN	10.00	10.02
cifar10	**Max_Hamach**	NaN	NaN	10.00	10.02
cifar10	**Max(Min)**	1.57	73.39	1.63	71.05
cifar10	**Max_Prod**	1.41	79.98	1.57	74.42
cifar10	**Sum_Hamach**	NaN	NaN	10.00	10.02
cifar10	**Sum(Min)**	1.37	81.50	1.53	75.99
cifar10	**Sum_Prod**	1.36	81.69	1.55	75.61
cifar10	**Vanilla**	1.36	82.04	1.55	76.01
cifar100	**Choquet**	1.03	99.89	2.92	52.79
cifar100	**Fi_Min**	NaN	NaN	1.00	0.99
cifar100	**Fi_Prod**	NaN	NaN	1.00	0.99
cifar100	**Max_Hamach**	NaN	NaN	1.00	0.99
cifar100	**Max(Min)**	1.06	99.77	2.99	50.44
cifar100	**Max_Prod**	1.03	99.89	3.04	50.26
cifar100	**Sum_Hamach**	NaN	NaN	1.00	0.99
cifar100	**Sum(Min)**	1.03	99.91	2.92	52.65
cifar100	**Sum_Prod**	1.03	99.91	2.91	53.53
cifar100	**Vanilla**	1.03	99.89	2.92	52.79
coco	**Choquet**	1.57	80.04	3.57	34.53
coco	**Fi_Min**	NaN	NaN	19.02	9.07
coco	**Fi_Prod**	NaN	NaN	19.02	19.07
coco	**Max_Hamach**	NaN	NaN	19.02	19.07
coco	**Max(Min)**	1.67	76.52	3.52	34.08
coco	**Max_Prod**	1.61	78.73	3.58	34.47
coco	**Sum_Hamach**	NaN	NaN	19.02	19.07
coco	**Sum(Min)**	1.51	82.90	3.53	34.79
coco	**Sum_Prod**	1.55	81.13	3.55	34.61
coco	**Vanilla**	1.57	80.14	3.58	34.79

Tabela 5.2: Resultados de desempenho de modelos ViT em diferentes *datasets*.

A análise comparativa dos resultados revela que a escolha da *função de agregação* pode influenciar significativamente o desempenho dos modelos ViT, especialmente em *datasets* de maior complexidade. *Funções* como **Choquet** e **Max(Min)** demonstraram

vantagens específicas em determinados contextos, sugerindo que a seleção da *função de agregação* deve considerar as características do *dataset* e os objetivos específicos da aplicação.

Além disso, os resultados indicam que, embora a **Vanilla** seja uma escolha sólida, alternativas como **Choquet** podem oferecer melhor desempenho em cenários que exigem a captura de interações complexas entre características. Por outro lado, *funções* mais seletivas como **Max(Min)** podem ser preferíveis quando a consistência entre todas as entradas é crucial para o desempenho do modelo.

5.4 Análise Estatística

Para avaliar a significância das diferenças de desempenho entre os modelos com funções de agregação alternativas e o modelo **Vanilla**, aplicamos o teste de Wilcoxon signed-rank pareado, adequado para amostras pareadas e distribuições não paramétricas. Adotou-se nível de significância $\alpha = 0.05$.

Tabela 5.3: Resultados do teste de Wilcoxon pareado: estatística W, p-valor e indicação de significância ($\alpha = 0.05$)

Dataset	**Função de Agregação**	**W (Wilcoxon)**	**p-valor**	**Significância ($\alpha = 0.05$)**
Caltech101	**Max(Min)**	1244.0	$1{,}06 \times 10^{-5}$	Sim
CIFAR-10	**Choquet**	1845.0	0.655	Não
CIFAR-100	**Sum(Prod)**	1456.0	0.0561	Não
COCO	**Sum(Min)**	1723.0	0.122	Não

Conforme a Tabela 5.3, apenas a função **Max(Min)** no Caltech101 produziu um p-valor muito inferior a α, indicando que a melhoria observada não é atribuível ao acaso. Para os demais *datasets*, como $p > 0.05$, não rejeitamos a hipótese nula de que as diferenças em relação à **Vanilla** são nulas, sugerindo desempenho estatisticamente equivalente. Esses achados ressaltam a importância de selecionar a função de agregação

de forma adequada às características específicas de cada *dataset* e da tarefa para o modelo ViT.

5.5 Considerações Finais

Os experimentos realizados evidenciam a relevância da escolha da *função de agregação* em modelos ViT, impactando diretamente a capacidade de generalização e a precisão em diferentes tarefas de classificação. A compreensão aprofundada das características de cada *função* e sua adequação ao contexto específico do *dataset* é essencial para otimizar o desempenho do modelo.

Futuras pesquisas podem explorar a combinação de múltiplas *funções de agregação* ou o desenvolvimento de novas *funções* que capturem de forma mais eficaz as complexidades inerentes a *datasets* desafiadores. Além disso, a investigação sobre o impacto do ajuste de hiperparâmetros associados às *funções de agregação* pode fornecer insights adicionais para aprimorar a performance dos modelos ViT em diversas aplicações.

5.6 Conclusões dos resultados

Os experimentos realizados demonstram que as *funções de agregação* impactam significativamente o desempenho dos modelos de autoatenção em tarefas de classificação de imagens e séries temporais. A **Vanilla** destacou-se pela consistência e capacidade de generalização em diferentes cenários, sendo uma escolha confiável. Entretanto, *funções* alternativas como **Sum(Min)** e **Max(Min)** também apresentaram resultados competitivos, indicando que podem ser úteis em contextos específicos, especialmente quando se busca menor custo computacional ou sensibilidade a padrões complexos.

Esses achados abrem caminho para futuras pesquisas, onde análises mais aprofundadas dos custos computacionais e da eficiência de cada *função* poderão fornecer

insights valiosos para a comunidade científica. Além disso, a exploração de novas *funções de agregação* e a integração com outras técnicas de aprendizado de máquina podem potencialmente levar a avanços significativos no campo de modelos baseados em autoatenção.

6 Conclusão

Neste estudo, investigamos alternativas inovadoras para o mecanismo de autoatenção dos modelos Transformers, introduzindo diferentes **funções de agregação**, conhecidas como FG-funcionais, para aprimorar a capacidade dos modelos em capturar dependências complexas e sutis em sequências de dados. Nosso objetivo foi avaliar como essas **funções de agregação** poderiam melhorar a representatividade e a eficácia dos modelos em contextos variados, incluindo tanto dados espaciais quanto temporais. Através de uma série de experimentos rigorosos, aplicando os modelos a tarefas de classificação de imagens e reconhecimento de linguagem gestual, adquirimos insights valiosos sobre o impacto dessas **funções** na melhoria da performance dos mecanismos de autoatenção.

Os resultados demonstraram que **funções de agregação** específicas, especialmente aquelas que replicam o comportamento de operadores lógicos, como a FG de soma (semelhante ao operador lógico OR) e a FG de produto (semelhante ao operador lógico AND), oferecem vantagens significativas dependendo do contexto de aplicação. A FG de soma mostrou-se particularmente eficaz em capturar variações amplas e em lidar com interações menos restritivas entre os elementos dos dados, sendo adequada para tarefas onde a diversidade de padrões é crítica. Em contraste, a FG de produto destacou-se por sua capacidade de identificar interações mais específicas e restritivas, mostrando-se vantajosa em cenários que exigem uma análise detalhada e seletiva das relações entre os dados.

Apesar de a **função de atenção vanilla** ter mantido uma performance consistente e elevada em diversos cenários, confirmando-se como uma escolha robusta para tarefas gerais de classificação, variações como a **soma dos mínimos** apresentaram resultados igualmente competitivos em situações mais complexas, como na análise de séries temporais para reconhecimento de linguagem de sinais. Esses achados sugerem que a seleção da **função de agregação** deve ser cuidadosamente adaptada ao tipo específico de problema, abrindo espaço para a personalização de modelos Transformers e ViTs com base nas características dos dados e nas necessidades particulares das aplicações.

Durante o desenvolvimento deste trabalho, foram publicados dois estudos significativos. O primeiro, apresentado no Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI) 2023, concentrou-se na substituição da multiplicação de matrizes tradicional por **funções de agregação** alternativas, visando aprimorar a eficiência e a precisão em tarefas de classificação de imagens e reconhecimento de linguagem gestual. O segundo, apresentado na Federation of Software and Advanced Technology (FSTA) 2024, explorou estratégias avançadas de agregação, como as **integrais de Choquet** adaptadas, para otimizar o mecanismo de autoatenção em Transformers. Além disso, estudos complementares abordando técnicas de agregação *fuzzy*, como relatado em Autor(es) 2024, e na *FSTA 2024 Book of Abstracts* 2024, reforçam a relevância e aplicabilidade das propostas deste trabalho.

Essas contribuições evidenciam como novas **funções de agregação** podem potencializar o desempenho dos modelos Transformers, melhorando sua capacidade de captar interações complexas nos dados, com aplicação em áreas como visão computacional e PLN.

Gostaria de expressar minha profunda gratidão ao grupo de pesquisa Grupo de Informática (GINFO) (`http://www.ginfo.c3.furg.br`) pelo suporte essencial ao longo desta jornada. As discussões nas reuniões semanais e o apoio técnico contínuo foram cru-

ciais para o desenvolvimento e consolidação dos conceitos e experimentos apresentados nesta dissertação. Essas interações colaborativas foram fundamentais para superar desafios e gerar novas perspectivas que enriqueceram significativamente este trabalho.

Com base nos avanços teóricos e experimentais apresentados, este estudo oferece uma base sólida para futuras investigações em **funções de agregação** no contexto de arquiteturas Transformers, destacando seu potencial para transformar abordagens em IA e Deep Learning (DL).

Bibliografia

Abdi, Hervé, Dominique Valentin e Betty Edelman (1999). *Neural networks*. 124. Sage (ver p. 3).

Alahmadi, Dimah, Arwa Wali e Sarah Alzahrani (2022). "TAAM: Topic-aware abstractive arabic text summarisation using deep recurrent neural networks". Em: *Journal of King Saud University-Computer and Information Sciences* 34.6, pp. 2651–2665 (ver p. 4).

Autor(es) (2024). "Título do artigo". Em: *Título do livro (Springer)*. Springer, pp. XXX–XXX. DOI: 10.1007/978-3-031-45368-7_16. URL: https://link.springer.com/chapter/10.1007/978-3-031-45368-7_16 (ver p. 78).

Bardozzo, Francesco, Borja De La Osa, Ľubomíra Horanská, Javier Fumanal-Idocin, Mattia delli Priscoli, Luigi Troiano, Roberto Tagliaferri, Javier Fernandez e Humberto Bustince (2021). "Sugeno integral generalization applied to improve adaptive image binarization". Em: *Information Fusion* 68, pp. 37–45. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2020.10.020. URL: https://www.sciencedirect.com/science/article/pii/S1566253520304012 (ver p. 23).

Beliakov, Gleb, Ana Pradera, Tomasa Calvo et al. (2007). *Aggregation functions: A guide for practitioners*. Vol. 221. Springer (ver pp. 6, 17).

Bustince, Humberto, Jose Antonio Sanz, Giancarlo Lucca, Graçaliz P Dimuro, Benjamin Bedregal, Radko Mesiar, Anna Kolesárová e Gustavo Ochoa (2016). "Pre-aggregation functions: definition, properties and construction methods". Em: *2016 IEEE Inter-*

national Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, pp. 294–300 (ver pp. 7, 16).

Carion, Nicolas, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov e Sergey Zagoruyko (2020). "End-to-end object detection with transformers". Em: *European conference on computer vision*. Springer, pp. 213–229 (ver p. 34).

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski e Armand Joulin (2021). "Emerging properties in self-supervised vision transformers". Em: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660 (ver p. 34).

Choquet, Gustave (1954). "Theory of capacities". Em: *Annales de l'institut Fourier*. Vol. 5, pp. 131–295 (ver pp. 7, 16, 21).

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". Em: *Mathematics of control, signals and systems* 2.4, pp. 303–314 (ver p. 25).

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le e Ruslan Salakhutdinov (2019). "Transformer-xl: Attentive language models beyond a fixed-length context". Em: *arXiv preprint arXiv:1901.02860* (ver p. 31).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee e Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". Em: *arXiv preprint arXiv:1810.04805* (ver p. 28).

Di Gangi, Mattia A, Matteo Negri e Marco Turchi (2019). "Adapting transformer to end-to-end spoken language translation". Em: *Proceedings of INTERSPEECH 2019*. International Speech Communication Association (ISCA), pp. 1133–1137 (ver p. 3).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly et al. (2020). "An image is worth 16x16 words: Transformers for image recognition at scale". Em: *arXiv preprint arXiv:2010.11929* (ver pp. 8, 27, 31, 60).

Engel, Nico, Vasileios Belagiannis e Klaus Dietmayer (2021). “Point transformer”. Em: *IEEE access* 9, pp. 134826–134840 (ver p. 36).

FSTA 2024 Book of Abstracts (2024). Página 43. URL: https://www.fsta.sk/materials/fsta2024bookofabstracts.pdf#page=43 (ver p. 78).

Ganea, Octavian, Gary Bécigneul e Thomas Hofmann (2018). “Hyperbolic neural networks”. Em: *Advances in neural information processing systems* 31 (ver p. 25).

Gao, Bolin e Lacra Pavel (2017). “On the properties of the softmax function with application in game theory and reinforcement learning”. Em: *arXiv preprint arXiv:1704.00805* (ver p. 16).

Ghosh-Dastidar, Samanwoy e Hojjat Adeli (2009). “Spiking neural networks”. Em: *International journal of neural systems* 19.04, pp. 295–308 (ver p. 23).

Golestani, Negar e Mahta Moghaddam (2020). “Human activity recognition using magnetic induction-based motion signals and deep recurrent neural networks”. Em: *Nature communications* 11.1, p. 1551 (ver p. 5).

Google Colab (2024). *Google Colab*. URL: https://colab.research.google.com/ (ver p. 61).

Goyal, Rupali, Parteek Kumar e VP Singh (2024). “Automated question and answer generation from texts using text-to-text transformers”. Em: *Arabian Journal for Science and Engineering* 49.3, pp. 3027–3041 (ver p. 5).

Grabisch, M., J. Marichal, R. Mesiar e E. Pap (2009). *Aggregation Functions*. Cambridge: Cambridge University Press (ver pp. 16–18).

Grabisch, Michel (2016). *Set Functions, Games and Capacities in Decision Making*. Vol. 46. Theory and Decision Library C. Cham: Springer (ver p. 18).

Gulati, Anmol, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu et al. (2020). “Conformer:

Convolution-augmented transformer for speech recognition". Em: *arXiv preprint arXiv:2005.08100* (ver p. 35).

Hewamalage, Hansika, Christoph Bergmeir e Kasun Bandara (2021). "Recurrent neural networks for time series forecasting: Current status and future directions". Em: *International Journal of Forecasting* 37.1, pp. 388–427 (ver p. 5).

Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". Em: *IEEE Signal processing magazine* 29.6, pp. 82–97 (ver p. 3).

Hong, Danfeng, Zhu Han, Jing Yao, Lianru Gao, Bing Zhang, Antonio Plaza e Jocelyn Chanussot (2021). "SpectralFormer: Rethinking hyperspectral image classification with transformers". Em: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–15 (ver p. 5).

Huang, Wenli, Ye Deng, Siqi Hui, Yang Wu, Sanping Zhou e Jinjun Wang (2024). "Sparse self-attention transformer for image inpainting". Em: *Pattern Recognition* 145, p. 109897 (ver p. 36).

Jelodar, Hamed, Yongli Wang, Rita Orji e Shucheng Huang (2020). "Deep sentiment classification and topic discovery on novel coronavirus or COVID-19 online discussions: NLP using LSTM recurrent neural network approach". Em: *IEEE Journal of Biomedical and Health Informatics* 24.10, pp. 2733–2742 (ver p. 5).

Jiang, Weitao, Xiying Li, Haifeng Hu, Qiang Lu e Bohong Liu (2021). "Multi-gate attention network for image captioning". Em: *IEEE Access* 9, pp. 69700–69709 (ver p. 36).

Jing, Yongcheng, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu e Mingli Song (2019). "Neural style transfer: A review". Em: *IEEE transactions on visualization and computer graphics* 26.11, pp. 3365–3385 (ver p. 3).

Junior, Joelson S, Giancarlo Lucca, Diego Bottero, Graçaliz P Dimuro e Helida Santos (2023). "Revisao sistematica das teorias norteadoras do Visual Transformers". Em: *Anais do VII Workshop-Escola de Informática Teórica*. SBC, pp. 87–94 (ver p. 28).

Khandelwal, Urvashi, Kevin Clark, Dan Jurafsky e Lukasz Kaiser (2019). "Sample efficient text summarization using a single pre-trained transformer". Em: *arXiv preprint arXiv:1905.08836* (ver p. 5).

Krizhevsky, Alex, Geoffrey Hinton et al. (2009). "Learning multiple layers of features from tiny images". Em: (ver pp. 9, 55, 56, 63).

Kshatri, Sapna Singh e Deepak Singh (2023). "Convolutional neural network in medical image analysis: a review". Em: *Archives of Computational Methods in Engineering* 30.4, pp. 2793–2810 (ver p. 4).

LeCun, Yann, Léon Bottou, Yoshua Bengio e Patrick Haffner (1998). "Gradient-based learning applied to document recognition". Em: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (ver pp. 3, 4, 16).

Li, Fei-Fei, Marco Andreeto, Marc'Aurelio Ranzato e Pietro Perona (abr. de 2022). *Caltech 101*. DOI: 10.22002/D1.20086 (ver pp. 9, 57, 58, 63).

Li, Ruiwen, Zheda Mai, Zhibo Zhang, Jongseong Jang e Scott Sanner (2023). "Transcam: Transformer attention-based cam refinement for weakly supervised semantic segmentation". Em: *Journal of Visual Communication and Image Representation* 92, p. 103800 (ver p. 38).

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common Objects in Context". Em: *European Conference on Computer Vision (ECCV)*. URL: https://arxiv.org/abs/1405.0312 (ver pp. 9, 58, 59, 63).

Liu, Peng-ran, Lin Lu, Jia-yao Zhang, Tong-tong Huo, Song-xiang Liu e Zhe-wei Ye (2021). "Application of artificial intelligence in medicine: an overview". Em: *Current Medical Science* 41.6, pp. 1105–1115 (ver p. 3).

Murofushi, T. e M. Sugeno (1991). "Fuzzy t-conorm integral with respect to fuzzy measures: Generalization of Sugeno integral and Choquet integral". eng. Em: *Fuzzy sets and systems* 42.1, pp. 57–71. ISSN: 0165-0114 (ver pp. 16, 23).

Ouyang, Fan e Pengcheng Jiao (2021). "Artificial intelligence in education: The three paradigms". Em: *Computers and Education: Artificial Intelligence* 2, p. 100020 (ver p. 3).

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". Em: *Advances in Neural Information Processing Systems* 32, pp. 8024–8035 (ver p. 61).

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever et al. (2019). "Language models are unsupervised multitask learners". Em: *OpenAI blog* 1.8, p. 9 (ver p. 32).

Rumelhart, David E, Geoffrey E Hinton e Ronald J Williams (1986). "Learning representations by back-propagating errors". Em: *nature* 323.6088, pp. 533–536 (ver pp. 3, 4).

Russell, Stuart J (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc. (ver p. 3).

Song, Zikai, Junqing Yu, Yi-Ping Phoebe Chen e Wei Yang (2022). "Transformer tracking with cyclic shifting window attention". Em: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8791–8800 (ver p. 37).

Subakan, Cem, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi e Jianyuan Zhong (2021). "Attention is all you need in speech separation". Em: *ICASSP 2021-2021*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 21–25 (ver pp. 37, 40).

Sugeno, Michio (1974). "Theory of fuzzy integrals and its applications". Em: *Doctoral Thesis, Tokyo Institute of Technology* (ver p. 18).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin (2017). "Attention is all you need". Em: *Advances in neural information processing systems* 30 (ver pp. 5, 8, 27, 60).

Vydana, Hari Krishna, Martin Karafiát, Katerina Zmolikova, Lukáš Burget e Honza Černockỳ (2021). "Jointly trained transformers models for spoken language translation". Em: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7513–7517 (ver p. 5).

Wilkin, Tim e Gleb Beliakov (2015). "Weakly monotone aggregation functions". Em: *International Journal of Intelligent Systems* 30, pp. 144–169 (ver p. 16).

Wu, Shang-Lin, Yu-Ting Liu, Tsung-Yu Hsieh, Yang-Yin Lin, Chih-Yu Chen, Chun-Hsiang Chuang e Chin-Teng Lin (2017). "Fuzzy Integral With Particle Swarm Optimization for a Motor-Imagery-Based Brain–Computer Interface". eng. Em: *IEEE transactions on fuzzy systems* 25.1, pp. 21–28. ISSN: 1063-6706 (ver p. 16).

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov e Quoc V Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". Em: *Advances in neural information processing systems* 32 (ver p. 30).

Zadeh, L. A. (1965). "Fuzzy Sets". Em: *Information and Control* 8.3, pp. 338–353 (ver p. 16).

Zhang, Jun e Kim-Fung Man (1998). "Time series prediction using RNN in multi-dimension embedding phase space". Em: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*. Vol. 2. IEEE, pp. 1868–1873 (ver p. 3).

Zhao, Jinghua, Dalin Zeng, Shuang Liang, Huilin Kang e Qinming Liu (2021). “Prediction model for stock price trend based on recurrent neural network”. Em: *Journal of Ambient Intelligence and Humanized Computing* 12, pp. 745–753 (ver p. 5).