UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Brenno Cavalcanti Curtolo

Otimização do Vision Transformer no CIFAR-100: Avaliação em Ambientes Computacionais Distintos e Estratégias de Aprendizado

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Brenno Cavalcanti Curtolo

Otimização do Vision Transformer no CIFAR-100: Avaliação em Ambientes Computacionais Distintos e Estratégias de Aprendizado

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientadora: Profª Drª Rita Maria da Silva Julia

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Brenno Cavalcanti Curtolo

Otimização do Vision Transformer no CIFAR-100: Avaliação em Ambientes Computacionais Distintos e Estratégias de Aprendizado

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 19 de setembro de 2025:

Profª Drª Rita Maria da Silva Julia
Orientadora

Professor

Professor

Uberlândia, Brasil
2025

Resumo

O presente trabalho tem por objetivo principal estudar e analisar, por meio de experimentos, o funcionamento do modelo *Vision Transformer* (ViT), uma arquitetura de aprendizado profundo originalmente proposta para tarefas de visão computacional e que, neste projeto, foi aplicada ao processamento e classificação de imagens do conjunto de dados *CIFAR-100*. Neste contexto, a proposta busca reproduzir e expandir os experimentos de um trabalho base de modo a avaliar estratégias que mantenham o melhor nível de desempenho possível mesmo em cenários de recursos limitados. Assim, no desenvolvimento deste trabalho, foram exploradas diferentes configurações de execução e técnicas de otimização, incluindo políticas dinâmicas de *learning rate*, ajustes de hiperparâmetros e métodos de regularização, com o objetivo de melhorar a acurácia, acelerar a convergência e ampliar a capacidade de generalização do modelo. Com isso, este estudo contribui ao sistematizar escolhas que permitem treinar o ViT de forma eficaz sem necessidade de pré-treinamento em bases massivas, aproximando a prática acadêmica das condições reais de muitos contextos de pesquisa.

Palavras-chave: Vision Transformer. CIFAR-100. Aprendizado profundo. Learning rate. Otimização de hiperparâmetros. Regularização.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AM	*Aprendizado de Máquina*
CLS	*Class Token*
CIFAR	*Canadian Institute for Advanced Research*
CNN	*Convolutional Neural Network*
GPU	*Unidade de Processamento Gráfico*
MLP	*Multilayer Perceptron*
OOM	*Out of Memory*
PMC	Perceptron Multicamadas
RNN	*Recurrent Neural Network*
ViT	*Vision Transformer*

Sumário

1 Introdução

A tecnologia está presente em diversas atividades cotidianas e setores inteiros da sociedade, impulsionando soluções cada vez mais eficientes e acessíveis. Nos últimos anos, um vetor central desse avanço tem sido o Aprendizado de Máquina (AM), ramo da Inteligência Artificial em que modelos computacionais aprendem padrões a partir de dados e os aplicam de forma autônoma a novos problemas. Dentro desse campo, o Aprendizado Profundo consolidou-se como paradigma dominante ao empregar redes neurais profundas capazes de extrair representações hierárquicas e altamente expressivas, conquistando resultados de ponta em visão computacional, linguagem e fala (GOODFELLOW; BENGIO; COURVILLE, 2016).

Após ciclos de amadurecimento, os *Transformers* emergiram como marco conceitual ao modelar dependências de longo alcance via autoatenção, com paralelismo massivo e sem necessidade de recorrência ou convoluções (VASWANI et al., 2017). Adaptado ao domínio visual, o *Vision Transformer* (ViT) trata a imagem como uma sequência de *patches* e aplica autoatenção para capturar relações globais entre regiões, tradicionalmente focadas em padrões locais (DOSOVITSKIY et al., 2020). Essa mudança de perspectiva abriu caminho para investigações sobre estabilidade, generalização e custo computacional do ViT em cenários realistas.

Nesse cenário, a presente proposta se inspira no estudo-base de (SILVA, 2024), o qual implementa o ViT com hiperparâmetros fixos e taxa de aprendizado (*learning rate*) estático para conduzir investigações sobre estratégias de treinamento e regularização, buscando um equilíbrio entre desempenho, estabilidade e custo computacional. Mais especificamente, este trabalho estende o estudo-base mencionado ao estudar o ViT aplicado ao benchmark *CIFAR-100* (KRIZHEVSKY; HINTON, 2009), que é um contexto propositalmente desafiador, onde tem-se a combinação entre muita variabilidade intraclasse e pouca informação por imagem, o que exige que o modelo generalize bem mesmo diante de sinais visuais limitados.

Assim, foram feitos experimentos que comparam diferentes ambientes de execução entre CPUs e GPUs e exploram ajustes de treinamento e regularização que a literatura aponta como promissores, sempre com foco em soluções reprodutíveis e compatíveis com as limitações práticas de hardware. A intenção não é esgotar o tema, mas organizar um caminho de investigação claro, que auxilie na compreensão do comportamento do ViT nesse regime e ofereça subsídios para escolhas informadas em trabalhos futuros.

Como contribuição, este estudo apresenta um protocolo simples e fundamentado para treinar o ViT sem recorrer a pré-treinamento em bases massivas, discute implicações

de escolhas de treinamento em termos de estabilidade e generalização e reúne evidências empíricas úteis para orientar decisões em contextos de recursos limitados. Em linhas gerais, busca-se aproximar a pesquisa das condições reais em que muitos projetos são conduzidos, conciliando rigor experimental e viabilidade prática.

1.1 Motivação

As Transformers consolidaram-se como paradigma dominante em tarefas de visão, mas muitos estudos partem de pré-treinamento em bases massivas e de recursos computacionais pouco acessíveis. Na prática acadêmica e em ambientes profissionais com restrições de hardware, é comum treinar modelos do zero (sem pré-treinamento) em bases menores e com máquinas modestas. Nesse cenário, o *Vision Transformer* (ViT) aplicado ao *CIFAR-100* impõe desafios particulares: alta variabilidade intra-classe, baixa resolução das imagens e grande número de categorias, o que tensiona estabilidade, custo e capacidade de generalização.

Este trabalho, que busca aprimorar e tornar mais eficiente o estudo-base desenvolvido por (SILVA, 2024), é motivado pela necessidade de compreender o impacto real do ambiente de execução no tempo por época e na qualidade do treinamento, sistematizar escolhas que estabilizem o ViT sem pré-treinamento e mensurar, de forma reprodutível, os efeitos de hiperparâmetros e de técnicas adicionais sobre métricas estatísticas de avaliação. Soma-se a isso o interesse em superar as limitações observadas no estudo-base, demonstrando que mesmo em um ambiente computacional modesto é possível alcançar ganhos relevantes de desempenho e generalização. Em síntese, busca-se reduzir a distância entre a literatura de ponta e as condições operacionais típicas, oferecendo um protocolo prático e reprodutível para treinar o ViT no *CIFAR-100* sob restrições reais de recursos.

1.2 Objetivos

O objetivo geral deste trabalho é investigar, sob restrições realistas de computação, estratégias de treinamento e otimização do *Vision Transformer* (ViT) no *CIFAR-100*, buscando melhorar o desempenho em termos de *acurácia*, *top-5 accuracy* e *F1-macro*, aumentar a estabilidade do treinamento e otimizar a eficiência temporal.

Para atingir esse propósito, foram definidos objetivos específicos:

- Comparar o treinamento em diferentes ambientes de execução, analisando CPUs de gerações distintas (i5 e i7) e uma GPU de entrada (GTX 1050 Ti, 4 GB), de modo a quantificar o impacto do hardware no tempo por época e nas limitações práticas do pipeline.

- Investigar políticas de *learning rate*, incluindo valor fixo, *Cosine Annealing* e a combinação *warmup* + *Cosine Annealing*, com o intuito de identificar a abordagem mais estável e eficaz.
- Mapear a sensibilidade do ViT a hiperparâmetros fundamentais, como a penalização por peso, o tamanho do Lote, a dimensão de projeção, número de cabeças de atenção e número de camadas Transformer.
- Explorar técnicas de regularização e aumento de dados, como *CutMix* e *Label Smoothing*, aplicadas sobre a configuração mais estável encontrada.
- Adotar um conjunto de métricas complementares: função de perda, acurácia, *Top-5 Accuracy* e F1-*macro*, de modo a oferecer uma avaliação mais completa, que abrange não apenas a taxa global de acertos, mas também o equilíbrio entre classes.
- Consolidar as análises em um protocolo de treinamento reprodutível, destacando escolhas que impactam diretamente tanto o custo computacional quanto a qualidade dos resultados alcançados, com ênfase em cenários de recursos limitados.

1.3 Justificativa

Do ponto de vista científico e prático, compreender como treinar o ViT do zero em um *dataset* desafiador e de baixa resolução, com recursos limitados, tem valor direto para pesquisadores e profissionais que não dispõem de infraestruturas de alto custo. A literatura frequentemente apresenta avanços baseados em grandes bases de dados e ambientes de alto desempenho, o que cria uma distância em relação às condições encontradas em muitos contextos acadêmicos e aplicados.

Nesse sentido, este trabalho justifica-se por investigar alternativas viáveis para cenários restritos, explorando como ajustes de hiperparâmetros, estratégias de aprendizado dinâmico e técnicas de regularização podem contribuir para maior estabilidade e capacidade de generalização do *Vision Transformer*. Além disso, a análise de diferentes configurações de hardware, incluindo CPUs e GPUs de entrada, reforça a relevância prática da pesquisa, pois permite compreender como tais limitações impactam o tempo de treinamento e a viabilidade de experimentos.

Dessa forma, este trabalho contribui ao oferecer uma referência acessível e reprodutível para o uso do ViT em cenários com restrições de recursos, aproximando a teoria das condições reais e fornecendo diretrizes práticas que auxiliam na tomada de decisão. Ao dar continuidade a um estudo já existente e propor melhorias, este trabalho busca manter o equilíbrio entre a seriedade da pesquisa, a utilidade prática e as condições reais de execução.

1.4 Estrutura do documento

O restante deste trabalho está organizado da seguinte forma: o **Capítulo 2** apresenta a base teórica necessária para a pesquisa, revisando conceitos de aprendizado profundo, a arquitetura Transformer, o *Vision Transformer*, o conjunto de dados *CIFAR-100*, os hiperparâmetros e técnicas de regularização. O **Capítulo 3** discute trabalhos relacionados que serviram de referência, com destaque para o estudo-base que orienta as melhorias propostas. Em seguida, o **Capítulo 4** descreve a metodologia experimental adotada, incluindo o ambiente computacional, as métricas de avaliação e o protocolo de análise dos hiperparâmetros e técnicas de otimização. O **Capítulo 5** apresenta os experimentos realizados e a análise crítica dos resultados, evidenciando os impactos das diferentes estratégias de treinamento sobre o desempenho do modelo. Por fim, o **Capítulo 6** reúne as conclusões do estudo e indica possibilidades para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos teóricos que sustentam este trabalho, com foco nas tecnologias e abordagens aplicadas à construção, treinamento e avaliação de modelos de classificação de imagens baseados em redes neurais profundas, em especial o *Vision Transformer* (ViT).

2.1 Aprendizado Profundo

O aprendizado profundo é uma vertente do aprendizado de máquina que utiliza redes neurais artificiais com múltiplas camadas. Segundo (SAP Portugal, 2025), a principal diferença para outros métodos está na profundidade das redes e na capacidade de descobrir automaticamente os dados mais relevantes para o aprendizado. Seu sucesso em tarefas complexas, sendo exemplo o reconhecimento de imagens, tradução automática e processamento de linguagem natural, deve-se à capacidade de aprender características relevantes de forma automática, sem depender de etapas manuais de análise prévia dos dados.

Essas redes são compostas por neurônios artificiais interconectados, onde cada camada aprende formas cada vez mais abstratas dos dados. Uma analogia seria quando reconhecemos o rosto de outra pessoa: primeiro vemos formas simples, depois partes do rosto (olhos, nariz) e, por fim, a pessoa completa. Esse processo em etapas reflete a inspiração das redes neurais no cérebro humano, onde os neurônios artificiais simulam, de forma simplificada, o funcionamento dos neurônios biológicos.

As redes neurais convolucionais (CNNs) dominaram a visão computacional por anos, extraindo padrões locais com grande eficiência por meio da aplicação de filtros que buscam na imagem formas como bordas, texturas e contornos. Mais recentemente, modelos baseados em atenção, como os Transformers, ganharam destaque por capturar relações globais entre os elementos da imagem. Enquanto a convolução foca em regiões específicas da imagem usando janelas deslizantes (filtros), a autoatenção permite que o modelo relacione todas as partes da imagem entre si, atribuindo pesos diferentes conforme a importância de cada região. Como descreve o GeeksforGeeks (GeeksforGeeks, 2025), essa abordagem trata a imagem como uma sequência de patches e aplica mecanismos de autoatenção para aprender as relações entre eles. Essa evolução permitiu o surgimento de arquiteturas como o *Vision Transformer* (ViT), que elimina convoluções em favor da autoatenção. A seguir, serão explorados em mais detalhes os fundamentos dos Transformers e sua aplicação no contexto da visão computacional.

2.2 Arquitetura Transformer

O Transformer é uma arquitetura originalmente proposta por (VASWANI et al., 2017) para tarefas de tradução automática no campo do processamento de linguagem natural. Ele utiliza unicamente mecanismos de atenção, dispensando o uso de convoluções ou recorrência para capturar dependências entre elementos de uma sequência.

Nesse contexto, seu componente central é o mecanismo de autoatenção (*self-attention*), que permite ao modelo avaliar ao mesmo tempo todos os elementos da entrada e atribuir pesos diferentes a cada um, de acordo com sua relevância.Essa operação é realizada por meio de projeções lineares, isto é, transformações matemáticas que mapeiam os vetores de entrada para um novo espaço de características, preservando relações lineares e permitindo gerar representações vetoriais enriquecidas e adaptativas.

A arquitetura é composta por blocos empilhados de codificadores (*encoders*), onde cada bloco contém uma camada de atenção com múltiplas cabeças (*multi-head attention*) seguida por uma rede *feedforward*, ou seja, uma rede onde a informação flui em uma única direção, da entrada para a saída. Esses componentes são acompanhados por conexões residuais e camadas de normalização que são utilizadas para estabilizar o treinamento e permitir arquiteturas mais profundas.

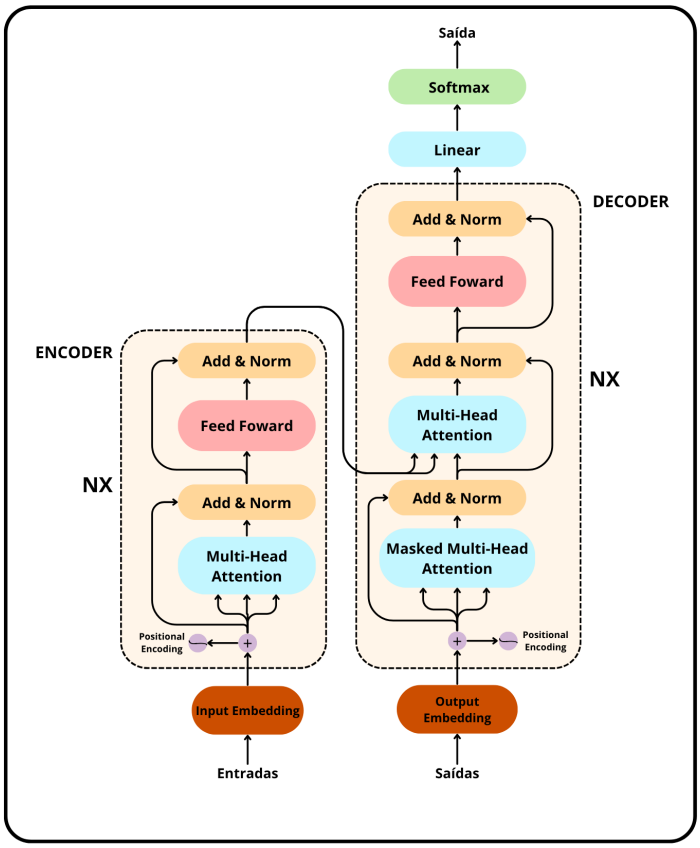


Figura 1 – Arquitetura do Transformer. Adaptada de (VASWANI et al., 2017).

A Figura 1 apresenta a arquitetura completa do Transformer, composta pelos módulos de codificação (Encoder) e decodificação (Decoder). O Encoder processa a entrada com atenção multi-cabeças, normalização e redes feedforward, gerando representações contextuais ricas. O Decoder, por sua vez, utiliza mecanismos de atenção mascarada e cruzada para gerar saídas passo a passo, o que é essencial em tarefas sequenciais, como quando traduzimos um texto. O presente trabalho envolve o treinamento de uma rede voltada ao processamento de imagens, dessa forma, apenas o módulo de Encoder é utilizado, tendo o objetivo da classificação da imagem como um todo, sem necessidade de geração sequencial de tokens.

Conforme ilustrado no lado esquerdo da Figura 1, o Encoder é composto por N blocos idênticos, cada um contendo um mecanismo de *multi-head self-attention* seguido de uma rede *feed forward*. Ambos os submódulos utilizam conexões residuais e normalização (*Add & Norm*), o que estabiliza o treinamento e preserva informações da entrada (VASWANI et al., 2017). Ao final, o Encoder gera uma representação com base em toda a entrada, utilizada pelo Decoder ou, como neste trabalho, para classificação de imagens.

Essa adaptação do Transformer à visão computacional, que divide a imagem em pequenos blocos chamados *patches*, substitui as operações convolucionais tradicionais e deu origem a modelos como o *Vision Transformer* (ViT), cuja estrutura será explorada na próxima seção.

2.3 Modelo Vision Transformer

O *ViT Vision Transformer* foi introduzido por (DOSOVITSKIY et al., 2020) como uma proposta inovadora para aplicar a arquitetura Transformer ao domínio da visão computacional. A principal ideia do modelo é tratar uma imagem como uma sequência de elementos, da mesma forma que é feito com palavras no contexto de linguagem natural, possibilitando assim o uso direto do mecanismo de atenção.

Para alcançar isso, cada imagem é segmentada em pequenos blocos fixos denominados patches, cujo tamanho é estipulado em pixels, variando conforme a resolução da imagem inicial. Cada um desses patches é comprimido e mapeado em um espaço vetorial de dimensão constante através de uma camada linear, produzindo uma série de vetores que irá representar a imagem de maneira integral. Esses vetores, em seguida, são enriquecidos com dados de posição por meio de *embeddings* dedicados (representações numéricas que mapeiam elementos para um espaço vetorial), possibilitando que o modelo mantenha a noção de sequência espacial entre os *patches*.

Assim, essa sequência vetorial é processada por várias camadas *encoders*, compostas por atenção multi-cabeça, redes *feedforward* e camadas de normalização. Ao final das camadas Transformer, o vetor correspondente ao token de classificação (*CLS — Classifi-*

cation Token, um token especial adicionado à sequência para agregar informações globais da entrada) é extraído e passado por uma *MLP* (*Multilayer Perceptron*, rede neural composta por múltiplas camadas totalmente conectadas), responsável por determinar a classe da imagem.

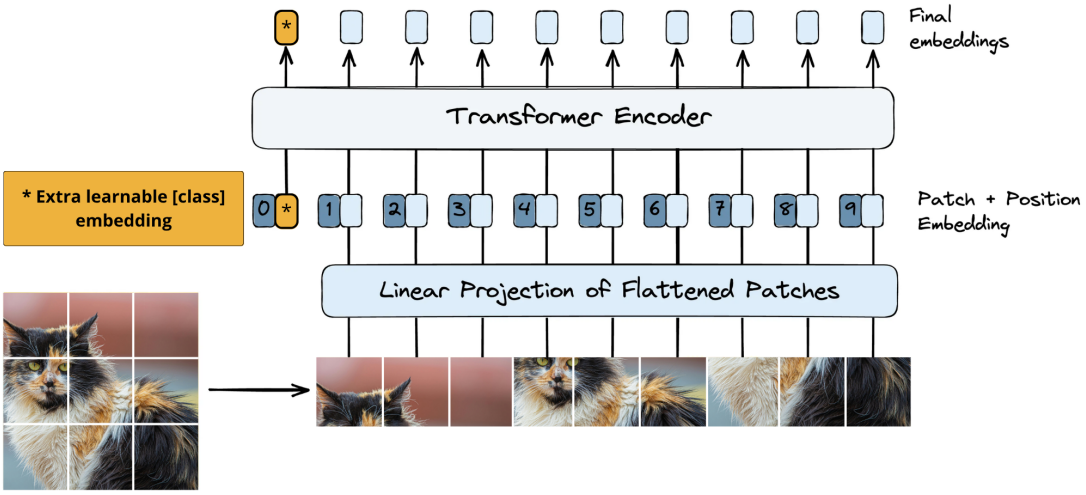


Figura 2 – Arquitetura do *Vision Transformer* (ViT). Adaptada de (DOSOVITSKIY et al., 2020; MORAIS, 2023).

A Figura 2 apresenta a arquitetura e o fluxo de processamento do *Vision Transformer*. Primeiramente, a imagem é dividida em pequenos blocos (*patches*), que são achatados e transformados em vetores por meio de uma projeção linear. Cada vetor recebe um *embedding* posicional, preservando a informação sobre sua localização original na imagem. Além disso, um token especial de classificação (*CLS*) é inserido no início da sequência. A sequência resultante é então passada por múltiplas camadas do *Transformer Encoder*, que aplicam mecanismos de autoatenção e redes *feedforward* para gerar representações refinadas. Ao final, apenas o vetor associado ao token *CLS* é encaminhado a uma *MLP*, cuja saída indica a classe atribuída à imagem.

No presente trabalho, essa arquitetura é empregada como base para os experimentos de classificação de imagens do conjunto CIFAR-100. A próxima seção descreve as características desse conjunto de dados, que impõem desafios específicos à tarefa de classificação devido ao seu número elevado de classes e à baixa resolução das imagens.

2.4 Base de Dados: CIFAR-100

O CIFAR-100 é um conjunto de dados amplamente utilizado para tarefas de classificação de imagens em visão computacional, proposto por (KRIZHEVSKY; HINTON, 2009) como benchmark para avaliação de modelos de aprendizado profundo.

Ele é composto por 60.000 imagens coloridas de baixa resolução (32×32 pixels), distribuídas em 100 classes distintas, com 600 imagens por classe. Os dados estão divididos em 50.000 amostras para treinamento e 10.000 para teste, com distribuição balanceada. Cada imagem pertence a uma única classe, sem sobreposição entre os conjuntos de treino e teste.

Um dos desafios do CIFAR-100 está na alta variabilidade intra-classe. Por exemplo, a classe "cachorro"inclui diversas raças, poses e contextos, exigindo que o modelo generalize bem mesmo diante de representações visuais muito distintas de um mesmo conceito.

Essas características tornam o CIFAR-100 um cenário ideal para avaliar a capacidade de generalização do *Vision Transformer* (ViT), especialmente em situações com resolução limitada e alto número de categorias.

2.5 Hiperparâmetros do Vision Transformer

O desempenho do modelo *Vision Transformer* depende fortemente da escolha adequada de seus hiperparâmetros. Esses parâmetros controlam aspectos fundamentais do treinamento, como a taxa de atualização dos pesos, a regularização, o tamanho do lote de dados e a estrutura interna da arquitetura. A seguir, são descritos os principais hiperparâmetros adotados no modelo *Vision Transformer* do presente trabalho.

❑ Taxa de aprendizado (*learning rate*)

A taxa de aprendizado é um dos hiperparâmetros mais relevantes no treinamento de redes neurais profundas, pois define o quanto os pesos do modelo são atualizados por iteração. Em outras palavras, essa taxa vai controlar o tamanho dos passos dados na direção do gradiente durante a otimização, influenciando na velocidade de convergência e na estabilidade do treinamento.

Um valor inadequado para esse parâmetro pode comprometer significativamente o desempenho do modelo:

- **Taxas muito baixas** tornam o treinamento lento e podem levar à estagnação em mínimos locais rasos;
- **Taxas muito altas** causam oscilações excessivas ou impedem a convergência, levando à divergência do modelo.

Conforme destacado por (GOODFELLOW; BENGIO; COURVILLE, 2016), a taxa de aprendizado exerce influência direta sobre a dinâmica da função de custo e é crítica para alcançar uma solução estável e eficiente. Essa sensibilidade torna sua definição ainda

mais desafiadora em tarefas complexas, como a classificação multiclasse no CIFAR-100, que envolve alta dimensionalidade e 100 categorias diferentes.

Estratégias de Ajuste da Taxa de Aprendizado

A taxa de aprendizado pode ser mantida constante durante o treinamento ou ajustada dinamicamente conforme a evolução das épocas. Estratégias de ajuste dinâmico são amplamente utilizadas para acelerar a convergência e evitar que o modelo fique preso em mínimos locais. Entre as abordagens mais eficazes para modelos baseados em Transformer, destacam-se o *Cosine Annealing* e o *Warmup*.

Cosine Annealing

O *Cosine Annealing*, proposto por (LOSHCHILOV; HUTTER, 2016), é uma estratégia de ajuste dinâmico que reduz a taxa de aprendizado ao longo do tempo seguindo uma curva cosseno. Com isso, o modelo realiza atualizações maiores nas etapas iniciais e vai diminuindo gradualmente nas etapas finais, permitindo ajustes mais refinados nos pesos e favorecendo uma convergência mais estável e eficiente.

A taxa de aprendizado em cada época t é definida pela seguinte equação:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}}\pi\right)\right) \tag{2.1}$$

Onde:

- η_t: taxa de aprendizado no passo t;
- $\eta_{\max}$: taxa inicial;
- $\eta_{\min}$: taxa final;
- T_{cur}: época atual dentro do ciclo;
- $T_{\max}$: duração total do ciclo.

A aplicação do *Cosine Annealing* tem-se mostrado vantajosa em arquiteturas como o *Vision Transformer*, que demandam maior cuidado na fase de ajuste fino dos pesos. Seus impactos práticos sobre a estabilidade e o desempenho do treinamento serão analisados em detalhes no Capítulo 4.

Warmup

A técnica de *warmup* consiste em introduzir uma fase inicial de aquecimento da taxa de aprendizado, na qual o valor de η cresce gradualmente a partir de um valor inicial

pequeno (η_0) até alcançar $\eta_{\max}$, ponto em que um agendador principal, como o *Cosine Annealing*, pode assumir o controle para iniciar a fase de decaimento.

Esse procedimento torna menos agressiva a transição entre a inicialização aleatória dos pesos e o início efetivo do processo de otimização, funcionando como uma "rampa de aceleração" para o aprendizado. Assim, temos uma redução de instabilidades nas primeiras iterações, quando os pesos ainda não foram suficientemente ajustados.

A eficácia do *warmup* foi amplamente demonstrada na literatura recente:

- (DOSOVITSKIY et al., 2020): afirmam que o uso conjunto de *warmup* e decaimento cosseno foi adotado em todos os experimentos com o *Vision Transformer*, como estratégia padrão de ajuste da taxa de aprendizado.
- (GOYAL et al., 2017): mostraram que o *warmup* acelera a convergência e evita instabilidades em treinamentos com redes ResNet no ImageNet, especialmente ao utilizar *batch sizes* elevados.

No contexto de modelos *Vision Transformer*, o *warmup* é especialmente importante, pois não possuem indutividades espaciais incorporadas (isto é, estruturas como convoluções que exploram relações locais entre pixels), tornando as primeiras atualizações ainda mais sensíveis. A adoção do *warmup*, portanto, ajuda a alcançar uma melhor estabilidade e eficácia do treinamento, como será discutido no Capítulo 4.

❑ Penalização por Peso (*weight decay*)

O *weight decay* é uma técnica de regularização que atua diretamente nos pesos do modelo, inserindo um termo de penalização à função de perda. Esse termo faz com que os valores dos pesos permaneçam pequenos durante o treinamento, ajudando a reduzir o risco de *overfitting* (quando o modelo se ajusta excessivamente aos dados de treino e perde capacidade de generalizar para novos dados) e a melhorar a generalização do modelo.

Matematicamente, a penalização por peso é incorporada como uma adição proporcional à norma dos pesos:

$$L_{\text{total}} = L_{\text{original}} + \lambda \|w\|_2^2 \tag{2.2}$$

Onde:

- L_{total}: função de perda total regularizada;
- L_{original}: função de perda original (sem regularização);

- λ: coeficiente de regularização (*weight decay*);
- w: vetor de pesos do modelo.

Em arquiteturas como o *Vision Transformer*, o uso de *weight decay* é fundamental para evitar que os embeddings ou projeções lineares cresçam fora de controle, o que pode prejudicar a estabilidade do treinamento. Conforme apontado por (LOSHCHILOV; HUTTER, 2018), essa técnica mostra-se especialmente eficaz quando combinada com otimizadores como o AdamW, que desacoplam a penalização da função de perda e a aplicam diretamente na atualização dos pesos. Essa é justamente a abordagem adotada no presente trabalho, a qual será abordada no capítulo 4.

❑ Tamanho do Lote (*batch size*)

O *batch size* define quantas amostras serão processadas antes que os parâmetros do modelo sejam atualizados. De acordo com estudos como os de (GOODFELLOW; BENGIO; COURVILLE, 2016), o tamanho do lote influencia diretamente a variância do gradiente e o tipo de mínimo encontrado durante o treinamento, afetando a estabilidade da otimização e a capacidade de generalização do modelo.

- Tamanhos de lote menores geralmente promovem uma maior variabilidade no gradiente, o que pode funcionar como uma forma de regularização implícita e ajudar a evitar mínimos locais rasos (regiões da função de perda onde o erro é baixo, mas a solução não é ideal). No entanto, isso também pode tornar o processo mais ruidoso e instável.
- Lotes maiores tendem a gerar estimativas de gradiente mais estáveis, mas exigem mais memória e podem levar a convergência para soluções sub ótimas, além de reduzir a capacidade de generalização do modelo.

Sendo assim, em estruturas como o *Vision Transformer*, o *batch size* afeta diretamente a utilização da memória e a eficácia do processamento, especialmente em situações sem uma GPU dedicada. Assim, a seleção desse parâmetro deve levar em conta o equilíbrio entre a estabilidade, a habilidade de generalização e os recursos de computação disponíveis.

❑ Número de Épocas (*num epochs*)

O número de épocas define quantas vezes o modelo percorrerá todo o conjunto de dados de treinamento. Em concordância com os dados do estudo em (GOODFELLOW; BENGIO; COURVILLE, 2016), esse hiperparâmetro está diretamente ligado ao

processo de aprendizado, influenciando a capacidade do modelo de ajustar seus pesos às características dos dados, onde:

- Poucas épocas podem resultar em sub ajuste (*underfitting*), pois o modelo não tem tempo suficiente para aprender os padrões presentes no conjunto.
- Por outro lado, um número excessivo de épocas pode levar ao sobre ajuste (*overfitting*), especialmente quando o modelo memoriza os dados de treino sem conseguir generalizar para novos exemplos.

❑ Tamanho da Imagem (*image size*)

Esse hiperparâmetro define a resolução das imagens de entrada após o redimensionamento. No *ViT*, imagens maiores aumentam a quantidade de informação visual disponível para cada patch, permitindo representações mais detalhadas. Como consequência, o número de patches (ou tokens) também cresce, o que eleva a complexidade do modelo e o custo computacional. Estudos como o de (KOLESNIKOV et al., 2021) indicam que resoluções mais altas tendem a melhorar a acurácia, mas com ganhos marginais e aumento significativo no uso de memória e tempo de treinamento.

❑ Tamanho do Patch (*patch size*)

O tamanho do patch define em quantas partes a imagem será dividida antes de ser processada pelo *Vision Transformer*. Conforme discutido por (DOSOVITSKIY et al., 2020), patches menores capturam mais detalhes locais, aumentando a capacidade do modelo de aprender padrões finos, mas também elevam o número de tokens que aumenta o custo computacional. Por outro lado, (TOUVRON et al., 2021) destacam que patches maiores reduzem significativamente o custo de processamento ao gerar menos tokens, embora possam resultar na perda de informações de pequenas estruturas.

(a) Imagem original

(b) Imagem dividida em patches

Figura 3 – Exemplo visual do efeito do `patch size` na divisão de uma imagem. Adaptado de (SINGH, 2020).

A Figura 3 apresenta uma comparação entre uma imagem original e sua versão dividida em patches. No exemplo, a imagem foi segmentada em 64 regiões quadradas de tamanho igual, correspondendo a um arranjo de 8×8 patches. Cada patch é tratado como um token independente pelo *Vision Transformer*, sendo posteriormente projetado em um espaço vetorial para captura de características. Esse processo permite que o modelo aprenda representações a partir de diferentes regiões da imagem, explorando tanto padrões locais quanto suas relações globais.

❑ Número de Patches (`num patches`)

O parâmetro `num patches` é definido a partir da combinação entre o `image size` e o `patch size`, determinando quantos vetores (tokens) serão gerados e enviados ao encoder. Conforme discutido por (DOSOVITSKIY et al., 2020; HAN et al., 2021), um maior número de patches aumenta a granularidade da representação, permitindo que o modelo capture relações mais detalhadas entre diferentes regiões da imagem. No entanto, esse ganho vem acompanhado de um custo computacional e consumo de memória proporcionalmente maiores.

Image size	**Patch size**	**Num patches**
32×32	8	16
64×64	8	64
128×128	16	64
224×224	16	196

Tabela 1 – Exemplos de combinações de `image size` e `patch size` e o número de patches resultante.

A Tabela 1 ilustra como diferentes combinações de `image size` e `patch size` afetam o número total de patches (`num patches`) gerados. Observa-se que, ao reduzir o tamanho do patch ou aumentar a resolução da imagem, o número de patches cresce rapidamente. Esse aumento significa que o modelo processará mais tokens, capturando detalhes mais finos, mas com maior custo computacional e maior custo de memória.

❑ Dimensão de Projeção (`projection dim`)

O `projection dim` define o tamanho do vetor que representa cada patch após a projeção linear. Conforme discutido por (TOUVRON et al., 2021), dimensões maiores oferecem maior capacidade de codificação e expressividade, permitindo que o modelo capture relações mais complexas entre os patches. Todavia, esse ganho vem acompanhado de um aumento no custo de armazenamento e processamento, impactando diretamente a velocidade de treinamento.

❑ Número de Cabeças de Atenção (`num heads`)

O parâmetro `num heads` define quantas "cabeças" paralelas compõem o mecanismo de *Multi-Head Attention*. Cada uma dessas cabeças atua como um ponto de observação independente, capaz de focar em diferentes regiões ou relações da imagem de entrada simultaneamente. Essa abordagem permite que o modelo capture padrões complementares em paralelo, enriquecendo a representação final, como descrito por (VASWANI et al., 2017) e adaptado ao contexto de visão computacional por (DOSOVITSKIY et al., 2020). Em contrapartida, aumentar `num heads` eleva o custo computacional (devido a mais projeções e operações de atenção) e pode exigir mais dados e regularização para evitar *overfitting*.

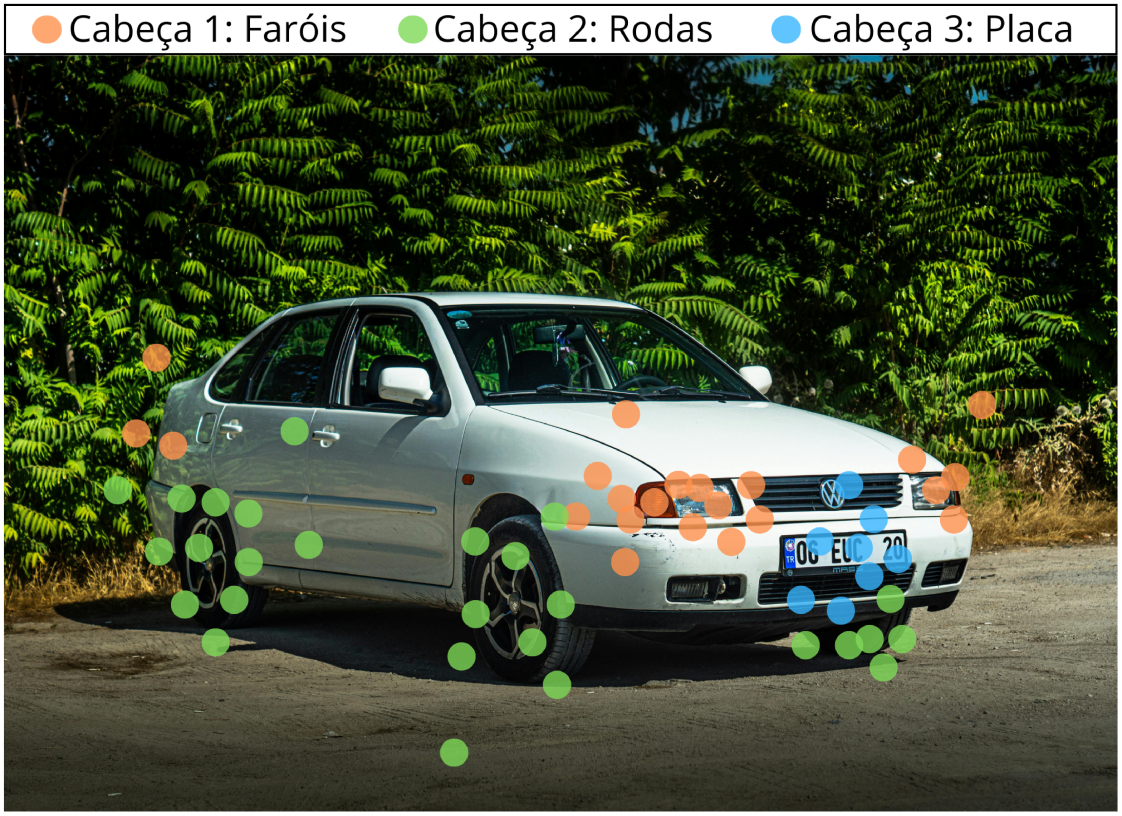


Figura 4 – Ilustração didática de *Multi-Head Attention* em uma imagem: **Cabeça 1** foca nos **faróis**, **Cabeça 2** foca nas **rodas** e **Cabeça 3** foca na **placa**. Adaptado de (SAGLAM, 2025).

Na Figura 4 cada cor representa uma cabeça distinta aprendendo um padrão próprio: faróis (laranja), rodas (verde) e placa (azul). Na prática, as cabeças não "recortam" pixels, mas atribuem pesos de atenção a pares de *tokens*/*patches*; por isso, aparecem alguns pontos fora do objeto principal esperado. Assim, aumentar o valor de `num heads` permite identificar uma variedade maior de padrões, mas também torna o treinamento e a inferência mais lentos e exigentes computacionalmente.

❑ Unidades Internas dos Blocos Transformer (`transformer units`)

O parâmetro `transformer units` define o tamanho das camadas internas da *Multi-Layer Perceptron* que estão em cada bloco Transformer. Essa *MLP* processa e transforma as representações extraídas pelo mecanismo de atenção, aplicando operações não lineares que aumentam a capacidade do modelo de capturar padrões complexos. Como observado por (VASWANI et al., 2017) e (DOSOVITSKIY et al., 2020), valores mais altos para esse parâmetro ampliam o poder de representação do modelo, mas também elevam o consumo de memória e o tempo de processamento, podendo exigir maior poder computacional para manter a eficiência do treinamento.

❑ **Número de Camadas Transformer (`transformer layers`)**

O parâmetro `transformer layers` indica a profundidade do encoder no *Vision Transformer*, ou seja, quantos blocos Transformer compõem a arquitetura. Cada bloco adicional permite que o modelo aprenda representações progressivamente mais complexas e de alto nível, melhorando sua capacidade de capturar padrões sutis nos dados. Porém, como observado por (TOUVRON et al., 2021; DOSOVITSKIY et al., 2020), arquiteturas mais complexas e profundas demandam maior tempo de treinamento e recursos computacionais, além de aumentarem o risco de *overfitting* quando não há dados suficientes.

❑ **Unidades da *MLP* Final (`mlp head units`)**

O parâmetro `mlp head units` especifica o número de unidades da *MLP* que transforma a representação final gerada pelo encoder em uma predição de classe, ou seja, atribui a cada amostra processada uma probabilidade de pertencer a cada categoria do problema. Como observado por (DOSOVITSKIY et al., 2020; TOUVRON et al., 2021), um número maior de unidades pode ampliar a capacidade de generalização do modelo, permitindo que ele capture relações mais complexas entre as características extraídas. No entanto, configurações mais amplas também elevam a complexidade da rede, aumentando o custo computacional e o risco de *overfitting* quando o conjunto de dados não é suficientemente grande ou diverso.

2.6 Técnicas Avançadas para Otimização e Generalização

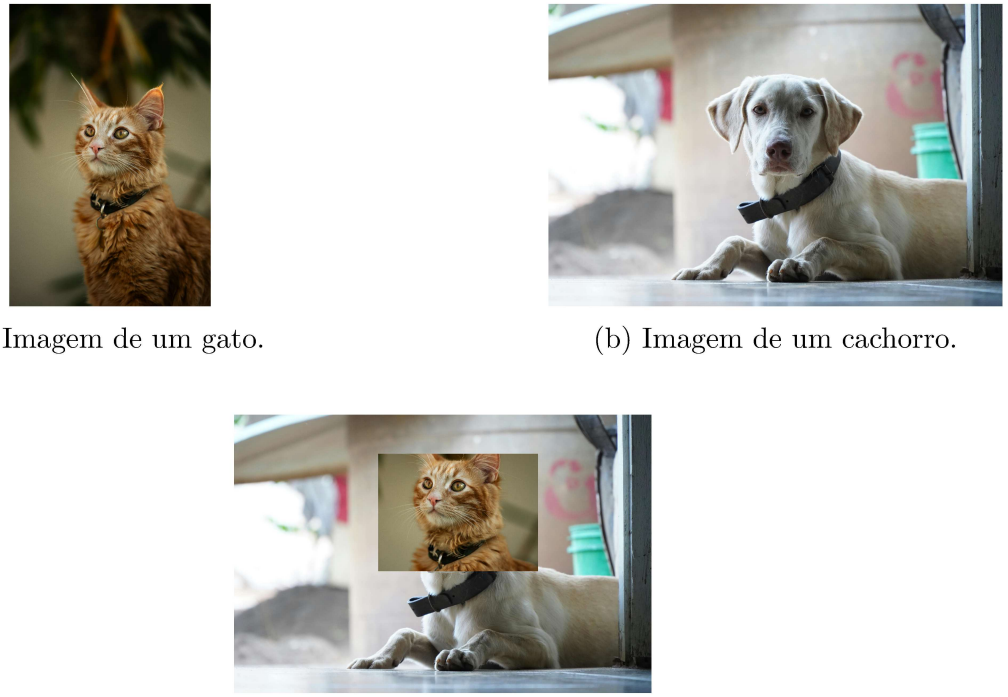
Além da escolha adequada de hiperparâmetros, o desempenho de um modelo de aprendizado de máquina pode ser aprimorado por meio de técnicas que atuam diretamente sobre os dados e o processo de treinamento. Neste trabalho, foram adotadas abordagens voltadas tanto para aumentar a capacidade de generalização quanto para reduzir o risco de *overfitting*. Entre elas, destacam-se o *CutMix*, que combina regiões de diferentes imagens durante o treinamento, e o *Label Smoothing*, que ajusta as distribuições de rótulos para evitar predições muito confiantes. A seguir, cada técnica será detalhada, juntamente com sua aplicação no contexto do *Vision Transformer*.

2.6.1 CutMix

O *CutMix* é uma técnica de aumento de dados (*data augmentation*) que combina duas imagens do conjunto de treino, substituindo uma região de uma pela mesma área da outra imagem. Além disso, os rótulos também são misturados de forma proporcional à área substituída, gerando *soft labels*, ou seja, rótulos com valores fracionários atribuídos

a cada classe, representando a proporção de pertencimento, em contraste com os rótulos tradicionais (*hard labels*) que utilizam apenas 0 ou 1.

Essa abordagem permite que o modelo aprenda a identificar padrões visuais mesmo quando apenas partes dos objetos estão presentes, além de incentivar a generalização. De acordo com (YUN et al., 2019), o *CutMix* apresenta vantagens em relação a métodos tradicionais como *Cutout* (remoção de uma região da imagem, substituindo-a por um bloco uniforme) ou *Mixup* (combinação linear de duas imagens e seus rótulos), pois preserva informações reais de textura e bordas na região substituída, contribuindo para uma melhor robustez contra o *overfitting*.



(a) Imagem de um gato.

(b) Imagem de um cachorro.

(c) Imagem resultante do *CutMix*.

Figura 5 – Exemplo visual do funcionamento do *CutMix*. Adaptado de (Pexels, 2024; Pexels, 2025).

O Figura 5 apresenta um exemplo visual do funcionamento do *CutMix*. As duas primeiras imagens são originais com um gato (a) e um cachorro (b). A imagem (c) representa o resultado da técnica, em que uma região da imagem do cachorro foi substituída por um recorte da imagem do gato. O rótulo associado à nova imagem é ajustado de forma proporcional à área ocupada por cada uma das imagens originais, gerando *soft labels*, que indicam a proporção de pertencimento a cada classe.

2.6.2 Label Smoothing

O *Label Smoothing* é uma técnica de regularização que, em vez de utilizar rótulos binários rígidos (1 para a classe correta e 0 para as demais), distribui uma pequena fração ε de probabilidade entre todas as classes. Com isso, a classe correta recebe um valor ligeiramente menor que 1, enquanto as demais recebem valores fracionários diferentes de zero. Essa suavização reduz a confiança extrema do modelo e contribui para melhorar a calibração das previsões, como proposto por (SZEGEDY et al., 2016) e posteriormente incorporado em arquiteturas baseadas em atenção, como o Transformer (VASWANI et al., 2017).

Dessa forma, o objetivo dessa técnica é evitar que o modelo atribua probabilidade máxima (100%) apenas à classe verdadeira, incentivando-o a manter margens de decisão mais robustas e a não depender excessivamente de padrões específicos de treino. Isso ajuda a reduzir *overfitting*, melhora a generalização e torna as predições mais calibradas.

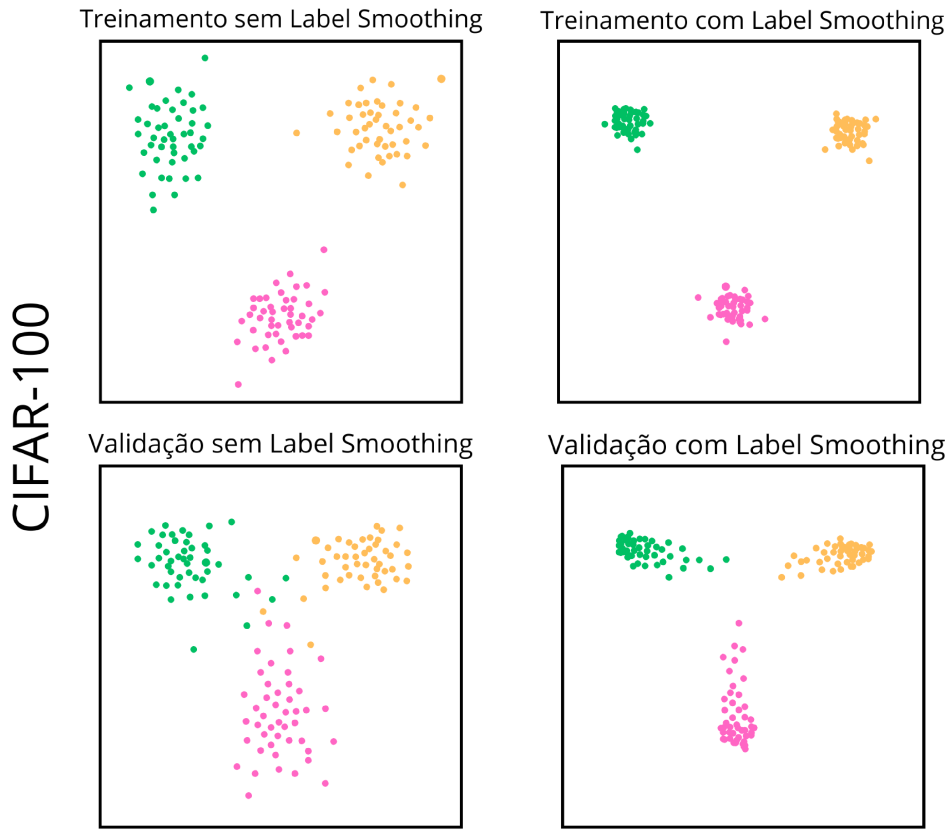


Figura 6 – Visualização de representações internas de três classes do CIFAR-100 sem e com *label smoothing*. Adaptado de (MüLLER; KORNBLITH; HINTON, 2019).

O Figura 6 ilustra o impacto do *Label Smoothing* na separação entre classes no

espaço de representações internas do modelo, considerando o conjunto CIFAR-100. Cada ponto representa uma amostra projetada para duas dimensões, agrupada conforme a classe prevista pelo modelo. Nos gráficos à esquerda, sem a técnica, observa-se que as amostras de cada classe estão mais dispersas e com margens menos definidas, especialmente na validação, o que indica que o modelo tende a memorizar excessivamente o conjunto de treino e a atribuir probabilidades muito altas para a classe prevista, mesmo em exemplos pouco representativos. Já nos gráficos à direita, com *Label Smoothing*, os grupos apresentam maior coesão e distribuição mais uniforme, tanto no treino quanto na validação, sugerindo uma organização interna mais estável, melhor capacidade de generalização e menor propensão a predições excessivamente confiantes.

3 Trabalhos Correlatos

Este capítulo apresenta trabalhos que servem de referência e fundamentação para o desenvolvimento desta pesquisa. Aqui serão descritas abordagens, técnicas e resultados relevantes, com destaque para estudos que utilizam o *Vision Transformer* (ViT) na classificação de imagens, bem como métodos de otimização aplicados a esse modelo. Entre eles, inclui-se o trabalho-base adotado como ponto de partida para as melhorias propostas neste TCC.

O trabalho de (VASWANI et al., 2017) foi responsável por introduzir a arquitetura Transformer, inicialmente voltada para tradução automática e outras tarefas de Processamento de Linguagem Natural, substituindo totalmente redes recorrentes e convolucionais. A proposta dele gira em torno do mecanismo de *self-attention*, capaz de processar todos os elementos de uma sequência em paralelo e capturar dependências de longo alcance com eficiência. A arquitetura é composta por camadas empilhadas de *multi-head attention*, redes totalmente conectadas e *positional encoding*, organizadas em blocos de *encoder* e *decoder*. Nos experimentos originais, o Transformer superou modelos RNN e CNN tanto em desempenho quanto em custo computacional. Este trabalho serviu de base conceitual para arquiteturas posteriores, como BERT, GPT e o próprio ViT, cuja adaptação para visão computacional mantém a essência do *self-attention* aplicado a dados não sequenciais, sendo exemplo imagens.

Já na pesquisa de (DOSOVITSKIY et al., 2020) foi apresentado o *Vision Transformer* (ViT), adaptando o Transformer para tarefas de classificação de imagens. Nesse modelo, a imagem é dividida em blocos fixos de 16×16 pixels, projetados de forma linear em vetores aos quais se somam codificações posicionais. Um token especial (`[CLS]`) representa a saída de classificação, que é processada por múltiplas camadas de *multi-head self-attention* e redes totalmente conectadas, com conexões residuais e normalização de camadas. Estratégias como *data augmentation*, *Dropout* e *stochastic depth* foram aplicadas para melhorar a generalização. Nos experimentos, o ViT obteve resultados similares e, em alguns casos, superiores a modelos convolucionais, sobretudo quando pré-treinado em grandes bases como JFT-300M e ajustado (*fine-tuning*) para conjuntos menores. Entretanto, os autores observaram que, em bases limitadas, o desempenho inicial pode ser inferior sem pré-treinamento.

O trabalho de (SILVA, 2024) teve como objetivo estudar e analisar, por meio de experimentos, o funcionamento do *Vision Transformer* (ViT) ao processar e classificar imagens, tendo como referência o conjunto de dados CIFAR-100. O autor apresentou uma revisão de arquiteturas de redes neurais amplamente utilizadas, incluindo Percep-

tron Multicamadas (PMC), Redes Convolucionais (CNN) e Redes Recorrentes (RNN), destacando a evolução que proporcionou o surgimento dos Transformers. O modelo foi implementado a partir da versão disponível no Keras, seguindo a proposta original de (DOSOVITSKIY et al., 2020), preservando etapas como a divisão da imagem em *patches*, projeção linear, adição de *positional encoding* e o uso de token `[CLS]` para classificação. Os experimentos foram conduzidos em um notebook com processador Intel Core i5 de 7ª geração, 8 GB de RAM e usando a CPU para a execução, o que resultou em cerca de 30 minutos por época. Foram mantidos hiperparâmetros fixos, variando apenas o número de épocas (20, 30, 50 e 100), com *learning rate* $= 10^{-4}$, *weight decay* $= 10^{-4}$, *image size* $= 72$, *batch size* $= 128$, *patch size* $= 6$, quatro cabeças de atenção e oito camadas Transformer. Os resultados mostraram um crescimento da acurácia (43,21% a 52,63%) e da *top-5 accuracy* (73,91% a 79,77%) conforme o aumento de épocas, mas também indicaram sinais de estagnação no ganho de desempenho, diretamente ligadas a limitações computacionais e à estabilidade da otimização.

O presente trabalho reúne contribuições dos estudos anteriores, aproveitando elementos centrais de cada um para construir sua base experimental. Nesse contexto, esta pesquisa parte diretamente do estudo-base de (SILVA, 2024), mantendo a mesma arquitetura ViT e o conjunto de dados CIFAR-100, mas buscando superar as limitações encontradas por meio de otimizações de hiperparâmetros, uso de agendadores dinâmicos de *learning rate* (como *cosine annealing* com *warmup*), inclusão de métricas como o F1-*macro* e aplicação de técnicas de regularização e aumento de dados, como CutMix e *label smoothing*. Além disso, apoia-se também nos princípios do Transformer de (VASWANI et al., 2017), que introduziram o *self-attention* como núcleo da arquitetura, e na adaptação feita por (DOSOVITSKIY et al., 2020) para o domínio da visão computacional com o *Vision Transformer*. Com essa integração, o objetivo é não apenas aumentar a acurácia, mas também ampliar a robustez e a capacidade de generalização do modelo, evitando a estagnação de desempenho observada no estudo-base.

4 Desenvolvimento

Este capítulo apresenta o desenvolvimento de coparações e melhorias baseado no *Vision Transformer* (ViT) para o conjunto de dados *CIFAR-100*. São descritos o ambiente computacional utilizado, o conjunto de dados, as métricas de avaliação, os principais hiperparâmetros investigados e as técnicas de otimização aplicadas. Também é apresentada a forma como os experimentos foram organizados, servindo de base para a análise dos resultados no Capítulo 5.

4.1 Arquitetura Utilizada

Os experimentos realizados neste trabalho foram conduzidos a partir da implementação base apresentada em (SILVA, 2024), discutida no Capítulo de Trabalhos Correlatos. Naquele estudo, o autor implementou o *Vision Transformer* (ViT) para o conjunto de dados *CIFAR-100*, seguindo a proposta original de (DOSOVITSKIY et al., 2020), e executou os testes em um ambiente restrito a CPU, com processador Intel Core i5 de 7ª geração e 8 GB de RAM. Esse cenário resultou em aproximadamente 30 minutos por época de treinamento, evidenciando limitações computacionais e impacto direto no desempenho obtido.

Com base nessa referência, o presente trabalho expandiu a análise, utilizando um computador com processador Intel Core i7 de 7ª geração e 32 GB de RAM. Além disso, buscou-se empregar uma GPU NVIDIA GTX 1050 Ti, com o objetivo de acelerar o treinamento, reduzindo o tempo por época e viabilizando experimentos mais complexos, inviáveis apenas em CPU. Essa escolha permitiu comparar o impacto do hardware no tempo de execução e nos resultados do modelo.

Além disso, o ambiente de desenvolvimento foi configurado em *Python 3*, utilizando como principal biblioteca o *TensorFlow/Keras* para construção, treinamento e avaliação do modelo. Foram empregadas também bibliotecas auxiliares como *NumPy* para manipulação numérica, *Matplotlib* para geração de gráficos e *scikit-learn* para cálculo da métrica *F1-score*. O código foi modularizado, com parâmetros definidos em arquivos externos para facilitar a reprodutibilidade dos experimentos.

Por fim, considerando as restrições computacionais, a análise dos resultados não se limitou apenas ao número de épocas, mas também levou em conta o tempo total de treinamento, permitindo uma comparação mais justa entre diferentes configurações de hardware.

4.2 Conjunto de Dados

Neste capítulo, detalha-se como o conjunto de dados *CIFAR-100*, já apresentado no Capítulo 2 quanto às suas características gerais, foi manipulado e adaptado para o treinamento do *Vision Transformer* (ViT).

O *dataset* foi carregado a partir da biblioteca *Keras*, preservando a divisão padrão em 50 mil imagens para treino e 10 mil para teste. Cada imagem possui dimensão original de 32×32 pixels e três canais de cor (RGB). As classes foram convertidas para o formato *one-hot encoding*, no qual cada rótulo é representado por um vetor em que apenas a posição correspondente à classe correta assume valor 1, sendo as demais iguais a 0. Esse formato é adequado à função de perda empregada.

Para aumentar a robustez do modelo e reduzir o sobreajuste, aplicou-se uma sequência de *data augmentation*, implementada como um bloco de camadas do *Keras*, composta por:

- **Normalização**: padronização dos valores de pixel em torno de média zero e variância unitária;
- **Redimensionamento**: ampliação das imagens para 72×72 pixels, de forma a possibilitar a extração de patches de tamanho definido pelo parâmetro `patch size`;
- **Flip horizontal aleatório**: inversão das imagens no eixo horizontal com probabilidade 0,5;
- **Rotação aleatória**: pequenas rotações aleatórias em até 2% da escala;
- **Zoom aleatório**: variação da escala de até 20% em altura e largura.

Esse pipeline de pré-processamento foi aplicado tanto ao conjunto de treino quanto de validação, sendo que a normalização foi ajustada com base nos dados de treino, garantindo consistência na transformação das imagens. Após o *data augmentation*, as imagens foram divididas em *batches* de tamanho definido pelo hiperparâmetro `batch size`, facilitando a eficiência no processamento.

Com essas transformações, o conjunto de dados foi preparado para entrada no modelo, onde as imagens redimensionadas foram segmentadas em *patches* e posteriormente codificadas para uso nas camadas do *Vision Transformer*.

4.3 Métricas Estatísticas de Avaliação

Para analisar o desempenho do modelo, foi necessário definir previamente um conjunto de métricas que orientassem tanto o processo de treinamento quanto a comparação

final entre diferentes configurações. Nesse sentido, este trabalho adotou quatro métricas principais:

- **Função de perda (loss)**: *Categorical Cross-Entropy*, que mede a divergência entre as predições do modelo e os rótulos reais. É utilizada durante o treinamento para orientar a atualização dos pesos da rede.
- **Acurácia**: proporção de previsões corretas em relação ao total de amostras. É a métrica mais intuitiva e fornece uma visão geral do desempenho do modelo.
- **Top-5 Accuracy**: considera a predição correta quando a classe real está entre as cinco mais prováveis apontadas pelo modelo. Essa métrica é especialmente relevante em problemas com muitas classes, como o *CIFAR-100*, pois tolera pequenas incertezas do modelo.
- **F1-score (macro)**: avalia o quanto o modelo acerta levando em conta duas coisas ao mesmo tempo: se ele consegue encontrar as respostas corretas (revocação) e se, quando responde, responde da forma certa (precisão). No cálculo *macro*, cada classe tem o mesmo peso na média final, evitando que o modelo pareça bom apenas por acertar mais nas classes com muitas imagens.

As três primeiras métricas, função de perda, acurácia e *top-5 accuracy*, já haviam sido utilizadas no trabalho base de (SILVA, 2024). Neste estudo foi incorporado o *F1-score (macro)* como diferencial, com o objetivo de oferecer uma avaliação mais completa. Essa métrica equilibra precisão e revocação em todas as classes, atribuindo a cada uma o mesmo peso. Dessa forma, o modelo não é avaliado apenas pela proporção global de acertos, mas também pela sua capacidade de manter um bom desempenho mesmo em classes menos representadas.

4.4 Otimização de Hiperparâmetros

Esta seção apresenta como foram definidas as configurações dos principais hiperparâmetros do ViT neste trabalho, com foco no que foi feito e por que foi adotado segundo a literatura. As comparações numéricas correspondentes estão no Capítulo 5.

Buscou-se seguiu-se duas etapas: (i) **teste isolado**, alterando apenas o próprio parâmetro e o restante do pipeline constante; e (ii) **testes combinados**, alteração de parâmetros em conjunto. Ambas etapas com intenção de comparação com os resultados baseados nos parâmetros base.

❑ Taxa de aprendizado (*learning rate*)

A título de ampliação em relação ao estudo de (SILVA, 2024), que adotava exclusivamente um valor fixo para a taxa de aprendizado, investigaram-se também estratégias de agendamento dinâmico. Inicialmente foi considerada a técnica de *Cosine Annealing*, e posteriormente avaliou-se a sua combinação com o uso de *warmup*. Nos testes em conjunto, essa política foi analisada em paralelo com os ajustes de capacidade do modelo e do tamanho do lote, uma vez que a literatura aponta que o agendamento da taxa de aprendizado interage tanto com a escala do modelo quanto com o ruído e a estabilidade induzidos pelo *batch*, recomendando a calibração conjunta desses fatores (DOSOVITSKIY et al., 2020; TOUVRON et al., 2021; GOYAL et al., 2017).

❑ Parâmetro de regularização (*weight decay*)

O *weight decay* foi explorado a partir do valor base de **0.0001**, variando entre limites mais baixos e mais altos dentro de intervalos (**1e-5** e **1e-3**) que foram adotados em outras pesquisas e considerados seguros pela literatura para mapear a sensibilidade do treinamento. Com o limite menor buscou-se verificar se uma penalização mais branda permitiria ao modelo memorizar melhor os padrões do conjunto de treino, possivelmente elevando a acurácia, ainda que com maior risco de sobreajuste. Já o aumento para valores mais altos foi testado para avaliar se uma regularização mais agressiva poderia favorecer a generalização, prevenindo sobreajuste e, assim, melhorar a performance em dados não vistos. Nos testes em conjunto, esse parâmetro foi ajustado em coordenação com a capacidade do modelo e outros controles (como profundidade do Transformer e tamanho das MLPs), pois o nível adequado de penalização depende do porte/profundidade da rede e do regime de treino, por isso, a calibração foi feita de forma conjunta (DOSOVITSKIY et al., 2020; TOUVRON et al., 2021).

❑ Tamanho do lote (*batch size*)

O *batch size* partiu do valor base de **128**, variando entre intervalos de **64** e **256** que são comumente utilizados em cenários de resolução menor como o caso do *CIFAR-100*, para mapear a sensibilidade do treinamento e o impacto no custo computacional. Nos testes em conjunto, o *batch size* foi ajustado em coordenação com o agendamento de *learning rate* e a regularização, pois estudos indicam que o tamanho do lote afeta o ruído/estabilidade do gradiente e o passo efetivo (tamanho do ajuste feito nos pesos a cada atualização), devendo ser ajustado junto a esses elementos (GOYAL et al., 2017; TOUVRON et al., 2021).

❑ Dimensão de projeção (*projection dim*)

O *projection dim* foi aplicado a partir do valor base de **64**, com aumentos moderados para **96** e **128** dentro da faixa usual para ViTs compactos. Esses incrementos são

recorrentes por elevar a dimensão por cabeça para faixas usuais (24 e 32 com 4 cabeças) e mantêm o *embedding* em múltiplos de 32 o que facilita a divisão entre cabeças e a eficiência prática, ampliando a capacidade com custo ainda administrável para o CIFAR-100. Valores abaixo dos apresentados não foram priorizados, pois referências e implementações tratam 64 como limite mínimo prático para preservar a expressividade dos *patches* em tarefas com muitas classes (VASWANI et al., 2017; DOSOVITSKIY et al., 2020; TOUVRON et al., 2021; STEINER et al., 2021). Este parâmetro não integrou os testes em conjunto.

❑ Número de cabeças de atenção (*num heads*)

O *num heads* foi explorado a partir do valor base de **4**, avaliando um aumento para **6** com *projection dim* mantido em **64**, para mapear a sensibilidade do modelo ao particionamento da projeção entre cabeças. Essa escolha segue o entendimento de que ajustes nesse parâmetro devem considerar a dimensão disponível por cabeça, por isso, valores maiores só são recomendados em conjunto com aumento proporcional de *projection dim* em cenários específicos (VASWANI et al., 2017; DOSOVITSKIY et al., 2020). Este parâmetro não integrou os testes em conjunto.

❑ Número de camadas Transformer (*transformer layers*)

O *transformer layers* partiu do valor base de **8**, com incremento seguro para **10** dentro da faixa usual para modelos ViTs. Esse aumento moderado tende a ser benéfico por acrescentar passos adicionais de atenção e feed-forward (MLP interna que amplia a capacidade de representação), o que melhora as representações, aumenta a interação entre *patches* e permite capturar padrões um pouco mais complexos sem inflar o custo. Nos testes em conjunto, ele foi ajustado em coordenação com *batch size*, *mlp head units* e o agendamento de *learning rate*, pois a literatura indica que profundidade interage com largura/capacidade e com o regime de otimização, devendo ser calibrada de forma conjunta quando se busca ampliar a expressividade do modelo (DOSOVITSKIY et al., 2020; TOUVRON et al., 2021; STEINER et al., 2021).

❑ Unidades da MLP final (*mlp head units*)

Partiu-se da configuração base **[2048, 1024]**, também comum em ViTs compactos. Não foram feitos testes isolados desse parâmetro, pois sua variação sozinha tende a ter efeito limitado sem ajustes estruturais. Nos testes em conjunto, a largura da MLP de saída foi ajustada em coordenação com *batch size*, *transformer layers* e o agendamento do *learning rate*, em busca de ampliar a capacidade do classificador de forma equilibrada. Assim, a maior profundidade/largura fornecem representações mais ricas, o *learning rate* dinâmico estabiliza a otimização e a regularização ajuda a conter sobreajuste, tornando

a ampliação da MLP efetiva quando tem-se muitas classes (TOUVRON et al., 2021; DOSOVITSKIY et al., 2020)

❑ **Unidades internas dos blocos (*transformer units*)**

Adotou-se a proporção base **[128, 64]** (isto é, [2× *projection dim*, 1× *projection dim*]), prática padrão em Transformers por equilibrar expressividade e estabilidade. Não foram realizados testes isolados nem testes em conjunto para esse parâmetro. Ele foi mantido fixo ao longo do desenvolvimento por já atender às recomendações usuais sem demandar ajustes adicionais neste cenário (VASWANI et al., 2017; DOSOVITSKIY et al., 2020).

❑ **Tamanho do patch (*patch size*)**

Manteve-se **patch size = 6** sobre imagens **72×72**, resultando em cerca de **144** patches por amostra, alinhado à faixa de tokens que costuma ser adotada em ViTs compactos. Aqui também não foram conduzidos testes isolados nem testes em conjunto, pois reduzir o *patch size* aumentaria excessivamente o comprimento da sequência e o custo computacional e amplia-lo reduziria a granularidade das informações. Assim, o *patch size* foi mantido fixo, e sua reavaliação fica reservada a cenários de mudanças arquiteturais mais amplas (DOSOVITSKIY et al., 2020; TOUVRON et al., 2021).

4.5 Técnicas de Otimização e Regularização

Esta seção descreve como técnicas adicionais foram incorporadas ao pipeline do ViT para aumentar robustez e controle de sobreajuste. As avaliações partiram da configuração consolidada de hiperparâmetros obtida nos testes em conjunto, isto é, do melhor cenário definido após a otimização combinada. Cada técnica foi aplicada isoladamente, mantendo arquitetura e hiperparâmetros fixos, o foco está no processo de adoção e integração ao fluxo de treino.

❑ **CutMix**

Nesta etapa, o *CutMix* foi inserido diretamente no fluxo de dados do TensorFlow, depois do batch e antes da entrada do modelo, ou seja, após as imagens já estarem agrupadas em lotes pelo pipeline de dados e imediatamente antes de entrarem na rede para o treinamento. Com isso, a mistura ocorre dentro do próprio lote (entre exemplos que já foram agrupados), preservando a eficiência do pipeline e evitando dependências externas ao treino.

Sendo assim, a cada lote, os exemplos são embaralhados para formar pares e, para

cada par, recorta-se um retângulo aleatório de uma imagem e cola-se esse recorte na outra. A mesma proporção usada para definir o tamanho do recorte também determina o peso de cada rótulo na mistura. Após posicionar o recorte sem ultrapassar as bordas, essa proporção é ajustada pela área efetivamente colada. Em seguida, os rótulos inteiros são convertidos para *one-hot* e combinados de forma ponderada por amostra: quanto maior a área colada, maior a contribuição do rótulo da imagem de origem do recorte.

O *CutMix* foi adotado para ampliar a variedade de exemplos vistos pelo ViT e incentivar o uso de *pistas contextuais* distribuídas pela imagem, reduzindo sobreajuste e favorecendo a generalização em CIFAR-100, em conformidade com (YUN et al., 2019; TOUVRON et al., 2021). A implementação empregada foi escrita diretamente em TensorFlow a partir da formulação original e segue o padrão amplamente difundido em exemplos públicos do Keras e em implementações de referência do ecossistema (como `timm`), o que reforça seu caráter de implementação comum e reprodutível (YUN et al., 2019; NATH, 2021; WIGHTMAN, 2019).

❑ Label Smoothing

Nesta etapa, o *Label Smoothing* foi aplicado diretamente na função de perda. O ajuste atua apenas no instante do cálculo da perda em cada lote, suavizando os alvos usados pelo critério, isto é, os vetores de rótulos que a função de perda de classificação compara às saídas numéricas do modelo antes de convertê-las em probabilidades. Em vez do padrão *one-hot*, passa-se a usar uma distribuição "*soft*": a probabilidade para a classe correta é reduzida e uma parte é redistribuída para as outras classes. Isso diminui uma confiança extrema nas saídas e favorece a estabilidade do treinamento, sem alterações na arquitetura. A técnica não altera como as métricas são calculadas, mas modifica os gradientes durante o treino e, portanto, o que o modelo aprende.

Em termos práticos, os rótulos verdadeiros em *one-hot* são suavizados: a classe correta recebe peso $(1 - \varepsilon)$ e as demais dividem ε igualmente. Com $\varepsilon = 0{,}1$ no CIFAR-100, a classe correta fica com 0,9 e o 0,1 restante é distribuído entre as outras 99 classes. Durante validação e teste, a suavização incide apenas no cálculo da *loss*, os rótulos do conjunto e as métricas permanecem inalterados.

A técnica foi adotada para mitigar sobreajuste, reduzir a superconfiança das previsões e melhorar a calibração das probabilidades. No cenário do CIFAR-100 (muitas classes parecidas e rótulos por vezes ruidosos), ela reduz confusão entre classes parecidas e o efeito de rótulos imprecisos, deixando as fronteiras de decisão mais suaves sem custo computacional adicional. A literatura reporta ganhos consistentes de generalização e estabilidade em arquiteturas modernas, inclusive Transformers, quando se utiliza o valor de ε modesto (como 0,1), o que sustenta a utilização desta escolha (SZEGEDY et al., 2016; MÜLLER; KORNBLITH; HINTON, 2019).

5 Experimentos e Resultados

Este capítulo apresenta a análise dos resultados obtidos a partir dos experimentos conduzidos com o modelo *Vision Transformer* (ViT) aplicado ao conjunto de dados *CIFAR-100*. O foco está em avaliar como diferentes escolhas de ambiente computacional, ajustes de hiperparâmetros e técnicas adicionais de otimização impactaram o desempenho e a robustez do modelo.

A organização segue o fluxo do desenvolvimento experimental, permitindo observar de forma progressiva a evolução do modelo: da comparação inicial com o estudo de referência, passando pela exploração de melhorias arquiteturais e computacionais, até a aplicação de estratégias avançadas de regularização.

5.1 Comparação entre Resultados em CPU

A primeira etapa da avaliação consistiu em comparar o desempenho do modelo *Vision Transformer* (ViT) utilizando o código original de (SILVA, 2024), executado em diferentes configurações de CPU, mas mantendo constantes os hiperparâmetros e o número de épocas. O objetivo foi analisar o impacto do hardware no tempo de treinamento, estabelecendo uma linha de base para as etapas posteriores de otimização.

Os experimentos foram realizados em duas máquinas distintas: (i) um notebook com processador Intel Core i5 de 7ª geração e 8 GB de RAM; e (ii) um notebook com processador Intel Core i7-7700HQ, também de 7ª geração, com 32 GB de RAM. Ambas utilizaram Python, Keras, Numpy, Tensorflow e o conjunto de dados CIFAR-100.

A Tabela 2 apresenta os hiperparâmetros adotados nos experimentos, mantidos constantes para ambas as máquinas.

	Hiperparâmetros	**Valor**
(1)	Learning Rate	0.0001
(2)	Weight Decay	0.0001
(3)	Épocas	20 / 30 / 50 / 100
(4)	Image Size	72
(5)	Batch Size	128
(6)	Patch Size	6
(7)	Multi-Headed Attention Heads	4
(8)	Transformer Layers	8

Tabela 2 – Hiperparâmetros utilizados nos experimentos de comparação em CPU.

A Tabela 3 apresenta os valores de acurácia e *top-5 accuracy* obtidos em cada configuração de hardware, relacionando-os ao tempo total de treinamento necessário em cada máquina. Como esperado, uma vez que a arquitetura e os hiperparâmetros foram mantidos constantes, os resultados de acurácia são bastante próximos. Pequenas diferenças observadas entre i5 e i7 não refletem influência direta do hardware na qualidade do modelo, mas sim ruídos associados à natureza estocástica do treinamento, como variações na inicialização dos pesos ou cálculos em ponto flutuante.

Tempo (min)	**Acurácia (i5)**	**Top-5 (i5)**	**Tempo (min)**	**Acurácia (i7)**	**Top-5 (i7)**
600	43.21%	73.91%	240	43.63%	74.33%
900	46.91%	76.10%	360	48.03%	76.75%
1500	50.23%	78.01%	600	52.14%	80.59%
3000	52.63%	79.77%	1200	54.31%	82.62%

Tabela 3 – Comparação entre resultados em CPU i5 (7ª geração) e CPU i7 (7ª geração), considerando tempo total de treinamento.

O principal diferencial observado foi o tempo médio por época, reduzido de aproximadamente 30 minutos no i5 para cerca de 12 minutos no i7, o que representa uma economia superior a 60%. Essa diferença está relacionada à maior quantidade de núcleos e memória RAM da máquina com i7, que reduziu gargalos de entrada e saída (I/O) e manteve o pipeline de treinamento mais estável.

Assim, embora os valores de acurácia sejam semelhantes para o mesmo número de épocas, a máquina equipada com i7 atinge esses resultados em muito menos tempo. Por exemplo, enquanto no i5 são necessários cerca de 1500 minutos para alcançar aproximadamente 50% de acurácia, no i7 o mesmo patamar é atingido em apenas 600 minutos. Essa diferença de desempenho temporal viabiliza execuções mais longas e múltiplos experimentos em prazos reduzidos, tornando o ambiente computacional um fator crítico para a praticidade do processo de pesquisa.

5.2 Integração e Treinamento com GPU

O treinamento de modelos de aprendizado profundo, especialmente arquiteturas complexas como o *Vision Transformer* (ViT), costuma ser acelerado por unidades de processamento gráfico (GPUs), uma vez que estas são projetadas para operações altamente paralelizáveis, como multiplicações de matrizes. Nesta seção, avalia-se o impacto da utilização da GPU disponível (NVIDIA GTX 1050 Ti) em relação ao treinamento exclusivamente em CPU, considerando tanto o tempo de execução quanto as limitações impostas pelo hardware.

Os testes foram conduzidos no mesmo notebook com processador Intel Core i7-7700HQ, 32 GB de RAM DDR4 e GPU NVIDIA GTX 1050 Ti (4 GB de VRAM). Para permitir a utilização da GPU com TensorFlow, foi necessário configurar o ambiente com *CUDA Toolkit*, *cuDNN* e uma versão compatível do TensorFlow.

Inicialmente, buscou-se utilizar o valor base do *batch size* de 128 e depois 112, mas ambos resultaram em erro de memória insuficiente (Out of Memory – OOM). Assim, foi necessária a redução progressiva do lote, com os seguintes resultados:

Configuração	**Tempo médio por época**
CPU (i7, batch 128)	≈ 12 minutos
GPU (GTX 1050 Ti, batch 96)	≈ 18 minutos
GPU (GTX 1050 Ti, batch 64)	≈ 14 minutos

Tabela 4 – Tempo médio por época no treinamento com GPU e CPU.

A Figura 7 apresenta a comparação entre os tempos médios por época obtidos com diferentes configurações de *batch size* na CPU (i7) e na GPU (GTX 1050 Ti). Observa-se que a CPU manteve desempenho consistentemente superior, mesmo para lotes menores, enquanto a GPU enfrentou limitações de memória de vídeo que aumentaram o tempo médio por época.

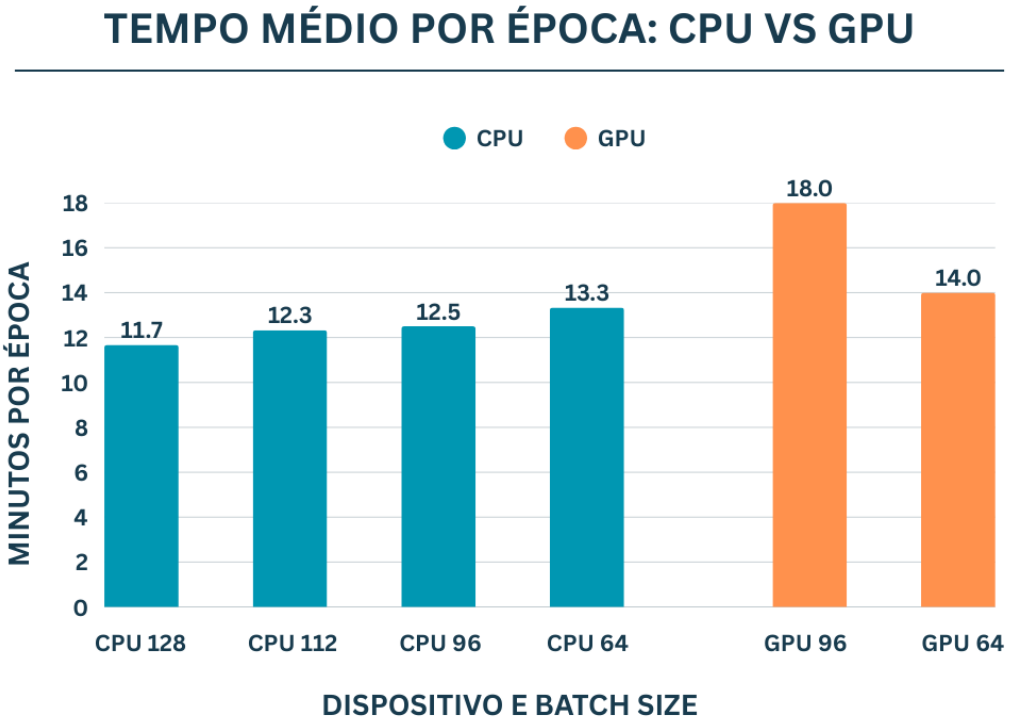


Figura 7 – Comparação do tempo médio de treinamento por época em CPU (i7) e GPU (GTX 1050 Ti). Fonte: autor.

Apesar da expectativa de aceleração, observou-se que a GPU não superou o desempenho da CPU. Enquanto a CPU aproveitou os 32 GB de RAM disponíveis para manipular lotes maiores (128 amostras) de forma eficiente, a GPU enfrentou limitações de memória de vídeo com seus 4 GB de VRAM, resultando em tempos mais altos de treinamento por época.

Diante desse comportamento, foi investigado de forma mais detalhada o uso da GPU durante os experimentos. Para isso, recorreu-se à ferramenta *nvidia-smi*, que permite monitorar em tempo real tanto a ocupação da memória de vídeo (VRAM) quanto a utilização computacional dos núcleos CUDA.

Monitoramento do Uso da GPU

A Figura 8 mostra a evolução temporal da utilização da GPU e do subsistema de memória durante o treinamento. Nela conseguimos observar picos sincronizados quando um lote é processado (GPU $\approx$ 90%, memória $\approx$ 75-80%) e longos intervalos de baixa utilização (GPU $\approx$ 5-10%, memória $\approx$ 0-10%). Ou seja, a GPU trabalha em rajadas e permanece ociosa entre lotes, indicando subalimentação do dispositivo, onde a VRAM de 4 GB impôs *batches* menores (64-96), aumentando o número de passos por época e o overhead por passo, enquanto as transferências host-device via PCIe introduziram latências adicionais.

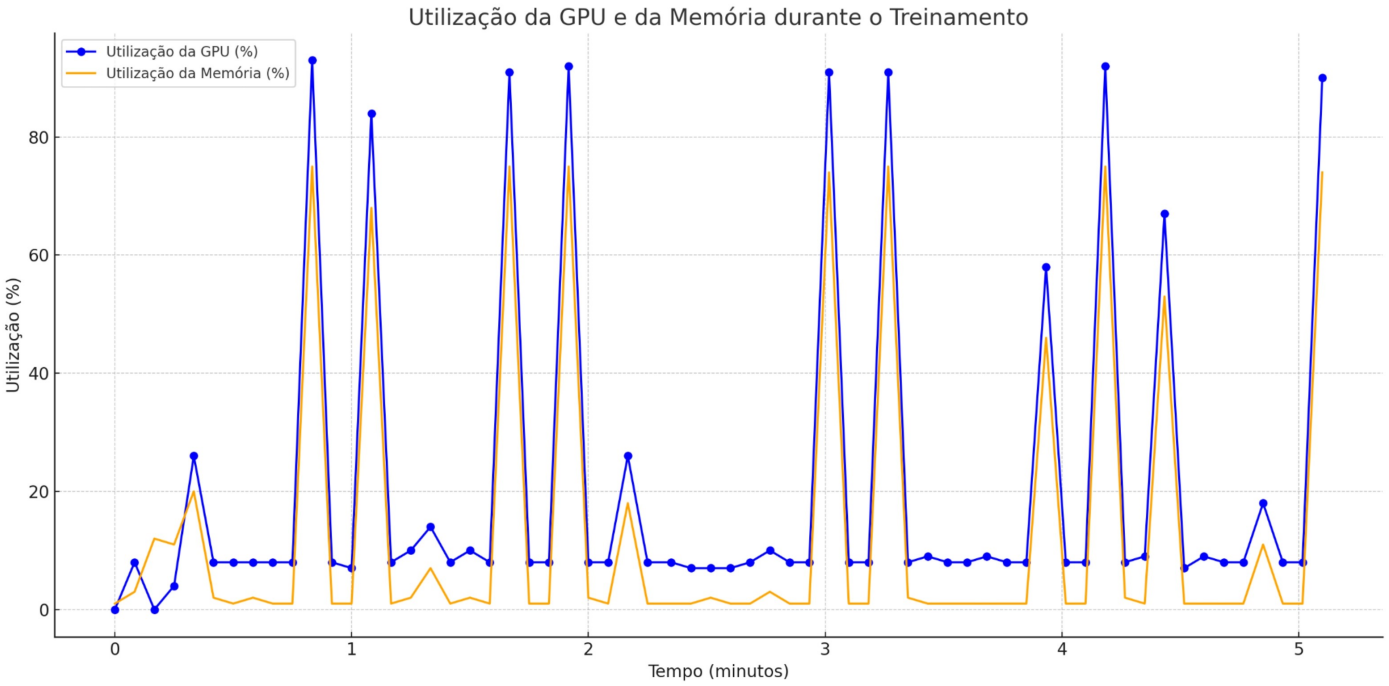


Figura 8 – Monitoramento de utilização da GPU e da memória VRAM durante o treinamento. Fonte: autor.

De forma prática, os fatores que explicam o aumento no tempo por época foram:

- **Memória de vídeo reduzida (4 GB):** obrigou a utilização de *batches* menores. Isso aumentou o número de passos necessários por época e elevou o custo de controle em cada iteração.
- **Largura de banda limitada do barramento PCIe:** a comunicação entre CPU e GPU foi mais lenta do que o necessário, fazendo com que a GPU passasse tempo aguardando a chegada de novos lotes de dados.
- **Subutilização dos núcleos CUDA:** a capacidade de processamento não foi plenamente explorada, pois os dados não chegavam de forma contínua, mantendo grande parte dos núcleos ociosos.

Depois de perceber que a GPU não reduziu o tempo por época, foram feitos alguns ajustes para tentar aumentar sua utilização. Esses ajustes diminuíram um pouco as esperas, mas o ganho foi pequeno:

- Implementação da sobreposição entre leitura/preparo de dados e o treinamento por meio do uso do `tf.data` com funções de embaralhamento, divisão em lotes e `prefetch`, de modo a manter uma fila de lotes prontos e reduzir a ociosidade entre lotes. A estratégia mitigou algumas pausas, mas o gargalo de transferência via PCIe e a limitação de *batch size* menor continuaram predominando.
- Realizou-se a separação manual do conjunto de validação, buscando liberar memória e evitar cópias internas durante o treino, o que permitiu a utilização de buffers maiores no pipeline. Embora tenha gerado alguma economia de recursos, essa medida mostrou-se insuficiente para possibilitar lotes maiores ou reduzir de forma significativa o tempo por época.
- Operações de normalização e *data augmentation* foram transferidas para camadas do Keras, antecipando as transformações para o grafo computacional. Com isso, reduziu-se parte do trabalho no host e o custo por passo, porém a estratégia apenas diminuiu o overhead pontual, sem eliminar a ociosidade decorrente de *batch size* reduzido e das transferências constantes entre CPU e GPU.

Em síntese, as otimizações aplicadas ajudaram a reduzir esperas pontuais do pipeline, mas não superaram os limites estruturais da GPU utilizada. Como consequência, o tempo médio por época permaneceu entre 14-18 minutos na GPU, enquanto na CPU i7, com *batch* 128, manteve cerca de 12 minutos por época. Diante desse resultado, as etapas seguintes de otimização de parâmetros e aplicação de técnicas adicionais foram realizadas exclusivamente em CPU, que se mostrou mais eficiente no contexto avaliado.

5.3 Análise de resultados da Otimização de Parâmetros

Esta seção apresenta e discute os resultados dos experimentos de otimização de hiperparâmetros do ViT sobre o *CIFAR-100*, tomando como referência a configuração base descrita na seção 4.4. O objetivo é quantificar ganhos de desempenho e o impacto no custo computacional, bem como interpretar os efeitos de cada ajuste realizado.

A análise está organizada em duas partes complementares. Primeiro, em **Testes Isolados**, foi alterado um único hiperparâmetro por vez, mantendo o restante do pipeline constante, para mapear a sensibilidade do modelo de forma controlada. Em seguida, em **Testes em Conjunto**, combinou-se os ajustes que se mostraram promissores de forma isolada, avaliando sinergias e eventuais compensações entre capacidade do modelo, políticas de *learning rate* e regularização.

Os resultados são reportados com foco em acurácia (top-1), *top-5* e *F1-score* macro, além de métricas operacionais como tempo por época e tempo total de treino.

5.3.1 Testes Isolados

Cada parâmetro foi alterado em relação à configuração base representado na seção 5.1, mantendo arquitetura, dados, semente e protocolo de treino constantes. Foram investigados *learning rate* (e seu agendamento), *weight decay*, *batch size*, *projection dim*, *num heads* e *transformer layers*. Os resultados são reportados em acurácia, *Top-5 Accuracy*, *F1-score* e custos computacionais, permitindo identificar tendências e faixas de maior benefício para cada parâmetro.

❑ Learning Rate

Os experimentos tiveram inicio com o *learning rate* partindo da configuração com valor fixo de $\eta = 0{,}001$, onde foi aplicado o *Cosine Annealing*. A Figura 9 ilustra o comportamento da taxa de aprendizado com esse agendador em um experimento com 50 épocas, observa-se a característica curva cossenoide, que promove uma redução progressiva e contínua da taxa de aprendizado desde o valor máximo até valores próximos de zero.

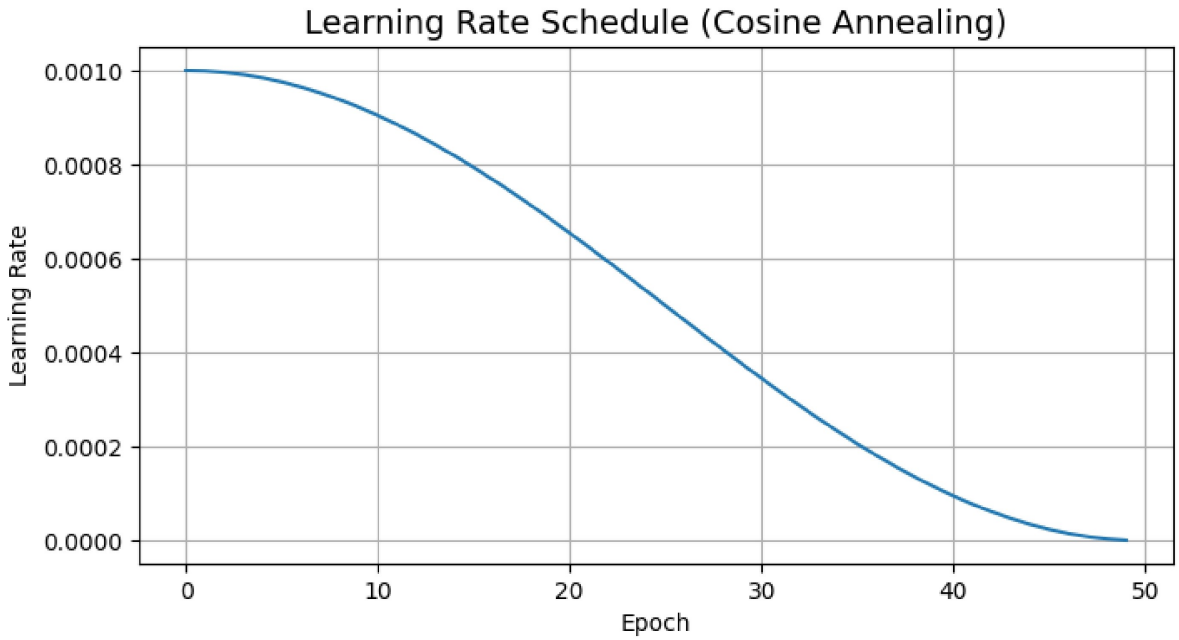


Figura 9 – Variação da *Taxa de Aprendizado*: 50 épocas com *Cosine Annealing*. Fonte: autor.

Na Tabela 5, observa-se que, apesar da suavização da convergência e uma pequena melhora na F1-Score no cenário de 50 épocas, a estratégia de *Cosine Annealing* não apresentou ganhos expressivos em relação à configuração com taxa de aprendizado fixa. Inclusive, nos experimentos com menor número de épocas (10 e 20), observou-se um desempenho inferior, sugerindo que a redução antecipada da taxa pode limitar a capacidade de exploração do modelo nos estágios iniciais do treinamento.

Épocas	**Cenário**	**Acurácia**	**Top-5 Accuracy**	**F1-score (macro)**
10	Fixo (0,001)	36,60%	67,11%	–
	Cosine Annealing	35,15%	65,30%	34,07%
20	Fixo (0,001)	43,63%	74,33%	–
	Cosine Annealing	41,24%	71,23%	40,50%
50	Fixo (0,001)	52,14%	80,59%	–
	Cosine Annealing	52,23%	80,21%	52,02%

Tabela 5 – Taxa fixa (0,001) vs. *Cosine Annealing* em 10, 20 e 50 épocas.

Uma hipótese levantada inicialmente foi a de que o valor adotado para a taxa de aprendizado máxima ($\eta_{\max} = 0{,}001$) poderia ser baixo demais, limitando o efeito exploratório nos estágios iniciais. Contudo, a revisão de literatura indicou que esse valor encontra-se na faixa usual para classificação no *CIFAR-100* com *Vision Transformers*.

Assim, concluiu-se que o fator limitante estava na ausência de aquecimento inicial (*warmup*). Iniciar diretamente com uma taxa relativamente alta na primeira época pode gerar instabilidades, especialmente em arquiteturas sensíveis como Transformers, que não possuem vieses indutivos espaciais fortes (como convoluções locais) capazes de estabilizar a fase inicial do aprendizado. Na prática, isso pode resultar em explosões de gradiente ou atualizações ruidosas, prejudicando a aprendizagem de representações úteis logo de início.

Com base nessa fundamentação, implementou-se um agendamento com *warmup* + *Cosine Annealing*. A estratégia consistiu em iniciar a taxa de aprendizado em `1e-6` e elevá-la linearmente ao longo das 5 primeiras épocas até alcançar 0,001. A partir daí, o *Cosine Annealing* passou a controlar o decaimento progressivo até o fim do treinamento. A Figura 10 ilustra essa estratégia, onde temos uma rampa inicial suave (warmup) seguida da curva cossenoide de redução.

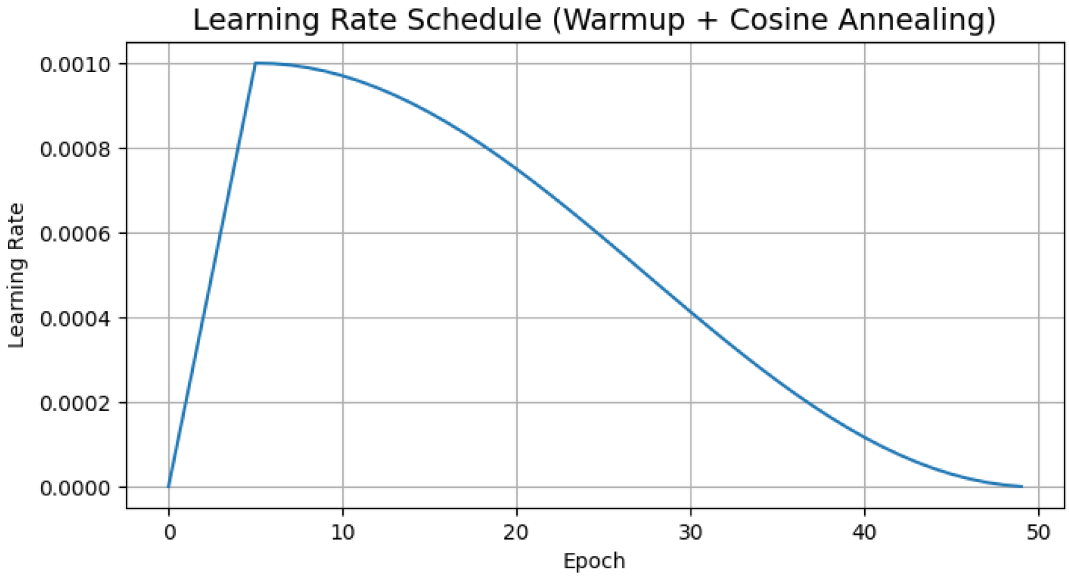


Figura 10 – Evolução da *Taxa de Aprendizado* com *Warmup* seguido de *Cosine Annealing* (até 50 épocas). Fonte: autor.

Os resultados comparativos em 50 épocas entre as três configurações (taxa fixa, *Cosine Annealing* sem *warmup* e *warmup* + *Cosine Annealing*) são apresentados na Tabela 6.

Configuração (50 épocas)	**Acurácia**	**Top-5 Accuracy**	**F1-score (macro)**
LR estático (0,001)	52,14%	80,59%	–
Cosine Annealing ($\eta_{\max} = 0{,}001$)	52,23%	80,21%	52,02%
Warmup + Cosine Annealing ($\eta_{\max} = 0{,}001$)	**54,87%**	**81,84%**	**54,59%**

Tabela 6 – Comparação entre taxa fixa, *Cosine Annealing* e *Warmup + Cosine Annealing* em 50 épocas.

Observa-se ganho consistente em acurácia, o *Top-5 Accuracy* atinge 81,84% com *warmup*, frente a 80,21% sem *warmup*, evidenciando ganho de generalização multiclasse, isto é, melhor capacidade de reconhecer diversas categorias e ranquear a correta entre as primeiras posições, e o *F1-score* salta de 52,02% para 54,59% com *warmup*, sinalizando maior equilíbrio entre classes, aspecto crucial no *CIFAR-100*, onde é comum haver disparidades de acerto entre categorias.

Com base nos resultados, foi adotado *warmup + Cosine Annealing* como política de taxa de aprendizado para os experimentos dos testes em conjunto, por oferecer ganho em todas as métricas.

A seguir, na Tabela 7, apresentam-se os resultados dos demais hiperparâmetros avaliados de forma isolada. Optou-se por reuni-los em uma única tabela, pois, diferentemente do *learning rate*, não houve aplicação de estratégias dinâmicas de variação.

Todos os experimentos com os demais hiperparâmetros foram executados com **50 épocas**, seguindo a metodologia de alterar apenas um por vez. Após cada teste, o valor modificado era retornado ao seu padrão base, garantindo que a avaliação seguinte fosse conduzida sem interferências. Dessa forma, assegura-se que os resultados reflitam exclusivamente o impacto do hiperparâmetro em análise, mantendo os demais parâmetros fixos em seus valores de referência.

Parâmetro	Configuração	Acurácia	Top-5 Acc.	Tempo/época
Base (referência)		52,14%	80,59%	12m30s
weight decay	0,00001	50,95%	79,54%	13m
weight decay	0,001	38,05%	67,12%	13m
batch size	256	52,67%	80,31%	11m30s
batch size	64	46,51%	75,41%	13m
projection dim	96	49,34%	77,14%	19m
projection dim	128	-	-	25m
num heads	6	34,28%	63,17%	15m30s
transformer layers	10	49,21%	77,77%	14m30s

Tabela 7 – Resultado dos testes isolados (50 épocas) para hiperparâmetros.

❑ Parâmetro de regularização (*weight decay*)

Foram testados dois extremos em relação ao valor base (0,0001): menor (0,00001) e maior (0,001). O valor reduzido apresentou leve queda (50,95% de acurácia e 79,54% de *Top-5 Accuracy*), mantendo tempo similar (13m), efeito típico de **sub-regularização**, com maior tendência a sobreajuste. Já o valor elevado resultou em queda acentuada (38,05% / 67,12%), caracterizando **underfitting**, pois a penalização passou a restringir excessivamente a aprendizagem dos pesos.

Nesse sentido, o valor base de 0,0001 mostrou-se o mais adequado neste cenário de 50 épocas. Novas variações só se justificariam em expansões estruturais do modelo, como aumento do *projection dim*, mais *transformer layers* ou ampliação das *mlp head units*.

❑ Tamanho do lote (*batch size*)

Foram testados 64 e 256 em comparação ao valor base (128). O lote 64 apresentou pior desempenho (46,51% / 75,41%), resultado de mais passos por época e gradientes mais ruidosos, que comprometeram a estabilidade do treinamento, porém, manteve tempo similar (13m). Já o lote 256 reduziu o tempo médio por época (11m30s) e trouxe leve aumento de acurácia (52,67%), com *Top-5 Accuracy* praticamente estável (80,31%).

Assim, lotes pequenos mostraram-se prejudiciais, enquanto lotes maiores trouxeram ganhos computacionais sem melhora expressiva de generalização, possivelmente devido a limitações estruturais do modelo. O valor base (128) permanece como escolha equilibrada, porém o valor de 256 foi adotado para seguir nos testes em conjunto por priorizar eficiência computacional.

❑ **Dimensão de projeção (*projection dim*)**

Foram testados valores de 96 e 128 em relação ao base (64). Com 96, houve queda de acurácia e *Top-5 Accuracy* (49,34% / 77,14%) e grande aumento no tempo (19m/época). O teste em 128 foi interrompido (*tempo estimado* de 25m/época), pois a tendência já desfavorável em 96 indicava baixo potencial de ganho e maior risco de **overfitting**.

A análise sugere duas causas principais para o insucesso do aumento isolado: (i)subutilização da capacidade extra, já que a estrutura do modelo não foi ajustada em paralelo, e (ii) falta de mecanismos de controle da maior capacidade, o que intensificou a tendência ao sobreajuste e reduziu a generalização. Desse modo, o valor 64 permanece como o mais adequado isoladamente, equilibrando eficiência e desempenho.

❑ **Número de cabeças de atenção (*num heads*)**

O aumento para 6 cabeças, mantendo $projection\_dim = 64$, reduziu a dimensão por cabeça para apenas $\approx 10{,}7$ ($64/6$). O resultado foi uma queda severa no desempenho (34,28% / 63,17%) e maior tempo por época (15m30s). Isso confirma que dividir a mesma projeção em mais cabeças sacrificou a expressividade individual, comprometendo a atenção e a generalização. Sendo assim, o valor 4 cabeças permanece como melhor equilíbrio entre capacidade e estabilidade computacional no cenário atual.

❑ **Número de camadas Transformer (*transformer layers*)**

O aumento de 8 para 10 camadas reduziu o desempenho (49,21%/ 77,77%) e elevou o tempo por época para 14m30s. As causas prováveis incluem: (i) overfitting leve, pela maior capacidade sem aumento correspondente de regularização; (ii) saturação de profundidade, dado que o modelo já se mostrava suficientemente profundo para CIFAR-100; e (iii) ausência de ajustes complementares (como *dropout*, *mlp_head_units* ou *weight decay*).

A configuração com 8 camadas segue como a mais adequada, e sua ampliação só deve ser considerada em conjunto com alterações estruturais mais amplas, analisadas posteriormente nos testes combinados.

5.3.2 Testes em Conjunto

Após a análise isolada de cada hiperparâmetro, observou-se que alterações individuais nem sempre resultavam em melhorias significativas de desempenho. Por esse motivo, foram definidos grupos de testes com combinações específicas, fundamentadas na literatura e nos resultados anteriores.

Todos os experimentos desta seção foram realizados com learning rate dinâmico com warmup seguido de cosine annealing, técnica que superou o uso de *learning rate* fixo,

conforme demonstrado na Tabela 6.

❑ Grupo 1 — Ampliação da Capacidade de Saída

Nesse grupo o objetivo foi valiar se a combinação de *batch size* maior, aumento da profundidade e ampliação da MLP final resulta em melhor desempenho. Essas foram as configurações utilizadas:

- $batch\_size = 256$
- $transformer\_layers = 10$
- $mlp\_head\_units = [3072, 2048]$
- Learning rate dinâmico (warmup + cosine annealing)

O aumento de profundidade amplia a capacidade hierárquica de aprendizado, MLPs maiores aumentam o poder discriminativo e o *batch size* = 256 já havia mostrado eficiência computacional e estabilidade. Assim, o modelo teria maior capacidade, o que tende a se beneficiar de estratégias de *learning rate* dinâmico, evitando sobreajuste inicial.

A configuração apresentou o melhor resultado até então (55,01% de acurácia), com 54,73% F1-score, e uma leve queda para 81,74% no *Top-5 Accuracy*, que está dentro da margem de erro esperada. A curva de treino mostrou tendência a sobreajuste após 20 épocas, mas em intensidade moderada, justificando futuras estratégias de regularização.

❑ Grupo 2 — Regularização em Modelo Otimizado

Esse grupo teve como objetivo avaliar se o aumento do *weight decay* poderia conter o sobreajuste observado no Grupo 1, mantendo o bom desempenho do modelo. A configuração se baseou no Grupo 1, alterando apenas o *weight_decay* para 0.0005.

No Grupo 1, a partir da época 20, a acurácia de treino continuava subindo enquanto a validação estabilizava, sinal de sobreajuste leve. Segundo (LOSHCHILOV; HUTTER, 2018), ajustes sutis nesse hiperparâmetro atuam como regularização L2, penalizando pesos excessivamente grandes e ajudando a estabilizar a generalização sem comprometer a performance. Também foram feitas tentativas de ampliar a capacidade por meio do *projection dim*, mas o aumento substancial no tempo por época tornou essa linha de exploração inviável dentro do escopo do trabalho.

Entretanto, o aumento do *weight decay* para 0.0005 não trouxe ganhos em relação ao Grupo 1: o modelo manteve desempenho praticamente idêntico (55,01% de acurácia,

54,75% de F1-score e 81,72% de *Top-5 Accuracy*). Isso ocorreu porque a configuração do Grupo 1 já apresentava regularização suficiente para conter o sobreajuste leve observado, de modo que a penalização adicional não teve efeito prático sobre a generalização. Nesse sentido, como o Grupo 2 não superou o desempenho do Grupo 1, a configuração do Grupo 1 foi mantida como **melhor resultado final**, equilibrando acurácia, F1-score e custo computacional.

5.4 Técnicas de Otimização e Regularização

Modelos de *Deep Learning*, em especial arquiteturas como o Vision Transformer, estão sujeitos a problemas de sobreajuste e instabilidade durante o treinamento. Nesse contexto, técnicas de otimização e regularização têm papel central para melhorar a capacidade de generalização, reduzir a confiança excessiva em rótulos específicos e acelerar a convergência. Entre as abordagens investigadas neste trabalho, destacam-se o *CutMix*, que introduz amostras compostas a partir da sobreposição de imagens, e o *Label Smoothing*, que suaviza os rótulos-alvo para mitigar previsões excessivamente confiantes. A seguir, são apresentados os resultados dessas duas técnicas aplicadas de maneira isolada.

❑ CutMix

O *CutMix* foi aplicado isoladamente à melhor configuração consolidada de hiperparâmetros definida nos testes em conjunto na subseção 5.3.2. Essa validação independente é importante para mensurar o impacto direto da técnica sem interferência de outros fatores, permitindo avaliar se os ganhos observados na literatura se reproduzem neste cenário específico.

Após a implementação, o treinamento foi realizado por 50 épocas para permitir uma comparação direta com a melhor configuração encontrada nos testes em conjunto. Os resultados mostraram uma queda expressiva de desempenho, com apenas 32,97% de acurácia, 63,82% de *Top-5 Accuracy* e 30,33% de F1-score macro, além de um tempo médio de 17 minutos por época.

Na Figura 11, observa-se que a perda de treino apresenta uma queda inicial acentuada nas primeiras épocas (de aproximadamente 7,0 para 4,5), mas logo em seguida estagna, permanecendo em torno de 4,0 até o final das 50 épocas. Esse comportamento indica que o modelo não conseguiu aprender representações consistentes a partir dos dados modificados pelo *CutMix*, que, no caso do CIFAR-100, tornam-se altamente ambíguos devido à baixa resolução das imagens. Em contraste, a perda de validação reduz-se de forma mais contínua, estabilizando próxima de 3,0. Essa divergência sugere que, embora o modelo consiga reduzir parcialmente o erro sobre os exemplos originais do conjunto de teste, o aprendizado em treino permaneceu limitado, refletindo a dificuldade do ViT em

lidar com os rótulos mistos e recortes aplicados durante o treinamento.

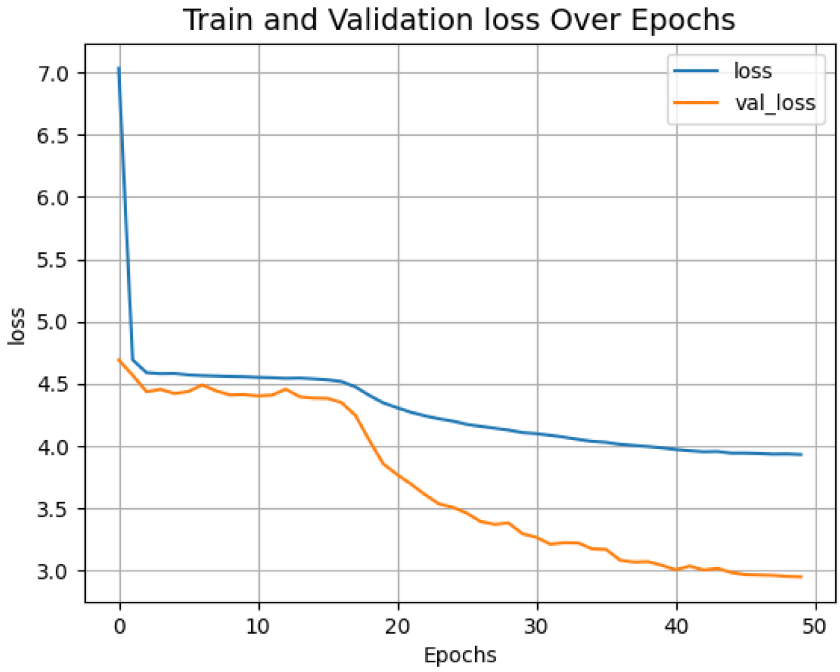


Figura 11 – Evolução da função de perda (*loss*) em treino e validação ao longo de 50 épocas com a técnica *CutMix*. Fonte: autor

A Figura 12 reforça esse padrão ao analisar a *Top-5 Accuracy*. Enquanto a acurácia de treino cresce de forma lenta e limitada, estabilizando abaixo de 0,30, a validação dispara a partir da época 20, alcançando cerca de 0,64 ao final. Essa discrepância evidencia que o modelo passou a explorar pistas parciais dos recortes para acertar "por aproximação" no conjunto de validação, sem consolidar um aprendizado robusto em treino. Em outras palavras, a técnica induziu o ViT a reconhecer fragmentos das imagens misturadas, mas sem desenvolver representações gerais e estáveis das classes, o que explica o baixo desempenho global. Essa falsa sensação de ganho em validação, não acompanhada por evolução em treino, confirma que o *CutMix* não se mostrou eficaz neste cenário.

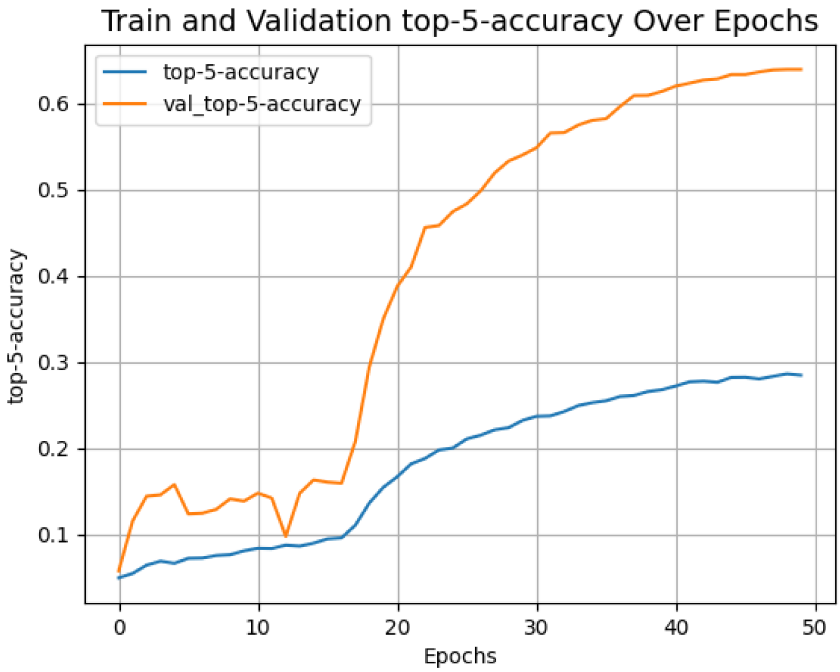


Figura 12 – Evolução da *Top-5 Accuracy* em treino e validação ao longo de 50 épocas com a técnica *CutMix*. Fonte: autor

Essa degradação pode ser explicada por fatores específicos do contexto:

- A baixa resolução das imagens do CIFAR-100 faz com que os recortes eliminem partes essenciais dos objetos, comprometendo a informação disponível para o modelo.
- O grande número de classes visuais semelhantes aumenta a chance de gerar imagens ambíguas quando misturadas, confundindo a rede durante o aprendizado.
- O treinamento foi limitado a 50 épocas, o que pode não ser suficiente para que o modelo estabilize o aprendizado frente a rótulos mistos.

Dessa forma, a técnica não apresentou benefícios no cenário adotado, resultando em perdas significativas nas métricas de avaliação. Considera-se que o *CutMix* tende a ser mais adequado em contextos com imagens de maior resolução, menor número de classes similares e treinamentos mais longos. Assim, por não se mostrar vantajoso dentro dos limites e objetivos deste trabalho, optou-se por não utilizá-lo nas etapas subsequentes.

❑ Label Smoothing

Após o treinamento com o *CutMix* não ter gerado resultados satisfatórios, explorou-se a técnica de *Label Smoothing*, também avaliada isoladamente sobre a configuração

consolidada de hiperparâmetros definida nos testes em conjunto. Essa técnica suaviza os rótulos verdadeiros, reduzindo a confiança excessiva do modelo em uma única classe.

Os resultados mostraram ganhos consistentes em relação ao cenário base. Com 50 épocas, o modelo alcançou 55,15% de acurácia, 81,81% de *Top-5 Accuracy* e 54,91% de F1-score macro, superando a configuração de referência apresentada na subseção 5.3.2. Embora os avanços absolutos sejam modestos, evidenciam que, além de conter o sobreajuste, o *Label Smoothing* contribuiu para acelerar a estabilização das curvas de treino, resultando em métricas superiores.

A Figura 13 apresenta a evolução da função de perda. Nota-se que tanto treino quanto validação apresentam reduções consistentes e próximas, sem a discrepância observada no *CutMix*. A perda de treino cai de mais de 7,0 no início para cerca de 1,5 ao final, enquanto a validação estabiliza em torno de 2,3. Essa proximidade entre as curvas demonstra que o *Label Smoothing* reduziu o sobreajuste, forçando o modelo a manter um aprendizado mais equilibrado e estável entre os dois conjuntos.

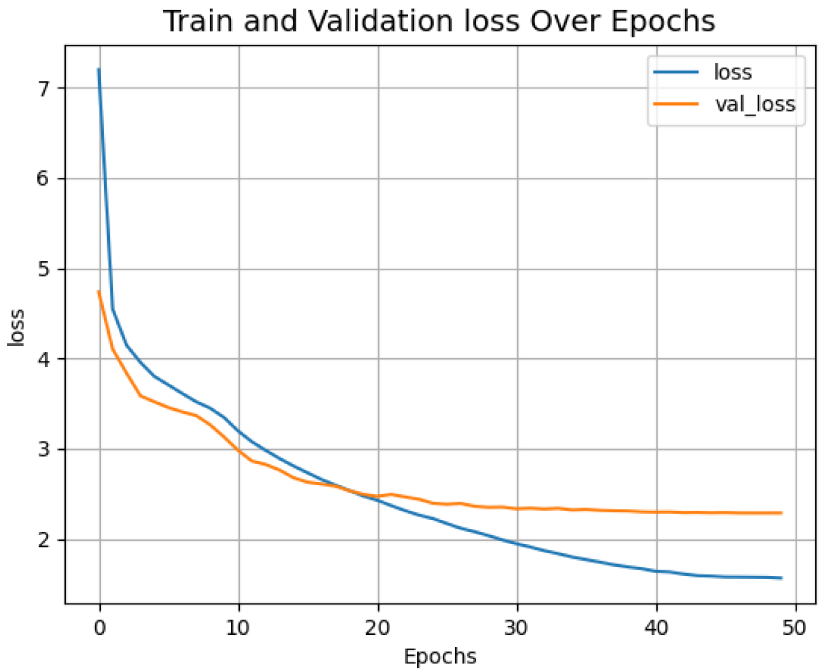


Figura 13 – Evolução da função de perda (*loss*) em treino e validação ao longo de 50 épocas com a técnica *Label Smoothing*. Fonte: autor

A Figura 14 reforça esse comportamento ao analisar a *Top-5 Accuracy*. Ambas as curvas crescem de forma rápida até a época 20, atingindo aproximadamente 0,80, e mantêm tendência de estabilização em seguida. O treino continua subindo até próximo de 0,98, enquanto a validação estabiliza em torno de 0,83. Essa diferença moderada indica que o modelo conseguiu aprender representações gerais mais sólidas sem perder capacidade de

generalização. Assim, o *Label Smoothing* contribuiu não apenas para reduzir a confiança excessiva do modelo nos rótulos, mas também para tornar o aprendizado mais uniforme, refletindo nos ganhos de acurácia, *Top-5 Accuracy* e F1-score macro.

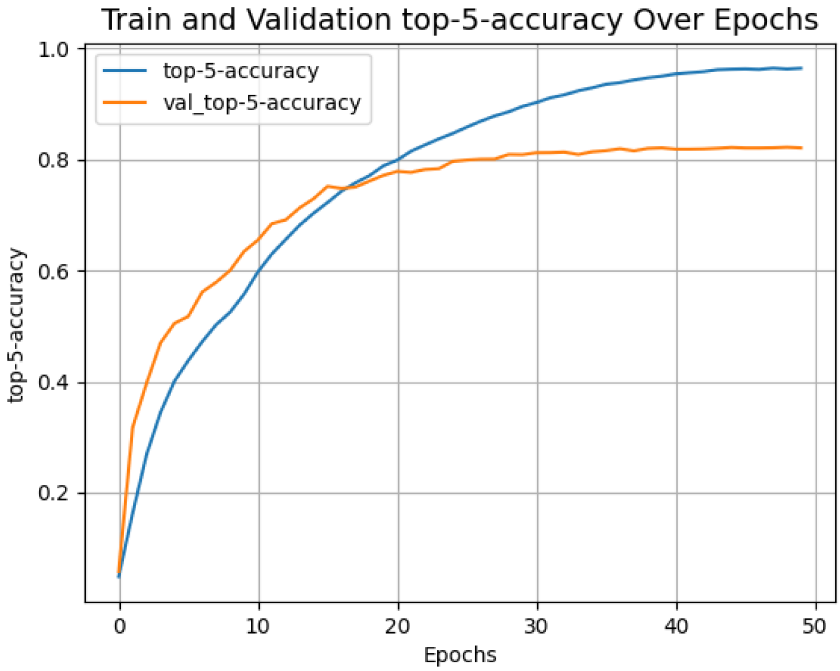


Figura 14 – Evolução da *Top-5 Accuracy* em treino e validação ao longo de 50 épocas com a técnica *Label Smoothing*. Fonte: autor

A análise das curvas de perda e acurácia indica que a técnica atuou como uma forma de regularização: reduziu a oscilação entre treino e validação e proporcionou maior consistência ao aprendizado ao longo das épocas. Esse efeito foi especialmente relevante no *F1-score* macro, que apresentou aumento significativo, demonstrando melhor equilíbrio entre as classes do CIFAR-100, onde a distribuição é altamente desbalanceada. Outro ponto positivo é que não houve impacto negativo relevante no tempo de execução por época, preservando a viabilidade computacional do modelo.

Dessa forma, diferentemente do *CutMix*, a aplicação isolada do *Label Smoothing* demonstrou ganhos efetivos, tanto em acurácia quanto em robustez das métricas. Por esse motivo, foram realizados testes adicionais com 20, 30, 50 e 100 épocas, permitindo comparar a progressão das métricas com a configuração base inicial na CPU (i7) e avaliar de forma mais aprofundada a evolução da técnica ao longo do treinamento.

Épocas	**Configuração Base (i7)**		**Label Smoothing (i7)**	
	Acurácia	**Top-5**	**Acurácia**	**Top-5**
20	43.63%	74.33%	44.46%	73.60%
30	48.03%	76.75%	49.98%	79.07%
50	52.14%	80.59%	55.15%	81.81%
100	54.31%	82.62%	57.34%	82.83%

Tabela 8 – Comparação entre a configuração base e a aplicação do *Label Smoothing* no CPU i7 (7ª geração).

A Tabela 8 evidencia que o *Label Smoothing* proporcionou ganhos consistentes em todas as métricas avaliadas. Nos testes de 30, 50 e 100 épocas, observa-se que a acurácia apresentou avanços superiores a 2 pontos percentuais em relação ao cenário base, enquanto a *Top-5 Accuracy* também se manteve em patamar ligeiramente mais alto. A principal contribuição, contudo, está na estabilidade do aprendizado: as curvas de treino e validação permaneceram próximas ao longo do processo, reduzindo o risco de sobreajuste e resultando em maior equilíbrio entre as classes, refletido no aumento do F1-score macro.

Em comparação com a configuração base, o incremento acumulado foi de aproximadamente 3 pontos percentuais em acurácia, consolidando o *Label Smoothing* como uma estratégia vantajosa neste trabalho.

6 Considerações Finais

6.1 Conclusão

O presente trabalho investigou o *Vision Transformer* (ViT) aplicado ao conjunto de dados *CIFAR-100*, partindo de um estudo-base e buscando superar suas limitações por meio da análise de ambientes de execução distintos, ajustes de hiperparâmetros e aplicação de técnicas de regularização. O objetivo central foi compreender como o ViT se comporta em cenários de recursos computacionais modestos, sem pré-treinamento em grandes bases, e oferecer um processo reprodutível que equilibre desempenho, estabilidade e viabilidade prática, cujo código-fonte se encontra disponível publicamente para consulta e reprodução dos experimentos realizados (CURTOLO, 2025).

A pesquisa mostrou que a escolha do hardware exerce influência significativa tanto no tempo de execução quanto na qualidade do treinamento, destacando que CPUs com maior memória e paralelismo podem superar GPUs de entrada em determinadas situações. No campo da otimização, confirmou-se a relevância de políticas dinâmicas de *learning rate*, em especial a combinação de *warmup* com *Cosine Annealing*, que favoreceu a estabilidade do treinamento e melhores resultados. Também foram identificadas faixas seguras de hiperparâmetros e validada a utilidade do *Label Smoothing* como técnica eficaz e de baixo custo para melhorar a generalização. Em contrapartida, o *CutMix* não se mostrou adequado no contexto do *CIFAR-100*, evidenciando a necessidade de cautela na adoção de técnicas de aumento de dados.

Ainda que apresente contribuições relevantes, este trabalho possui limitações. A utilização de apenas uma GPU de entrada restringiu a avaliação de cenários mais robustos de aceleração, e o escopo experimental não contemplou variações arquiteturais profundas no ViT, como alterações no tamanho de *patches* ou na largura das camadas internas. Além disso, o uso de um conjunto de dados relativamente pequeno limita a generalização dos achados para contextos mais complexos.

6.2 Trabalhos Futuros

Para trabalhos futuros, destacam-se algumas possibilidades. Uma primeira linha de investigação consiste na avaliação de técnicas adicionais de regularização e aumento de dados, como *Cutout* e *Mixup*, que podem complementar ou até superar o *Label Smoothing* em termos de robustez do modelo. Nesse sentido, torna-se relevante comparar essas abordagens de forma sistemática, investigando possíveis sinergias e compensações quando

aplicadas em conjunto.

Outra possibilidade envolve a exploração de arquiteturas variantes do ViT, incluindo modelos híbridos que combinem camadas convolucionais iniciais ou que ampliem a profundidade e a largura da rede de forma controlada. Além disso, recomenda-se expandir os experimentos para ambientes computacionais mais potentes, o que permitirá utilizar valores de hiperparâmetros diversos, arquiteturas mais complexas e regimes de treinamento mais longos, além de avaliar o comportamento do ViT em bases de dados de maior escala.

Essas perspectivas visam não apenas aprofundar os resultados deste trabalho, mas também ampliar o entendimento sobre a aplicação prática do ViT em cenários restritos e de maior escala, consolidando a arquitetura como alternativa viável e eficiente para diferentes contextos de visão computacional.

Referências

CURTOLO, B. **TCC - Vision Transformer Optimization on CIFAR-100**. 2025. <https://github.com/brennocurtolo/TCC>. Acessado em: 19 set. 2025. Citado na página 56.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020. Disponível em: <https://arxiv.org/pdf/2010.11929.pdf>. Citado 16 vezes nas páginas 4, 9, 15, 16, 19, 21, 22, 23, 24, 25, 29, 30, 31, 34, 35 e 36.

GeeksforGeeks. **Vision Transformers vs. Convolutional Neural Networks (ViTs vs CNNs)**. 2025. Website. Acesso em: 12 ago. 2025. Disponível em: <https://www.geeksforgeeks.org/deep-learning/vision-transformers-vs-convolutional-neural-networks-cnns/>. Citado na página 13.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. ISBN 9780262035613. Disponível em: <https://www.deeplearningbook.org>. Citado 3 vezes nas páginas 9, 17 e 20.

GOYAL, P.; DOLLáR, P.; GIRSHICK, R.; NOORDHUIS, P.; WESOLOWSKI, L.; KYROLA, A.; TULLOCH, A.; JIA, Y.; HE, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 19 e 34.

HAN, K.; XIAO, A.; WU, E.; GUO, J.; XU, C.; WANG, Y. Transformer in transformer. **Advances in Neural Information Processing Systems**, v. 34, p. 15908–15919, 2021. Disponível em: <https://proceedings.neurips.cc/paper/2021/hash/854d9fca60b4b0ed439b3f47e8ff8e5f-Abstract.html>. Citado na página 22.

KOLESNIKOV, A.; ZHAI, X.; HOULSBY, N.; BEYER, L.; UNTERTHINER, T.; HEIGOLD, G.; USZKOREIT, J.; DOSOVITSKIY, A. Image classification with vision transformers: A comparative study. **arXiv preprint arXiv:2103.02989**, 2021. Disponível em: <https://arxiv.org/abs/2103.02989>. Citado na página 21.

KRIZHEVSKY, A.; HINTON, G. **Learning Multiple Layers of Features from Tiny Images**. [S.l.], 2009. Disponível em: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. Citado 2 vezes nas páginas 9 e 16.

LOSHCHILOV, I.; HUTTER, F. **SGDR: Stochastic Gradient Descent with Warm Restarts**. 2016. Disponível em: <https://arxiv.org/abs/1608.03983>. Citado na página 18.

_____. Decoupled weight decay regularization. In: **International Conference on Learning Representations (ICLR)**. [s.n.], 2018. Disponível em: <https://arxiv.org/abs/1711.05101>. Citado 2 vezes nas páginas 20 e 49.

MORAIS, M. **Colorful brown black and white shaded cat**. 2023. Pexels License; imagem modificada para fins ilustrativos. Disponível em: <https://www.pexels.com/photo/colorful-brown-black-and-white-shaded-cat-15529573/>. Citado 2 vezes nas páginas 4 e 16.

MÜLLER, R.; KORNBLITH, S.; HINTON, G. When does label smoothing help? In: **Advances in Neural Information Processing Systems (NeurIPS)**. [s.n.], 2019. p. 4696–4705. Disponível em: <https://arxiv.org/abs/1906.02629>. Citado na página 37.

MüLLER, R.; KORNBLITH, S.; HINTON, G. E. When does label smoothing help? **arXiv preprint arXiv:1906.02629**, 2019. Disponível em: <https://arxiv.org/abs/1906.02629>. Citado 2 vezes nas páginas 4 e 27.

NATH, S. **CutMix data augmentation for image classification**. 2021. <https://keras.io/examples/vision/cutmix/>. Keras Code Example; last modified 2023-11-14; accessed 2025-08-25. Citado na página 37.

OPENAI. **ChatGPT**. 2025. Disponível em: <https://chat.openai.com/>. Acesso em: 30 ago. 2025. Nenhuma citação no texto.

Pexels. **A Red Cat With a Collar On Looking Up**. 2024. Acesso em: 11 ago. 2025. Disponível em: <https://www.pexels.com/photo/a-red-cat-with-a-collar-on-looking-up-19435364/>. Citado 2 vezes nas páginas 4 e 26.

_____. **Calm Labrador Resting Indoors in Brazil**. 2025. Acesso em: 11 ago. 2025. Disponível em: <https://www.pexels.com/photo/calm-labrador-resting-indoors-in-brazil-31220357/>. Citado 2 vezes nas páginas 4 e 26.

SAGLAM, K. **Vintage white Volkswagen Polo in nature setting**. 2025. Pexels License; imagem modificada para fins ilustrativos. Disponível em: <https://www.pexels.com/photo/vintage-white-volkswagen-polo-in-nature-setting-33351411/>. Citado 2 vezes nas páginas 4 e 24.

SAP Portugal. **O que é aprendizagem profunda?** 2025. Acesso em: 12 ago. 2025. Disponível em: <https://www.sap.com/portugal/resources/what-is-deep-learning>. Citado na página 13.

SILVA, R. Z. **Vision Transformers: Um estudo da arquitetura Transformer aplicada à Classificação de Imagens**. Monografia (Monografia de Graduação) — Universidade Federal de Uberlândia, Uberlândia, MG, Brasil, 2024. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Código disponível em: <https://github.com/rodrigozamb/TCC/blob/master/TCC_Rodrigo_Zamboni_Final.pdf>. Citado 8 vezes nas páginas 9, 10, 29, 30, 31, 33, 34 e 38.

SINGH, S. **Panda bear on green grass**. 2020. Pexels License; imagem modificada para fins ilustrativos. Disponível em: <https://www.pexels.com/photo/panda-bear-on-green-grass-3608263/>. Citado 2 vezes nas páginas 4 e 22.

STEINER, A.; KOLESNIKOV, A.; ZHAI, X.; WIGHTMAN, R.; USZKOREIT, J.; BEYER, L. How to train your vit? data, augmentation, and regularization in

vision transformers. **arXiv preprint arXiv:2106.10270**, 2021. Disponível em: <https://arxiv.org/pdf/2106.10270.pdf>. Citado na página 35.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2016. p. 2818–2826. Disponível em: <https://openaccess.thecvf.com/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf>. Citado 2 vezes nas páginas 27 e 37.

TOUVRON, H.; CORD, M.; DOUZE, M.; MASSA, F.; SABLAYROLLES, A.; JEGOU, H. Training data-efficient image transformers & distillation through attention. In: PMLR. **Proceedings of the 38th International Conference on Machine Learning**. 2021. p. 10347–10357. Disponível em: <https://proceedings.mlr.press/v139/touvron21a.html>. Citado 7 vezes nas páginas 21, 23, 25, 34, 35, 36 e 37.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [s.n.], 2017. Disponível em: <https://arxiv.org/pdf/1706.03762.pdf>. Citado 11 vezes nas páginas 4, 9, 14, 15, 23, 24, 27, 29, 30, 35 e 36.

WIGHTMAN, R. **PyTorch Image Models**. [S.l.]: GitHub, 2019. <https://github.com/rwightman/pytorch-image-models>. Citado na página 37.

YUN, S.; HAN, D.; OH, S. J.; CHUN, S.; CHOE, J.; YOO, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. p. 6023–6032. Citado 2 vezes nas páginas 26 e 37.