UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

Classificação de imagens de moda utilizando modelos de aprendizado profundo com pré-treinamento contrastivo multimodal

Vinicius Cin

Florianópolis, 2023

Vinicius Cin

Classificação de imagens de moda utilizando modelos de aprendizado profundo com pré-treinamento contrastivo multimodal

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Engenharia Eletrônica da Universidade Federal de Santa Catarina para a obtenção do Título de Bacharel em Engenharia Eletrônica.
Orientador: Prof. Danilo Silva, Ph.D.

Orientador: Prof. Danilo Silva, Ph.D.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Cin, Vinicius
Classificação de imagens de moda utilizando modelos de aprendizado profundo com pré-treinamento contrastivo multimodal / Vinicius Cin ; orientador, Danilo Silva, 2023.
67 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2023.

Inclui referências.

1. Engenharia Eletrônica. 2. Visão computacional. 3. Classificação. 4. CLIP. I. Silva, Danilo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Vinicius Cin

Classificação de imagens de moda utilizando modelos de aprendizado profundo com pré-treinamento contrastivo multimodal

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro Eletrônico" e aprovado em sua forma final pelo Curso de Graduação em Engenharia Eletrônica.

Florianópolis, 18 de dezembro de 2023

Documento assinado digitalmente
Fernando Rangel de Sousa
Data: 18/12/2023 13:16:56-0300
CPF: ***.649.114-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Fernando Rangel de Sousa, Dr.
Coordenador do Programa de Graduação
em Engenharia Eletrônica

Banca Examinadora:

Documento assinado digitalmente
Danilo Silva
Data: 18/12/2023 11:29:24-0300
CPF: ***.557.924-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Danilo Silva, Ph.D.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
FILIPE ROLIM CORDEIRO
Data: 18/12/2023 12:08:39-0300
CPF: ***.976.194-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Filipe Rolim Cordeiro, Dr.
Avaliador
Universidade Federal Rural de Pernambuco

Eng. Daniel de Souza Severo
Avaliador
Vector Institute

Dedico este trabalho aos meus pais, a minha irmã e avós.

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que contribuíram ao longo desta jornada acadêmica. Em primeiro lugar, agradeço aos meus pais, Cleusa e Adilson, por terem sido os pilares e maiores incentivadores em todas as etapas desta trajetória. À minha irmã Juliana, com quem pude contar em todos os momentos. Ao meu avô Antônio, por todas as palavras de apoio e energia positiva. Aos meus amigos de graduação, que tornaram esta jornada muito mais suave, e me proporcionaram incríveis momentos de diversão.

Agradeço aos meus colegas de laboratório, por todas as interações e momentos de descontração. A todos os meus professores, em especial ao meu orientador Danilo Silva, por ter me oferecido a oportunidade de participar do GAMA, ter me orientado com dedicação e paciência, e ter contribuído para minha evolução profissional e pessoal, compartilhando ensinamentos que levarei para toda a vida. Por fim agradeço a todos aqueles que fizeram parte desta jornada, por qualquer que tenha sido sua contribuição.

"What truly matters is to be surrounded by those who see the beauty in our every shade, who hear the silent songs of our souls. Just by being there, they have the power to make life's darkest moments shine with hope."

(unknown)

RESUMO

Este trabalho apresenta uma comparação entre diferentes arquiteturas de redes neurais profundas para a tarefa de classificação de imagens de moda. A utilização de modelos pré-treinados no ImageNet é pratica comum ao treinar-se um modelo para classificação de imagens. Em um trabalho recente, (RADFORD et al., 2021) demonstra em seu estudo que modelos de classificação pré-treinados com o método CLIP (*Contrastive Language-Image Pre-training*), performam significativamente melhor do que outros modelos, atingindo o estado da arte em 21 de 27 diferentes conjuntos de dados. O objetivo deste trabalho é avaliar a acurácia de modelos pré-treinados com o método CLIP, que utiliza aprendizado contrastivo a partir de imagens e textos, e compará-los com modelos pré-treinados no conjunto de dados ImageNet. Para a comparação, utiliza-se um subconjunto pré-processado das imagens do conjunto de dados *Deep Fashion*, contendo aproximadamente 80 mil imagens de 16 classes de roupas. O modelo pré-treinado com o método CLIP obteve a maior acurácia, com uma diferença de 3.00 pontos percentuais em relação ao melhor modelo pré-treinado no ImageNet, demonstrando a vantagem do pré-treinamento contrastivo multimodal para a classificação de imagens de moda.

Palavras-chaves: Visão computacional ; Classificação ; CLIP.

ABSTRACT

This work presents a comparison between different deep neural network architectures for the fashion image classification task. The uso of pre-trained models on ImageNet is a common practice when training a model for image classification. In a recent work, (RADFORD et al., 2021) demonstrates in his study that classification models pre-trained with the CLIP method (Contrastive Language-Image Pre-training), perform significantly better than other models, reaching the state of art in 21 of 27 different datasets. The objective of this work is to evaluate the accuracy of models pre-trained with the CLIP method, which uses contrastive learning from images and texts, and compare them with models pre-trained on the ImageNet dataset. For comparison, a pre-processed subset of images from the Deep Fashion dataset is used, containing approximately 80 thousand images from 16 clothing classes. The model pre-trained with the CLIP method achieved the highest accuracy, with a difference of 3.00 percentage points in relation to the best pre-trained model on ImageNet, demonstrating the advantage of multimodal contrastive pre-training for classifying fashion images.

Keywords: Computer vision ; Classification; CLIP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFSC	Universidade Federal de Santa Catarina
GAMA	Grupo de Pesquisa em Aprendizado de Máquina e Aplicações
DF1	Deep Fashion, conjunto de dados
DF2	Deep Fashion 2, conjunto de dados
BB	Caixa delimitadora (*bounding box*)
AR	Proporção (*aspect ratio*)
CLIP	*Contrastive image language Pre training*
ML	*Machine learning*
IA	Inteligência Artificial
ImageNet	Banco de dados de imagens hierárquico
WordNet	Banco de dados de palavras em inglês
COCO	Banco de dados de imagens (Common Objects in Context)
ILSVRC	ImageNet *Large Scale Visual Recognition Challenge*
AP	*Average precision*
mAP	*Mean average precision*
IoU	*Intersection over union*
CNNs	Redes neurais convolucional
ReLU	Unidade linear retificada
ResNet	Rede Residual
ConvNets	Redes neurais convolucionais
ViT	Vision Transformer
PLN	Processamento de linguagem natural
YOLO	Modelo de detecção de objetos (*you only look once*)

SUMÁRIO

1 INTRODUÇÃO

A classificação de imagens é uma tarefa fundamental no campo de aprendizado de máquina, que desempenha um papel crucial na identificação e categorização de objetos visuais em imagens. Essa tarefa tem inúmeras aplicações práticas, abrangendo desde o reconhecimento facial, detecção de doenças até a segurança e o entretenimento. A classificação de imagens envolve treinar modelos computacionais para extrair características relevantes das imagens e associá-las a rótulos predefinidos. No entanto, classificar itens de moda (roupas e acessórios) é uma tarefa complexa, pois, ao contrário de objetos genéricos, itens de moda sofrem de variações significativas de estilo e design e, ainda mais importante, possuem características de alto nível que muitas vezes são necessárias para diferenciar e classificar certas peças de roupas, o que torna este tópico desafiador (CHENG et al., 2021).

Uma abordagem promissora para a tarefa de classificação de imagens é o *deep learning*, esta técnica consiste na criação de um modelo de aprendizado de máquina através da utilização de varias camadas de processamento. As camadas do modelo extraem estruturas e padrões complexos de um conjuntos de dados, e utiliza-se o algoritmo de *backpropagation* para alterar os parâmetros internos do modelo, que são utilizados para o cálculo da representação em cada camada a partir da representação da camada anterior (LECUN; BENGIO; HINTON, 2015). Com os avanços contínuos na área de *deep learning*, diversas arquiteturas de modelos foram desenvolvidas, cada uma com suas características distintas. Por exemplo, a ResNet (HE et al., 2016), a EfficientNet (TAN; LE, 2019), os Visual Transformers (DOSOVITSKIY et al., 2020) e muitas outras representam um leque diversificado de opções. Essas arquiteturas se tornaram componentes essenciais na busca por soluções eficazes para a classificação de imagens.

Para avaliar o desempenho dessas arquiteturas, os pesquisadores frequentemente recorrem a um conjunto de dados comum, juntamente com métricas padronizadas, a fim de conduzir experimentos comparativos. Nesse sentido, é um hábito comum da comunidade utilizar o conjunto ImageNet (DENG et al., 2009), com isso é possível encontrar diversos modelos pré-treinados em suas classes. Este conjunto possui milhões de imagens rotuladas em milhares de categorias e serve como uma base sólida para avaliar e comparar a eficácia de várias arquiteturas de modelos.

A abordagem conhecida como transferência de aprendizado (*transfer learning*), permite aproveitar o conhecimento adquirido por um modelo pré-treinado em um grande conjunto de dados, como o ImageNet, e ajustar o modelo para se adequarem a classes específicas. Essa prática economiza tempo e recursos computacionais, tornando-a uma estratégia valiosa em projetos de classificação em diferentes domínios, nos quais as classes

não são as mesmas que as do ImageNet, entretanto, o ImageNet possui suas limitações, contendo apenas 1000 classes pré-definidas, o que afeta a sua capacidade de representação universal (AN et al., 2023).

Recentemente, a OpenAI [1] disponibilizou modelos pré-treinados com o método CLIP (Contrastive Language-Image Pre-training), que se baseia em aprendizado contrastivo. Segundo os autores do método CLIP (RADFORD et al., 2021), foram utilizados 400 milhões de pares de imagem e texto para a criação do conjunto de dados utilizado no treinamento dos modelos. A função objetivo do método é maximizar a similaridade entre pares de imagem e texto, aprendendo conceitos visuais por meio da supervisão de linguagem natural.

Este trabalho visa investigar a hipótese de que modelos CLIP podem ser uma alternativa eficaz e eficiente aos modelos pré-treinados no ImageNet, de forma a ser um possível caminho para a obtenção de melhores resultados na classificação de imagens de moda. (RADFORD et al., 2021) demonstra em seu estudo que modelos de classificação pré-treinados com CLIP, performam significativamente melhor do que outros modelos, atingindo o estado da arte em 21 de 27 diferentes conjuntos de dados, assim, demonstrando uma vantagem do aprendizado contrastivo multimodal em comparação ao pré-treinamento tradicional baseado em classificação no ImageNet. Segundo (AN et al., 2023), o método CLIP demostrou um grande sucesso em diversas tarefas, como recuperação de imagem (*retrieval*) e classificação, sendo este sucesso devido a capacidade superior de representação obtida pelo método contrastivo multimodal.

1.1 OBJETIVOS

O objetivo geral deste trabalho é comparar o desempenho de modelos pré-treinados com o método CLIP e modelos pré-treinados no ImageNet. O foco da comparação é a classificação de imagens de moda (peças de roupas), utilizando um conjunto de dados específico para essa tarefa.

1.1.1 Objetivos específicos

- Preparar um conjunto de dados de imagens de moda (peças de roupas) adequado para o treinamento dos modelos a partir de dados disponíveis publicamente.
- Treinar e avaliar modelos pré-treinados no ImageNet.
- Avaliar o desempenho do modelo CLIP (sem treinamento) no conjunto de dados desenvolvido.
- Treinar e avaliar o modelo CLIP.

[1] openai.com

- Comparar a acurácia de modelos pré-treinados com o método CLIP em relação aos modelos treinados no ImageNet.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 APRENDIZADO DE MÁQUINA

A área de Aprendizado de Máquina ou *Machine Learning* (ML) é um campo da Inteligência Artificial (IA) que se dedica a ensinar computadores a aprender com dados e experiências, em vez de depender de programação explícita. Os algoritmos de ML dependem da análise de dados para a extração de características e a tomada de decisões, com o objetivo de desenvolver sistemas capazes de realizar tarefas complexas de forma autônoma ou com orientação humana. Essas tarefas englobam desde o reconhecimento de imagens e tradução de texto até diagnósticos médicos e previsões estatísticas, abrangendo um amplo espectro de aplicações (SANTANA, 2018). No decorrer dos anos 2000, surgiu o conceito de Aprendizado Profundo, frequentemente denominado *Deep Learning*, sendo uma ramificação especializada do Aprendizado de Máquina, que emprega redes neurais, inspiradas na funcionalidade do cérebro humano, para abordar problemas de elevada complexidade (CHAGAS, 2019). Esse campo ganhou notoriedade após um artigo publicado por Geoffrey Hinton e Ruslan Salakhutdinov (CHAGAS, 2019), demonstrando como treinar redes neurais com várias camadas de maneira sequencial. A Figura 1, demostra a hierarquia dos campos da IA citados acima.

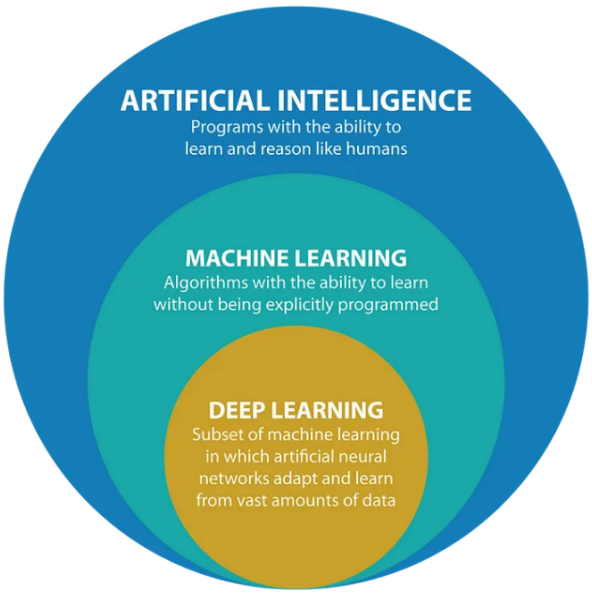
Existem diferentes maneiras com que uma rede pode aprender com os dados, a escolha da dessa abordagem depende fortemente da estrutura dos dados e aplicação desejada para o modelo. Os principais tipos são:

- Aprendizado supervisionado: o sistema recebe um conjunto de dados rotulados, ou seja, que contêm a resposta desejada para cada exemplo. O sistema aprende a partir desses dados e tenta generalizar para novos exemplos não vistos antes. Exemplos de técnicas de aprendizado supervisionado são regressão, classificação, árvores de decisão e redes neurais artificiais.

- Aprendizado não supervisionado: o sistema recebe um conjunto de dados não rotulados, ou seja, que não contêm a resposta desejada. O sistema tenta encontrar padrões, estruturas ou agrupamentos nos dados, sem ter uma orientação prévia. Exemplos de técnicas de aprendizado não supervisionado são análise de componentes principais, clusterização e detecção de anomalias.

- Aprendizado por reforço: o sistema interage com um ambiente dinâmico e recebe recompensas ou punições por suas ações. O sistema aprende a partir desses *feedbacks* e tenta maximizar sua recompensa total ao longo do tempo. Exemplos de técnicas de aprendizado por reforço são algoritmos genéticos, *Q-learning* e *policy gradient*.

Figura 1 – Hierarquia das áreas de estudo: Inteligencia Artificial, Aprendizado de Máquina e Aprendizado Profundo

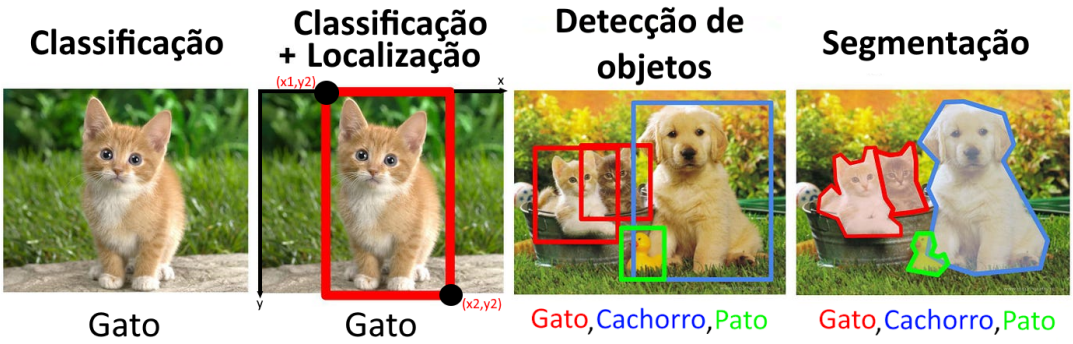


Fonte: (SANTANA, 2018).

2.2 VISÃO COMPUTACIONAL

A visão computacional é um campo que permite aos computadores "ver" e compreender dados visuais, obtendo informações relevantes a partir de imagens, podendo assim, tomar decisões ou auxiliar neste processo (JAIN; WAH, 2022); Trata-se de um campo interdisciplinar que combina técnicas de processamento de imagens, aprendizado de máquina, inteligência artificial e outras áreas. O objetivo da visão computacional é criar sistemas capazes de realizar tarefas que envolvem a percepção e o entendimento de imagens ou vídeos. A classificação de imagens é uma tarefa que envolve a atribuição de rótulos ou categorias a imagens com base em seu conteúdo visual. O objetivo é treinar um modelo capaz de reconhecer e distinguir objetos, padrões ou características em imagens. Outras tarefas típicas da visão computacional são a detecção de objetos e a segmentação. Na detecção de objetos, o modelo realiza a predição da classe e da localização do objeto na imagem, geralmente descrita a partir de duas coordenadas (canto superior "x1,y1" e canto inferior "x2,y2" da imagem). A segmentação de imagem consiste em classificar cada pixel da imagem referente ao objeto, obtendo assim o seu contorno. A Figura 2 exemplifica as tarefas descritas acima.

Figura 2 – Tarefas Típicas de Visão Computacional



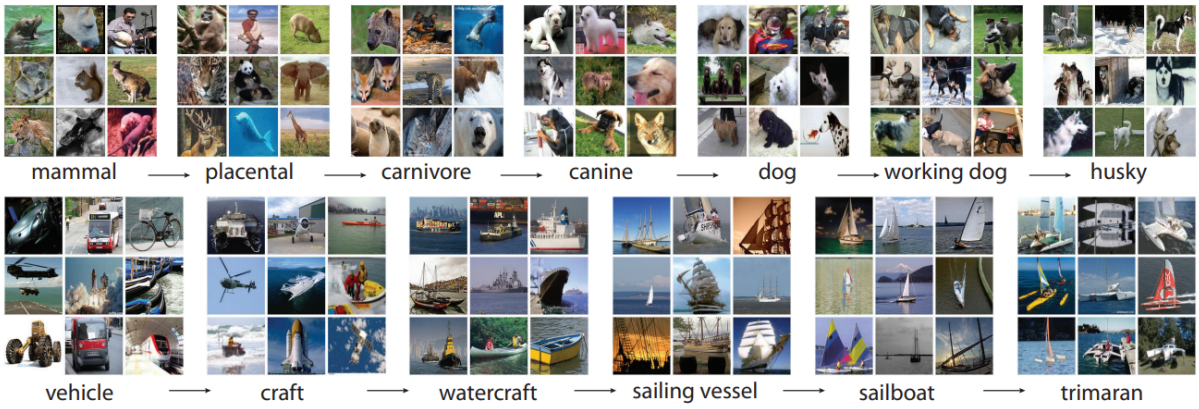
Adaptado de: (JAISWAL et al., 2021)

2.2.1 ImageNet

O ImageNet é um banco de dados de imagens que foi criado com o objetivo de fornecer uma cobertura abrangente e diversificada do mundo visual, utiliza a estrutura hierárquica do WordNet, uma ontologia de conceitos, para organizar imagens coletadas da internet em mais de 20 mil categorias, cada uma com centenas ou milhares de imagens de alta qualidade e anotadas por humanos, se tornando um recurso valioso para o desenvolvimento, treinamento e avaliação de algoritmos de detecção e classificação de imagens (DENG et al., 2009). O ImageNet Large Scale Visual Recognition Challenge (ILSVRC) é uma competição anual, lançada em 2010 que utiliza um subconjunto do ImageNet com cerca de 1000 categorias e mais de um milhão de imagens. O desafio tem sido realizado

anualmente até o presente momento, atraindo a participação de mais de dezenas de instituições (RUSSAKOVSKY et al., 2015) e tem como objetivo proporcionar uma base de comparação de resultados entre modelos participantes, estimulando o desenvolvimento de novas arquiteturas, técnicas e abordagens para a classificação de imagens. Na Figura 3 pode-se visualizar alguns exemplos de imagens do ImageNet.

Figura 3 – Exemplo de imagens - ImageNet

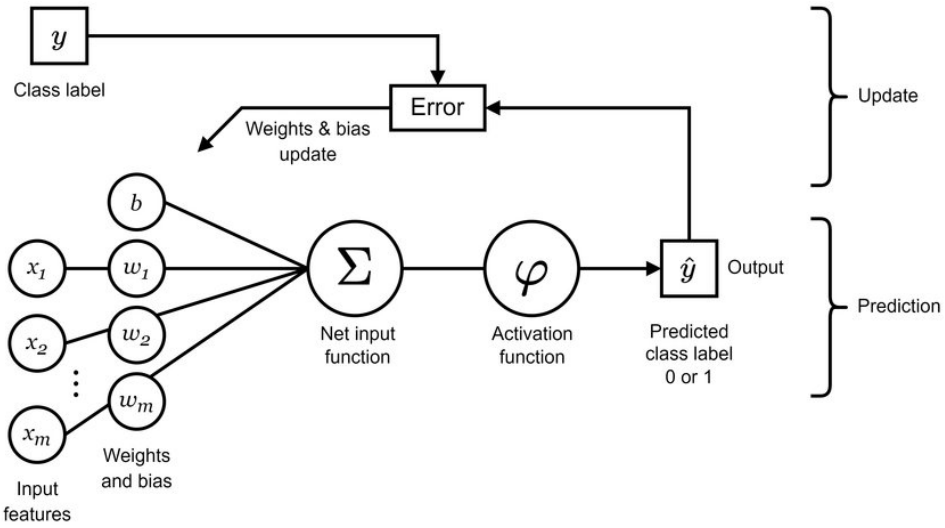


Fonte: (DENG et al., 2009)

2.2.2 Perceptron

O perceptron é um modelo simples de rede neural que recebe um vetor de entrada x e produz uma saída $\hat{y}$. A saída é obtida pela equação $\hat{y} = \varphi(x \cdot w + b)$, onde o vetor w representa os pesos (*weights*), b o viés (*bias*) e φ uma função com saída 0 ou 1.

Figura 4 – Perceptron



Fonte: (GUSTINELI, 2022)

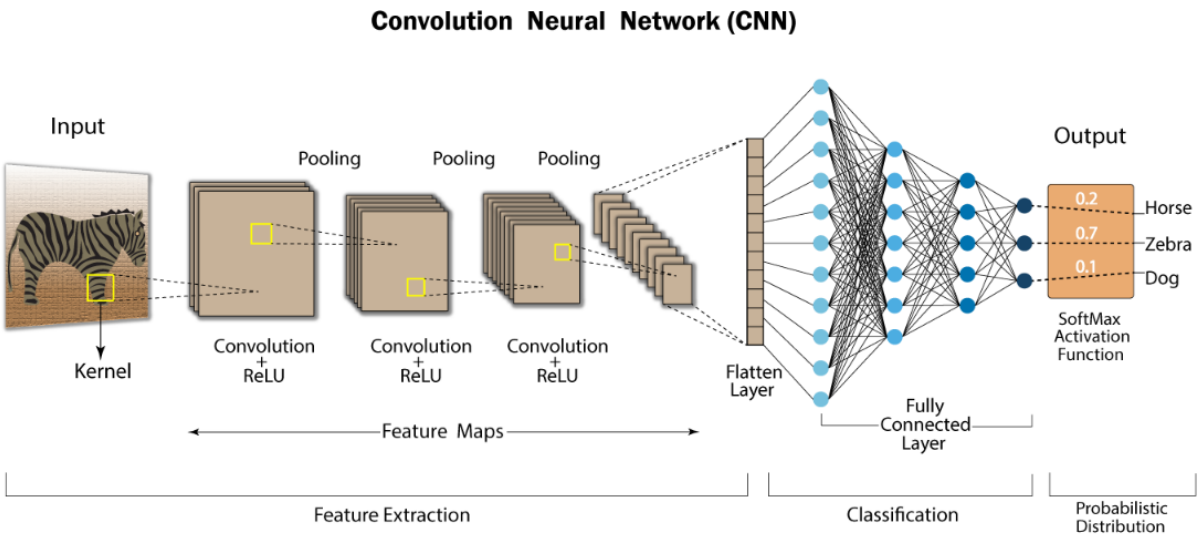
O Perceptron pode ser utilizado para realizar tarefas de classificação linear, ou seja, separar dados em duas classes distintas. Os pesos e o viés das conexões entre a entrada

e a saída são aprendidos por meio de um algoritmo de aprendizado supervisionado, que realiza ajustes nos valores de w e b de acordo com o erro entre a saída desejada (y) e a saída obtida ($\hat{y}$). A Figura 4 ilustra uma representação do perceptron. A representação de um perceptron pode variar dependendo da estrutura da rede neural (GUSTINELI, 2022).

2.2.3 Redes Convolutivas

As redes neurais convolutivas (CNNs) são um tipo de rede neural profunda que se destaca no campo da visão computacional. As CNNs podem realizar tarefas que antes eram consideradas impossíveis, como identificar rostos, dirigir carros autônomos, operar supermercados sem caixas e oferecer soluções para tratamentos médicos inteligentes (LI et al., 2021). Desde o surgimento das CNNs, a convolução é uma operação fundamental no processamento de imagens e é utilizada para extrair características dos dados de entrada de uma imagem, denominadas *features*.

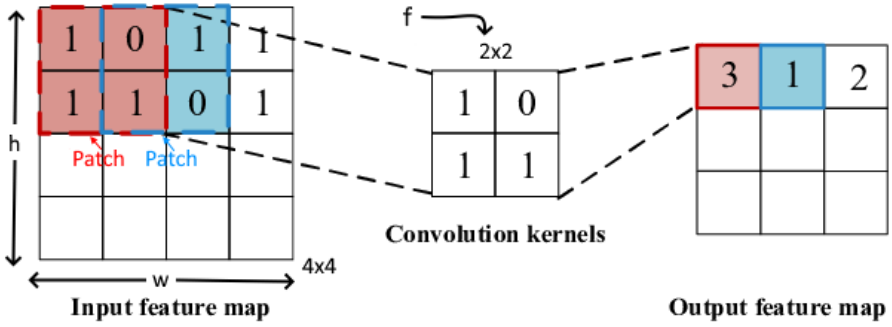
Figura 5 – Rede convolutiva



Fonte: (SWAPNA K, 2020)

Uma camada convolutiva (também chamada de camada convolucional) é composta por vários mapas de vetores de atributos (*features maps*), que são representadas por matrizes. Cada um dos quais aprende a detectar um recurso/característica específico(a) nos dados de entrada, usando um conjunto de pesos denominado *filter bank*. A camada convolutiva executa uma operação de convolução discreta nos dados de entrada, que é uma soma ponderada de trechos (*patches* locais) da camada anterior (Figura 6). O resultado é então passado por uma função não linear, como uma unidade linear retificada (ReLU). A camada convolutiva pode aprender a extrair vetores de atributos locais e invariantes dos dados de entrada, que as torna úteis para tarefas envolvendo o processamento de imagens (LECUN; BENGIO; HINTON, 2015). A Figura 5 ilustra uma rede convolutiva para classificação de imagens.

Figura 6 – Convolução 2D

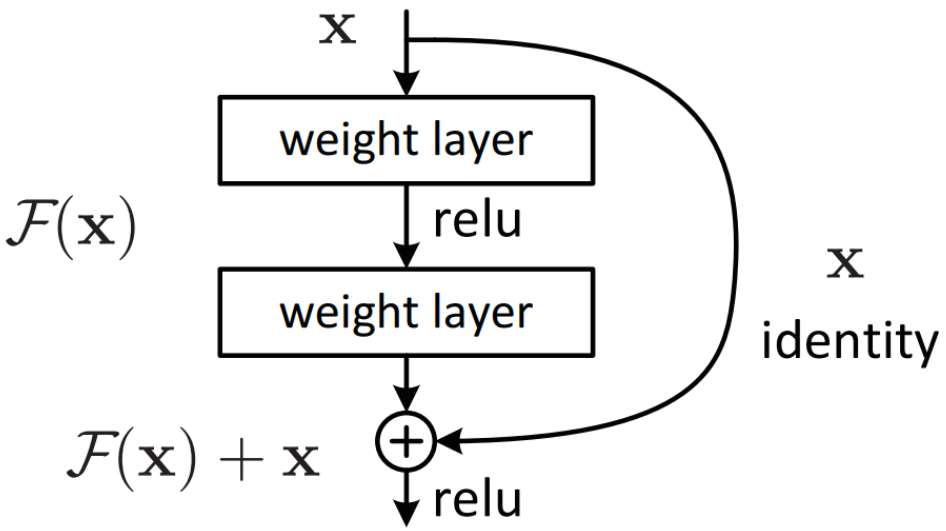


Adaptado de: (LIANG et al., 2020).

2.2.4 Redes Residuais

Intuitivamente, podemos pensar que, ao aumentar o tamanho de uma rede, i.e, adicionar mais camadas, alcançaremos melhores resultados até um ponto de estagnação, porém, não é o que ocorre. Ao passar de um determinado tamanho, o valor da perda começa a aumentar conforme o aumento da rede. Isso ocorre pois, fica cada vez mais difícil de propagar a informação das primeiras camadas para as últimas, ocasionando perdas (dissipação do gradiente). Juntamente a perda de informação, pode ocorrer de que o resultado de uma camada anterior já seja ótimo, fazendo com que a camada seguinte seja desnecessária e tenha que aprender a função identidade.

Figura 7 – Camada Residual (sub-rede)



Fonte: (HE et al., 2016)

A ResNet (Rede Residual) é uma arquitetura de rede neural profunda proposta por (HE et al., 2016), que ganhou o primeiro lugar nas competições ILSVRC e COCO naquele ano. A ResNet se baseia na ideia de aprendizado residual, que consiste em adicionar conexões diretas entre as camadas de entrada e saída de uma sub-rede (Figura 7), chamadas

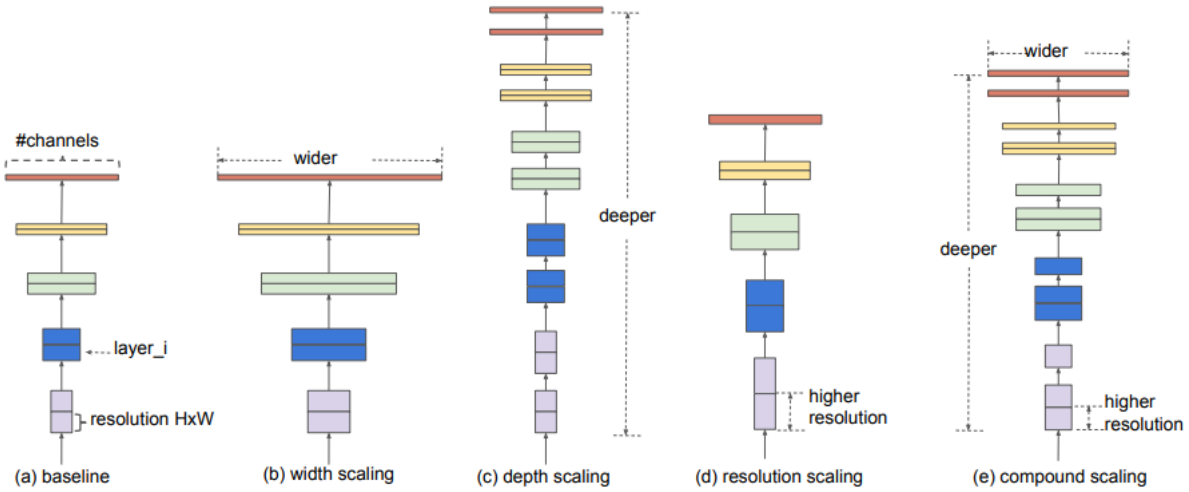
de atalhos (*shortcuts*). Essas conexões permitem que a sub-rede aprenda uma função residual em relação à entrada, em vez de aprender a função desejada diretamente (HE et al., 2016). A hipótese dos autores é que essa reformulação facilita a otimização e o aprendizado de redes profundas, evitando o problema de degradação do desempenho com o aumento da profundidade.

2.2.5 EfficientNet

A EfficientNet é uma família de redes neurais convolucionais (ConvNets) que foram desenvolvidas por (TAN; LE, 2019) com o objetivo de obter alta acurácia e eficiência no reconhecimento de imagens. A EfficientNet se baseia em dois conceitos principais: a busca neural de arquitetura e o escalonamento composto de modelo.

A busca neural de arquitetura é um método que utiliza algoritmos de otimização para encontrar a melhor arquitetura de rede para uma determinada tarefa, explorando um grande espaço de possibilidades. (TAN; LE, 2019) utilizaram a busca neural para projetar uma rede base, chamada de EfficientNet-B0, que possui uma combinação ótima de tipos e tamanhos de camadas convolucionais.

Figura 8 – Método proposto por Tan e Le (2019)



Fonte: (TAN; LE, 2019)

O escalonamento composto é uma técnica proposta por (TAN; LE, 2019) que consiste em escalar uniformemente todas as dimensões de uma rede (Figura 8), ou seja, a profundidade (número de camadas), a largura (número de canais) e a resolução (tamanho da imagem de entrada). Os autores mostraram que esse método é mais eficaz do que escalar apenas uma dimensão, pois permite balancear os recursos computacionais e capturar padrões mais complexos nas imagens (TAN; LE, 2019).

A partir da rede base (EfficientNet-B0), os autores utilizaram o escalonamento composto para obter uma família de redes chamadas de EfficientNet-B1 a B7. Essas redes

alcançaram resultados superiores aos das redes convencionais, tanto em termos de acurácia quanto de eficiência, em diversos conjuntos de dados de reconhecimento de imagens, como ImageNet, CIFAR-100, Stanford Cars, entre outros (TAN; LE, 2019).

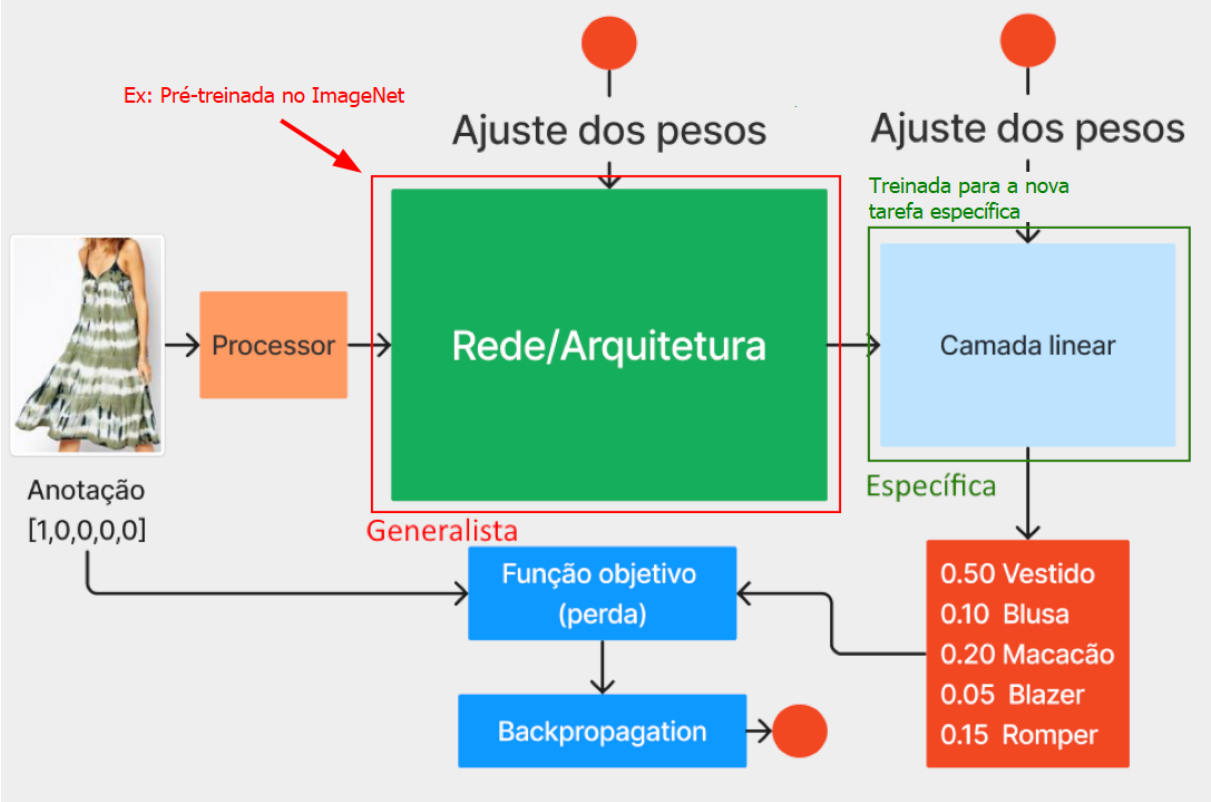
2.2.6 Transferência de aprendizado

Tradicionalmente, no aprendizado de máquina, os dados utilizados para treino e testes possuem a mesma distribuição de dados. Quando há uma diferença nestas distribuições, o desempenho preditivo do modelo pode ser prejudicado. Em alguns cenários, obter dados suficientes para um conjunto de treinamento pode ser caro ou difícil. Portanto, há uma necessidade de criar modelos com alto desempenho para uma tarefa mais generalizada, onde há uma quantidade elevada de dados disponíveis, de forma com que possa-se utilizar este aprendizado em uma tarefa relacionada mais específica, esta é a motivação para a transferência de aprendizado (WEISS; KHOSHGOFTAAR TAGHI; WANG, 2016).

A transferência de aprendizado (*transfer learning*) consiste no uso de uma rede base que foi previamente treinada em um conjunto de dados e tarefa base, para aproveitamento dos vetores de atributos extraídos (*embeddings*) que ela aprendeu, os transferindo para uma segunda rede que será treinada em um novo conjunto de dados e tarefa específica. Essa técnica tende a ser eficaz se os vetores de atributos extraídos do modelo base forem genéricos, ou seja, adequados tanto para a tarefa base quanto para a alvo, em vez de específicos para a tarefa base (YOSINSKI et al., 2014). A ideia por trás desta técnica se espelha no processo natural de aprendizagem do ser humano. Nós reconhecemos e aplicamos conhecimentos relevantes de aprendizagens anteriores quando encontramos novas tarefas (TORREY; SHAVLIK, 2010). Em vez de treinar um modelo do zero, economiza-se tempo e recursos ajustando um modelo pré-treinado. A Figura 9 ilustra um caso de transferência de aprendizado para classificação de imagens. Uma rede pré-treinada no ImageNet é utilizada para extrair informação da imagem e uma segunda rede (camada linear) aprende a realizar a classificação.

Existem diferentes formas de se utilizar um modelo pré-treinado, um caso particular é o ajuste fino (*fine tuning*), onde utiliza-se o modelo pré-treinado como ponto inicial, mas treina-se todas as camadas (generalista e específica). Outra abordagem, conhecida como *linear probing* consiste de congelar todas as camadas pré-treinadas (generalistas) e atualizar somente a camada final de classificação (específica), geralmente empregada quando o conjunto de dados para a tarefa específica é muito menor do que o conjunto utilizado para treinamento do modelo base.

Figura 9 – Transferência de aprendizado em um rede de classificação

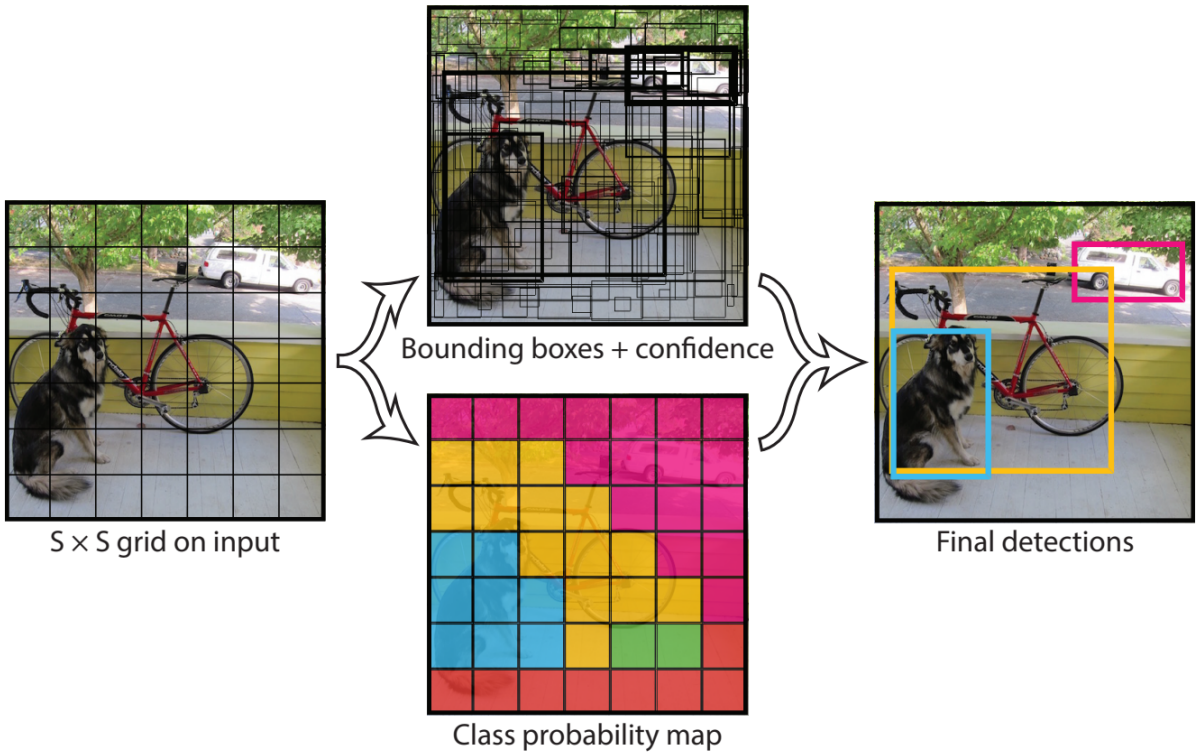


Fonte: Autor

2.2.7 You Only Look Once

A Figura 10 apresenta uma visão geral do modelo YOLO (REDMON et al., 2016), onde a imagem de entrada é subdividida em uma grade $S \times S$. Para cada célula da grade, o método prevê B caixas delimitadoras, a confiança associada a cada caixa e as probabilidades de pertencer a C classes distintas. Este processo de predição é codificado em um vetor de atributos (REDMON et al., 2016), que é posteriormente interpretado de forma a se obter as anotações de *bounding boxe* (BB), que são caixas que localizam o objeto dentro da imagem. O termo YOLO, "você só olha uma vez", reflete a capacidade da rede de realizar suas predições examinando a imagem apenas uma vez, eliminando a necessidade de abordagens anteriores com duas ou mais etapas, que dependiam de métodos para extração de propostas de regiões para classificação.

Figura 10 – Modelo YOLO



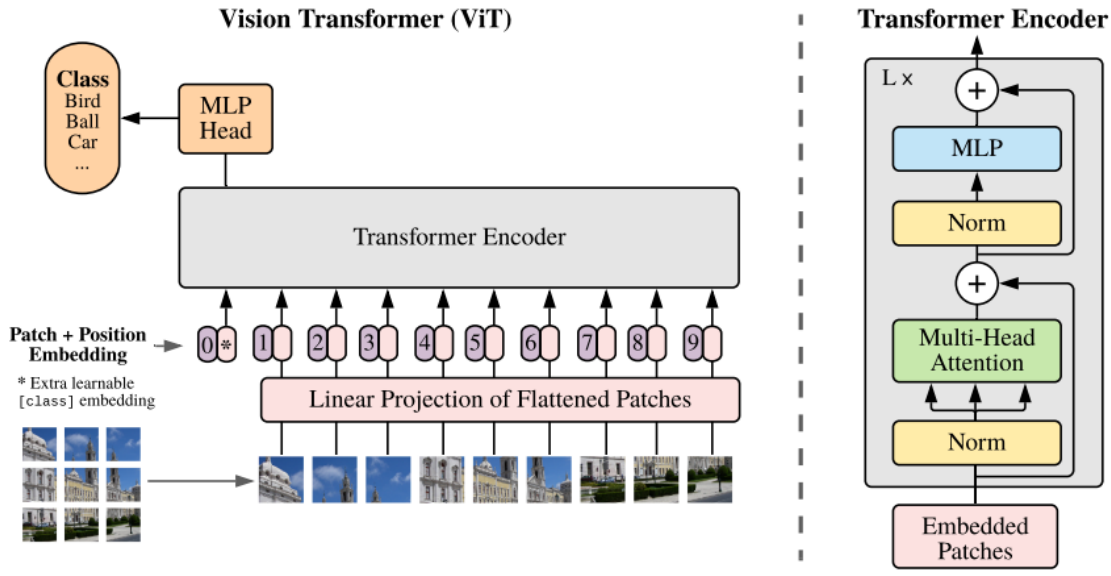
Fonte: (REDMON et al., 2016)

2.2.8 Vision Transformers

Vision Transformers (ViT) são modelos aplicados a tarefas visuais baseados na arquitetura *Transformers*, que foi proposta por (VASWANI et al., 2017) para processamento de linguagem natural (PLN). (DOSOVITSKIY et al., 2020) aplicou diretamente uma arquitetura *Transformer* em *patches* (recortes) bidimensionais de imagem de mesmo tamanho e não sobrepostos para classificação de imagens. Um *patch* de imagem é uma região quadrada que contém uma região da imagem original. A ideia é que cada *patch* possa representar alguma informação visual relevante, como uma parte de um objeto ou uma textura.

A arquitetura ViT funciona da seguinte forma: uma imagem é dividida em *patches* de tamanho fixo, cada um deles é transformado em um vetor de números em ponto-flutuante (vetor de atributos). Em seguida, são adicionados vetores de posição, que indicam a localização de cada *patch* na imagem. A sequência resultante de vetores é alimentada em um encoder *Transformer* padrão, que é uma estrutura composta por várias camadas. Uma destas camadas, a atenção, permite que o modelo aprenda as relações entre os *patches* de imagem, capturando assim o contexto e a semântica da imagem. O *encoder Transformer* produz uma nova sequência de vetores, que pode ser usada para aplicações de classificação, segmentação ou detecção de objetos. A Figura 11 ilustra a arquitetura de um ViT.

Figura 11 – Visão geral do Vision Transformer (ViT)



Fonte: (DOSOVITSKIY et al., 2020)

Os autores avaliaram os modelos ViT em diversos conjuntos de dados de classificação de imagens, como ImageNet, CIFAR-100, VTAB, entre outros. Os resultados de (VASWANI et al., 2017) mostraram que o ViT (no momento da publicação) atinge ou supera o estado da arte (BiT-L), desta forma, demonstrando uma boa capacidade de transferência de aprendizado, podendo se adaptar a tarefas com menos dados a partir de modelos pré-treinados em grandes conjuntos de dados.

2.3 MÉTRICAS AVALIATIVAS

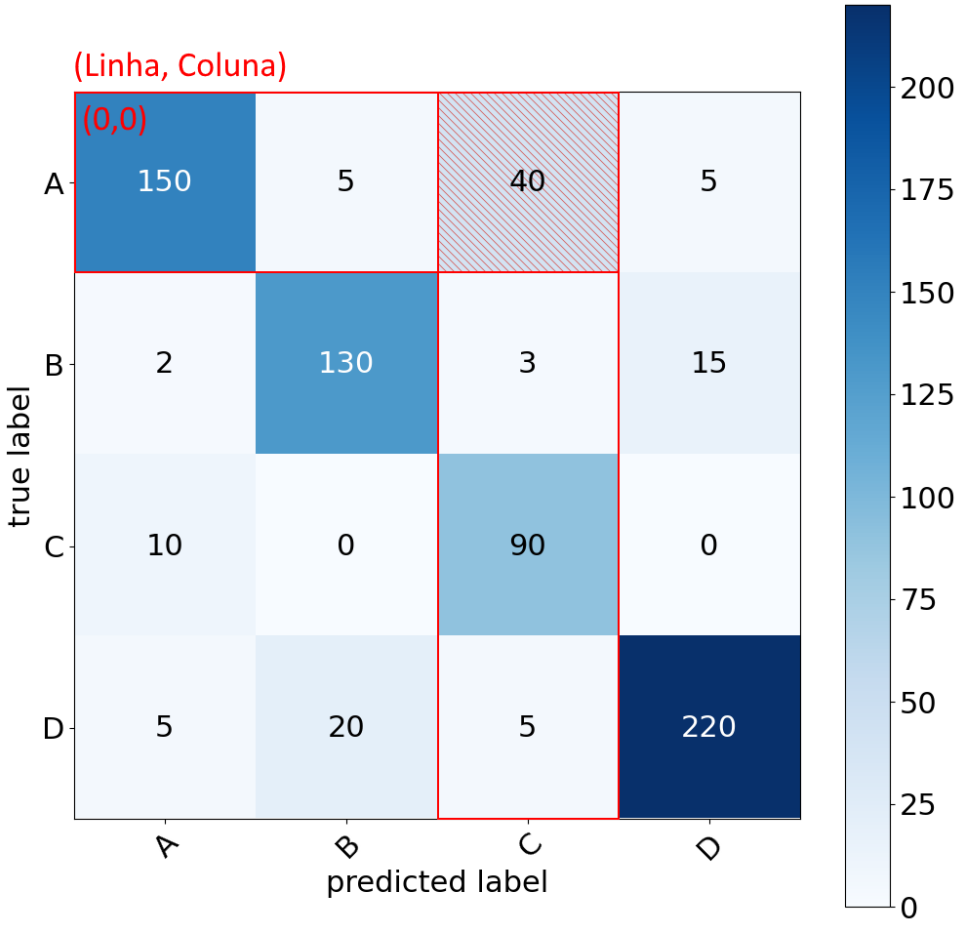
Um modelo de aprendizado de máquina é otimizado através de uma função matemática chamada de função objetivo ou perda. Para uma amostra, este valor representa o erro ou diferença entre o rótulo previsto e o rótulo verdadeiro. No entanto, a perda apresentada por um modelo é relativa e não nos fornece informações suficientes para a avaliação e comparação entre diferentes arquiteturas. Portanto, é necessário definir métricas apropriadas para a tarefa e conjunto de dados em questão. Uma métrica de avaliação pode ser descrita como a ferramenta que mede o desempenho do classificador, diferentes métricas avaliam diferentes características do modelo (HOSSIN; SULAIMAN, 2015). Para a tarefa de classificação, algumas das métricas mais comuns incluem Acurácia, precisão e a sensibilidade (*recall*). Essas métricas podem ser calculadas a partir da matriz de confusão.

2.3.1 Matriz de confusão para classificação multi-classe

A matriz de confusão é uma ferramenta que permite avaliar o desempenho de um modelo de classificação, comparando as classes reais e as classes preditas pelo modelo. Em

um problema de classificação multi-classe, a matriz de confusão é uma tabela quadrada de dimensão NxN, onde N é o número de classes distintas. Cada linha da tabela corresponde a uma classe real, e cada coluna corresponde a uma classe predita. O elemento i, j da matriz indica o número de exemplos que pertencem a classe i e foram classificados como pertencentes a classe j pelo modelo. Portanto, a diagonal principal da matriz contém os exemplos que foram classificados corretamente, e os elementos fora da diagonal representam os erros de classificação.

Figura 12 – Exemplo de matriz de confusão multi-classe



Fonte: Autor

A matriz de confusão permite visualizar a distribuição das predições do modelo, bem como identificar as classes que são mais confundidas entre si. Por exemplo, na Figura 12, observa-se que o modelo classificou corretamente, 150 exemplos da classe A, 130 da classe B, 90 da classe C e 220 da classe D. No entanto, o modelo também cometeu alguns erros, como a classificação de 40 exemplos da classe A como pertencentes a classe C (elemento destacado da Figura 12). Perceba que, a soma de uma linha i da matriz, resulta no número total de amostras verdadeiras da classe i, (A:200, B:150, C:100, D:250), enquanto que a soma de uma coluna da classe j, resulta no número total de amostras classificadas pelo modelo como pertencentes a classe j (A:167, B:155, C:138, D:240).

Com base na matriz de confusão, pode-se calcular as seguintes métricas definidas abaixo:

- Acurácia: Representa a proporção de acertos dentre todas as predições realizadas pelo modelo. Por exemplo, o modelo representado pela matriz da Figura 12 possui uma acurácia de 84.28%

$$\text{Acurácia} = \frac{\text{Nº total de acertos}}{\text{Nº total de amostras}} \tag{2.1}$$

- Precisão (P_i): Para uma classe i, a precisão representa a proporção de amostras da classe i classificadas corretamente dentre todas as amostras classificadas como i pelo modelo. Um modelo preciso trás confiança sobre sua predição.

$$P_i = \frac{\text{Nº de acertos para a classe } i}{\text{Nº total de predições realizadas para a classe } i} \tag{2.2}$$

- Sensibilidade (R_i): Representa a proporção de amostras da classe i classificadas corretamente dentre todas as amostras da classe i. Um modelo com alta sensibilidade na classe i trás a confiança de que todas as amostras verdadeiras da classe i foram recuperadas.

$$R_i = \frac{\text{Nº de acertos para a classe } i}{\text{Nº total de amostras da classe } i} \tag{2.3}$$

Além das métricas mencionadas, para obter-se uma visão geral sobre a precisão e a sensibilidade para todas as classes, pode-se utilizar a média macro (média não ponderada) da precisão e da sensibilidade. As métricas fornecidas pela matriz de confusão proporcionam uma visão abrangente do desempenho de um modelo de classificação. Um exemplo de calculo das métricas mencionadas acima, considerando-se o modelo representado pela matriz de confusão da Figura 12 podem ser vistos na Tabela 1 e na Tabela 2. A métrica principal a ser utilizada para a avaliação final de um modelo deve ser escolhida com base nas características do problema a ser resolvido, uma vez que cada métrica nós traz informações sobre diferentes propriedades do modelo preditivo.

Tabela 1 – Exemplo do calculo de P_i e R_i

Classe	P_i	R_i
A	$150/167 = \mathbf{0.8982}$	$150/200 = \mathbf{0.7500}$
B	$130/155 = \mathbf{0.8387}$	$130/150 = \mathbf{0.8666}$
C	$90\ /138 = \mathbf{0.6521}$	$90\ /100 = \mathbf{0.9000}$
D	$220/240 = \mathbf{0.9166}$	$220/250 = \mathbf{0.8800}$

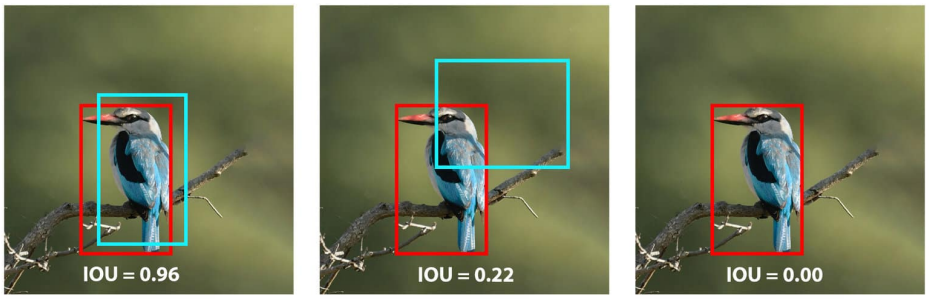
Tabela 2 – Exemplo do cálculo de Acurácia, Precisão e Sensibilidade

Métrica	**Cálculo**
Acurácia	$(150 + 130 + 90 + 220)/(200 + 150 + 100 + 250) =$ **0.8428**
Precisão macro	$(89.82 + 83.87 + 65.21 + 91.66)\ /\ 4 =$ **0.8263**
Sensibilidade macro	$(75.00 + 86.66 + 90.00 + 88.00)\ /\ 4 =$ **0.8491**

2.3.2 Métricas para Detecção de Objetos

A detecção de objetos é uma tarefa de visão computacional que consiste em localizar e classificar objetos em uma imagem. Para avaliar o desempenho de um modelo de detecção de objetos, é necessário definir algumas métricas que quantifiquem a qualidade das predições. Uma das métricas mais utilizadas é a *mean average precision* (mAP), que mede a *average precision* (AP) média do modelo para todas as classes e níveis de confiança (ZOU et al., 2023). Para entender como a mAP é calculada, é preciso entender alguns conceitos:

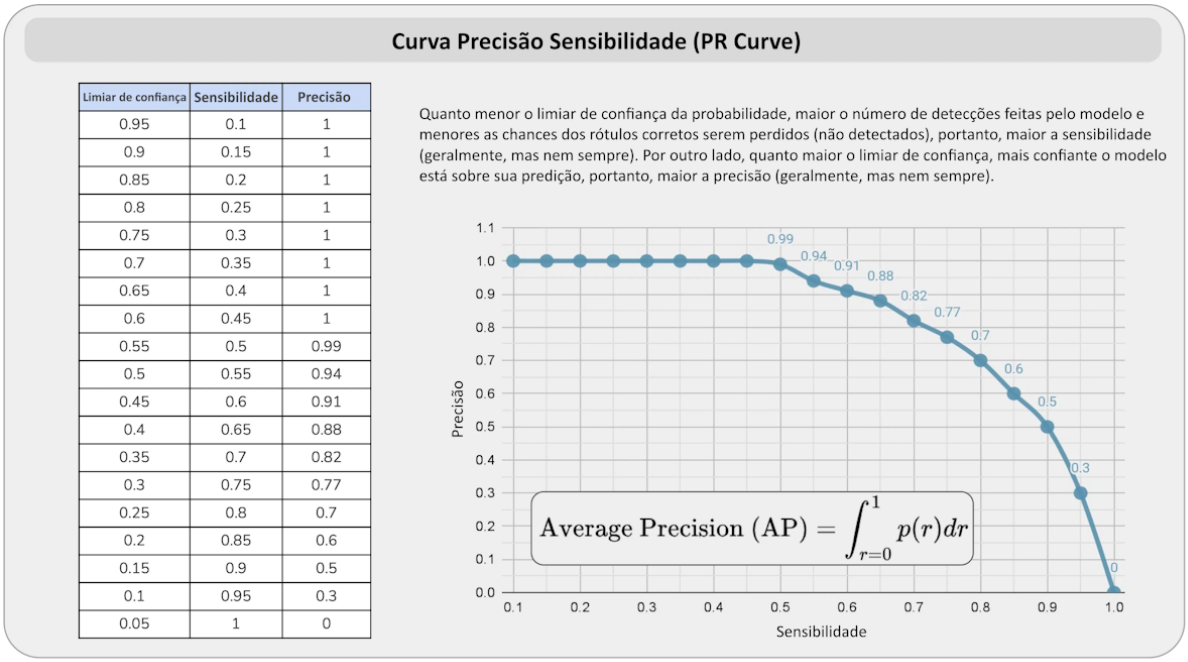
Figura 13 – Exemplo de predições para um *threshold* de IoU de 0.5



Fonte: (KUKIL, 2022)

- Intersection Over Union (IoU): Fundamental na detecção de objetos, a IoU quantifica o grau de sobreposição entre duas BB, geralmente entre a BB representando a região de interesse (*ground truth*) e a região prevista pelo modelo. Esta métrica, mede a razão da área de sobreposição entre as regiões e a área combinada das duas regiões, conforme ilustrado na Figura 15. A IoU varia de 0 a 1, onde 0 indica nenhuma sobreposição e 1 representa sobreposição perfeita. Em essência, o IoU avalia a precisão espacial da previsão em relação ao *ground truth* (KUKIL, 2022). É comum, utilizar-se a IoU com um limiar de 0.5 para determinar se um objeto foi ou não localizado, por exemplo, na Figura 15, considerando-se um limiar de 0.5, somente o primeiro caso é considerado como "detectado", enquanto os demais são considerados "perdas" ou erros de detecção. A Figura 13 ilustra alguns casos de IoU. Detecção correta (*True Positive*), Detecção incorreta (*False Positive*) e não detecção (*False Negative*).

Figura 14 – Exemplo de curva precisão x sensibilidade



Limiar de confiança	Sensibilidade	Precisão
0.95	0.1	1
0.9	0.15	1
0.85	0.2	1
0.8	0.25	1
0.75	0.3	1
0.7	0.35	1
0.65	0.4	1
0.6	0.45	1
0.55	0.5	0.99
0.5	0.55	0.94
0.45	0.6	0.91
0.4	0.65	0.88
0.35	0.7	0.82
0.3	0.75	0.77
0.25	0.8	0.7
0.2	0.85	0.6
0.15	0.9	0.5
0.1	0.95	0.3
0.05	1	0

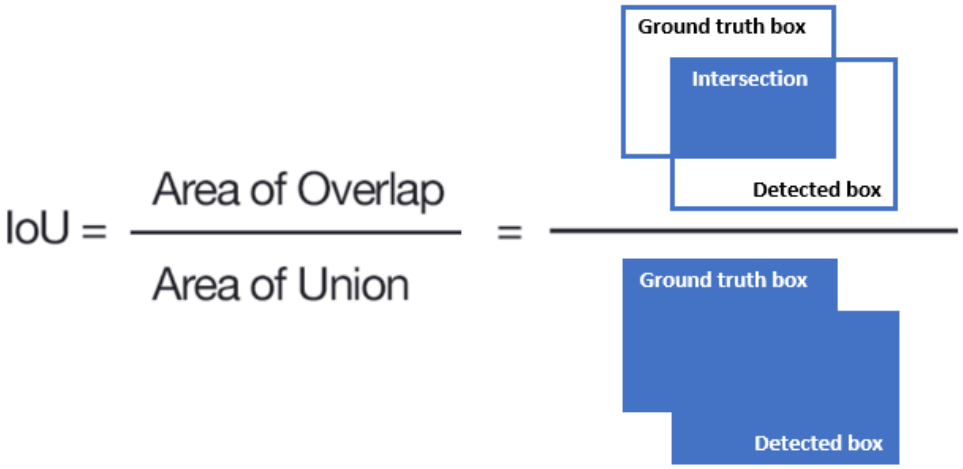
Adaptado de: (ANWAR, 2022)

- Average Precision (AP): Um modelo pode recuperar muitos positivos (alta sensibilidade), mas incluir muitas classificações erradas (baixa precisão) e vice versa. A curva Precisão vs Sensibilidade é uma representação gráfica deste *trade-off*. AP é calculada como sendo a área sob a curva Precisão vs Sensibilidade para diferentes valores (limiar) de confiança. A escolha dos valores de confiança pode-se basear na literatura ou pode-se utilizar todos os valores únicos. Valores mais altos de AP indicam melhor desempenho. A AP pode ser obtida pela equação 2.4, sendo, n o número de pontos na curva Precisão vs Sensibilidade, correspondendo a diferentes limites de confiança, R_n o valor de sensibilidade no n-ésimo ponto da curva e P_n o valor de precisão no n-ésimo ponto da curva. A Figura 14 ilustra a construção da curva Precisão vs Sensibilidade, também chamada de curva PR (*Precision Recall Curve*).

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{2.4}$$

Por exemplo, imaginemos um problema multi-classe, a AP é calculada para cada classe individualmente tratando-se o problema como uma classificação binaria. Um valor de limiar de IoU deve ser definido para se classificar as predições como acertos ou erros. Para a classe positiva i defini-se a classe negativa como todas as classes $j \neq i$. Calcula-se a precisão e sensibilidade para diferentes valores de confiança e plota-se a curva precisão vs sensibilidade. A AP_i é então a área em baixo desta curva segundo a equação 2.4.

Figura 15 – Intersecção sobre a união



Fonte: (BAELDUNG, 2023)

A mAP é calculada como sendo a média dos valores de AP para todas as classes, assim, proporcionando uma medida geral do desempenho do modelo. Um mAP alto indica que um modelo tem um bom desempenho de localização e classificação em várias classes. A mAP pode ser obtida pela equação 2.5, sendo k o número de classes no conjunto de dados e AP_i o valor de AP para a i-ésima classe.

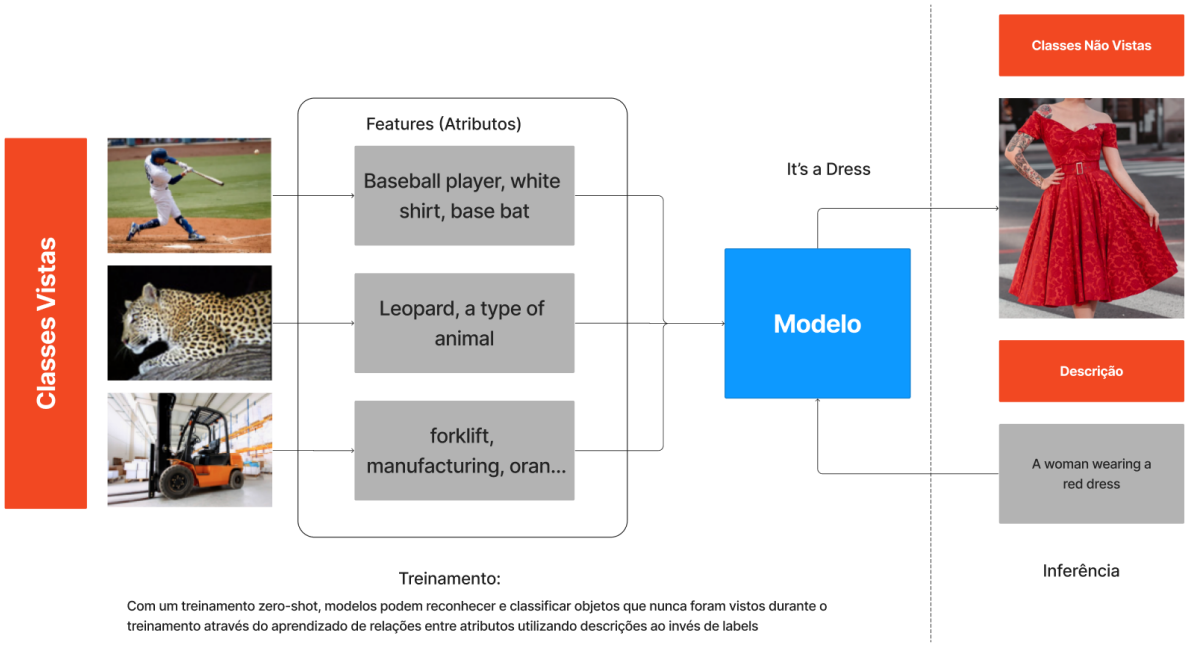
$$\text{mAP} = \frac{1}{k}\sum_{i}^{k} AP_i \tag{2.5}$$

A mAP pode ser calculada para diversos valores de IoU. Define-se mAP@0.5 a mAP calculada considerando-se um limiar de 0.5.

2.4 ZERO-SHOT LEARNING

Em aprendizado de máquina, *Zero-shot learning* é um paradigma onde um modelo é treinado em uma grande base de dados gerais para posteriormente, reconhecer ou classificar objetos, conceitos ou categorias que nunca foram vistas durante o treinamento. Em outras palavras, o modelo é capaz de realizar tarefas de classificação mesmo quando ele não tem exemplos prévios das classes ou conceitos a serem reconhecidos. Isso é alcançado através do uso de informações adicionais durante o treinamento, como descrições textuais das classes ou atributos, que permitem ao modelo generalizar e tomar decisões sobre novas classes com base em seu conhecimento prévio (Figura 16). O objetivo do *Zero-Shot learning* é estender a capacidade de um modelo além das classes específicas com as quais ele foi treinado, tornando-o mais flexível e adaptável a cenários do mundo real onde novas classes podem surgir.

Figura 16 – Capacidade de predição Zero-Shot



Fonte: Autor

2.5 CONTRASTIVE IMAGE-LANGUAGE PRE-TRAINING

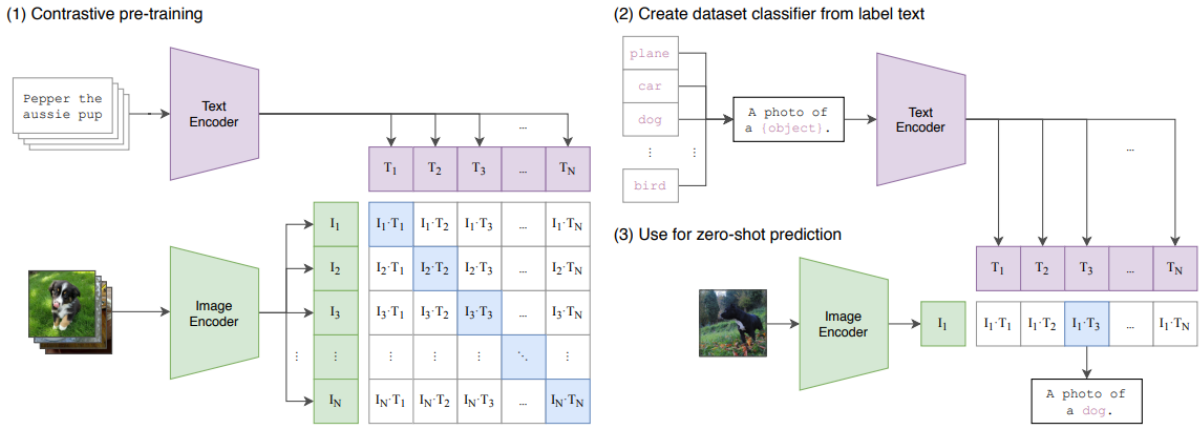
Classicamente, modelos de visão computacional para classificação são treinado em um conjunto fixo e pré-determinados de categorias (classes), no entanto, esta abordagem limita a usabilidade e capacidade de generalização do modelo, uma vez que para adicionar-se um novo conceito visual, se faz necessário o uso de dados adicionais anotados referentes a este novo conceito. Desenvolvido pela OpenAI, o CLIP (*Contrastive Language-Image Pre-training*) é um método de aprendizado treinado diretamente a partir de imagens e descrições em forma de texto. O método revela que pré-treinar um modelo para prever qual descrição combina mais com determinada imagem é uma maneira eficiente e escalável de se aprender representações de imagens, que se compara a modelos do estado da arte (RADFORD et al., 2021).

O CLIP se destacou como um modelo base (*foundation model*) em visão computacional, abrindo novas possibilidades em aplicações de classificação de imagens, uma vez que os modelos CLIP pré-treinados podem ser ajustados para tarefas específicas com relativamente poucos exemplos de treinamento. Isso o torna uma alternativa promissora para o treinamento de modelos de classificação de imagens, especialmente quando o conjunto de dados de treinamento é limitado. Modelos CLIP também possuem o paradigma *zero-shot learning*, neste contexto, a inferência *zero-shot* para classificação de k-classes, corresponde a: dada uma imagem e um conjunto de classes (potenciais descrições para a imagem), encontra-se a classe com maior similaridade com a imagem. Com isso modelos CLIP podem ser utilizados em tarefas específicas, sem a necessidade de treinamento adicional.

Em um estudo conduzido pelos autores do método, envolvendo a comparação entre um modelo CLIP e modelos de *linear probing* utilizando uma Resnet50 pré-treinada no ImageNet, o modelo *zero-shot* CLIP (modelo CLIP com paradigma *zero-shot learning*) supera o *baseline* em 16 dos 27 conjuntos de dados utilizados. Em um segundo estudo, medindo o desempenho da realização de *linear probe* (definição no 3º parágrafo da seção 2.2.6) de modelos CLIP em comparação com modelos de visão computacional do estado da arte (EfficientNet, MoCo, modelos ResNeXt, BiT, ViT, SimCLRv2 dentre outros), o modelo ViT-L/14 (CLIP) obteve o estado da arte em 21 dos 27 conjuntos de dados (RADFORD et al., 2021). A Figura 17 apresenta a ideia central do método (matriz de similaridade) e a inferência *zero-shot*.

O método CLIP original pode ser compreendido através do entendimento de 4 conceitos, o aprendizado contrastivo multimodal, o *tokenizer*, os codificadores de imagem e texto e a perda contrastiva.

Figura 17 – CLIP: Contrastive Image-Language Pre-training



Fonte: (RADFORD et al., 2021)

2.5.1 Aprendizado contrastivo multimodal

O aprendizado contrastivo é um paradigma de treinamento de modelos que se baseia na ideia de que o aprendizado de boas representações (*features*) pode ser alcançado ao aprender a distinguir entre exemplos similares e dissimilares de dados. Essa abordagem é amplamente utilizada no campo do processamento de linguagem natural (NLP) e na visão computacional.

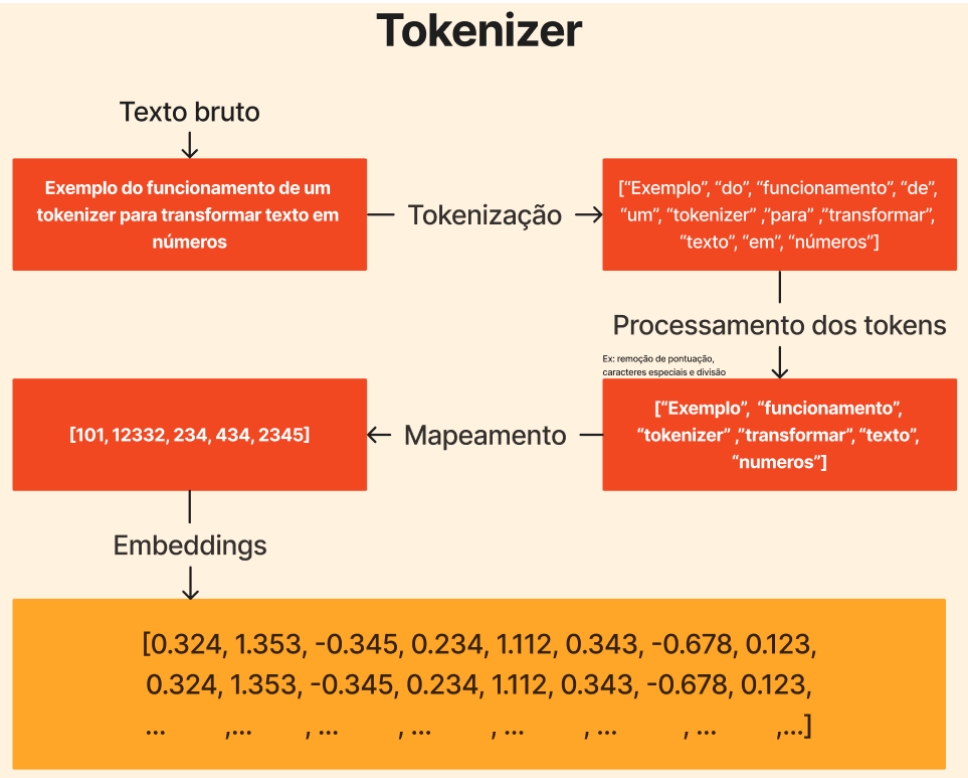
No CLIP, o aprendizado contrastivo é estendido para uma versão multimodal (entre diferentes tipos de dados), sendo implementado por meio da maximização da similaridade entre pares de imagem e texto positivos, por exemplo, o texto “A foto de um lobo cinzento” e uma fotografia de um lobo cinzento, e minimização da similaridade entre pares negativos, por exemplo, o texto “A foto de um camaleão” e uma fotografia de uma borboleta. Essa

abordagem permite que o CLIP aprenda a representar conceitos visuais e semânticos em um espaço comum, tornando-o extremamente versátil em tarefas de classificação de imagens (RADFORD et al., 2021).

2.5.2 *Tokenizer*

O *tokenizer* (Figura 18) é uma ferramenta essencial no processamento de linguagem natural (NLP) que converte texto em uma forma que um modelo de ML pode entender e processar (números). O termo *"tokenizer"* deriva de *token*, que é uma representação numérica de uma palavra ou pedaço de texto. Seu funcionamento se assemelha ao de um dicionário e é limitado em termos de representações. Este vocabulário essencialmente contém um número finito de pares chave-valor, onde as chaves são os *tokens* (palavras) e os valores são as representações numéricas correspondentes. Quando um *tokenizer* processa um texto, ele mapeia cada *token* encontrado no texto para sua representação numérica correspondente no vocabulário. Qualquer *token* que não esteja presente no vocabulário é substituído por um *token* especial como "desconhecido" ou dividido em sub *tokens* que estão presentes no vocabulário

Figura 18 – Tokenizer



Fonte: Autor

2.5.3 Codificadores

Os codificadores são redes constituídas de camadas convolucionais, *transformers*, ou uma combinação de ambos. De forma simplificada, o papel dos codificadores é o de extrair características dos dados de entrada, gerando uma descrição apropriada para a entrada de forma a minimizar a função perda do modelo. Para os codificadores determinísticos, o termo *embeddings* é geralmente utilizado para se referir às representações aprendidas.

2.5.4 Perda contrastiva

Dado um conjunto de N pares correspondentes de imagem e texto, o CLIP é treinado para prever quais pares dentre os possíveis $N \times N$ pares são correspondentes. Para isso o CLIP aprende um espaço de representação multi-modal, treinando um codificador de imagem e texto para maximizar a similaridade do cosseno entre os N pares correspondentes e minimizar a similaridade entre os $N^2 - N$ demais possíveis pares (RADFORD et al., 2021). A perda contrastiva consiste da otimização de uma perda de entropia cruzada (*cross entropy*) dos valores de similaridade entre cada texto e imagem, conforme o pseudocódigo ilustrado na Figura 19.

A similaridade do cosseno ($\cos\theta$) entre dois vetores de características A e B é dada pela equação 2.6. Ao calcular-se a similaridade entre cada possível par, obtém-se uma matriz $N \times N$, onde as linhas representam a similaridade de cada imagem com cada texto, enquanto as colunas, o inverso, representam a similaridade de cada texto com cada imagem, conforme a matriz da Figura 17.

$$\cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} \tag{2.6}$$

A perda de entropia cruzada para uma amostra o é dada pela equação 2.7, onde M representa o número de classes, c a classe, log, o logaritmo natural, $p_{o,c}$ a probabilidade da amostra o pertencer a classe c e $y_{o,c}$ um indicador binário, sendo 1 (um) caso a classe c seja o rótulo da amostra o e 0 (zero) caso contrario. Ou seja, a perda de entropia cruzada para uma amostra o de rótulo c é dada por $-\log p_{o,c}$.

$$\text{cross\_entropy} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2.7}$$

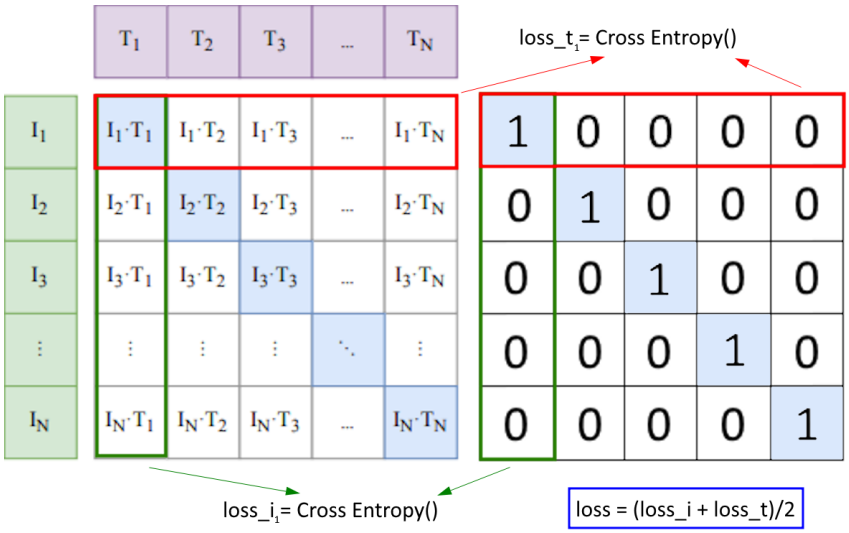
Como perda final, calcula-se a média entre a soma da entropia cruzada de cada linha (linha 1,2,...,N) e a soma da entropia cruzada de cada coluna (coluna 1,2,...,N), da matriz de similaridades com a matriz de rótulos, conforme ilustrado na Figura 20.

Figura 19 – Cálculo da perda em pseudocódigo

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Fonte: (RADFORD et al., 2021)

Figura 20 – Perda simétrica



Adaptado de: (RADFORD et al., 2021)

2.5.5 Inferência *zero-shot*

Um exemplo de como a inferência *zero-shot* é realizada no CLIP pode ser visto na Figura 17 a direita. Primeiro realiza-se a escolha de k-classes. As classes são convertidas para o formato descrição, isto é, são transformadas em um texto que pode descrever uma imagem da classe desejada, por exemplo para a classe "cachorro", uma possível descrição é: "A foto de um cachorro". Obtém-se os vetores de *embeddings* para cada descrição (classe) e para a imagem. Utilizando-se dos *embeddings* obtidos, calcula-se a similaridade do cosseno da imagem com cada texto e atribui-se a classe com maior similaridade entre o texto e a imagem com sendo o rótulo previsto.

3 MATERIAIS E MÉTODOS

3.1 DEEP FASHION E DEEP FASHION 2

O trabalho realizado por (LIU et al., 2016), apresenta o Deep Fashion, um conjunto de dados de roupas com mais de 800 mil imagens, que são anotadas com categorias, atributos e correspondências entre imagens de diferentes cenários (Figura 21). O conjunto de dados visa facilitar a pesquisa em reconhecimento e recuperação de roupas. Neste trabalho sera utilizado o subconjunto **Deep Fashion Category and Attribute Prediction Benchmark**, disponibilizado pelos autores mediante solicitação. O conjunto possui aproximadamente 290 mil imagens de peças de roupas em 50 diferentes categorias (classes), e anotações de *bounding boxe* (BB) para cada peça anotada na imagem. Denomina-se neste trabalho, por simplicidade, Deep Fashion (DF1) como sendo o conjunto **Deep Fashion Category and Attribute Prediction Benchmark**.

Figura 21 – Deep Fashion

Fonte: (LIU et al., 2016)

O conjunto Deep Fashion 2 (DF2), dos autores (GE et al., 2019), contém 491 mil imagens de 13 categorias populares de roupas, com 801 mil itens anotados com informações como estilo, escala, oclusão, zoom, ângulo de visão, BB, máscara de pixel e pares de imagens de consumidores e de lojas (Figura 22). O conjunto permite o desenvolvimento de algoritmos para diversas tarefas de visão computacional no campo da moda, como detecção, estimativa de pose, segmentação e *retrieval* e foi utilizado neste trabalho por possuir anotações de BB superiores ao conjunto DF1.

Figura 22 – Deep Fashion 2

Fonte: (GE et al., 2019)

3.2 PRÉ-PROCESSAMENTO

O conjunto de dados utilizado neste trabalho, foi criado a partir do conjunto DF1. Primeiramente analisou-se as classes disponíveis e definiu-se 16 classes de interesse (Figura 23). Alguns das classes de interesse foram criadas através do agrupamento de classes similares. Classes indesejadas ou com poucos exemplos foram eliminadas.

Figura 23 – Classes de interesse

Fonte: Autor

Para formar um conjunto de dados para classificação, as imagens resultantes do processo anterior foram recortadas com o uso das anotações disponíveis de *bounding boxes* (BB). Notou-se no entanto que, as BB provindas do DF1 possuem inconsistências. Algumas anotações de BB possuem uma área consideravelmente superior a área do objeto de interesse (Figura 25), enquanto outras cortam o objeto de interesse, ocasionando em perda de informação (Figura 26). Levantou-se a hipótese de que esta inconsistência no tamanho da BB poderia prejudicar ou enviesar o aprendizado do modelo. De forma a minimizar estes casos, um modelo de detecção YOLO foi treinado no conjunto de dados DF2, que possui uma qualidade de anotações de BB superior a primeira versão. Para o treinamento da rede YOLO, utilizou-se o modelo pré-treinado YOLOv5m, com as configurações de hiperparâmetros padrões média, tamanho de imagem igual a 640 e um tamanho de *batch* de 16. O modelo foi treinado por 100 épocas (número de vezes em que todo o conjunto de dados é processado pelo modelo). Os resultados do treinamento estão disponíveis na Figura 36. Utilizou-se então esta rede para realizar inferência em todas as imagens do conjunto DF1 e obter novas anotações de BB. De forma a confirmar a qualidade superior das novas BB, 100 imagens aleatórias de cada classes foram visualizadas. Na Figura 24, pode-se visualizar a comparação de algumas amostras com as anotações novas (em vermelho) e as originais (em azul). Uma particularidade do conjunto de dados DF1, é de que cada imagem possui apenas uma anotação de roupa, isto é, mesmo em uma imagem onde uma pessoa veste uma blusa e calça, há somente anotações para uma das peças de roupa. De forma a ser possível a utilização das novas anotações de BB, um algoritmo foi escrito para verificar se a nova anotação corresponde a mesma peça da anotação anterior, assim, atribui-se o mesmo rótulo de classe. As demais anotações (de BB) de peças com classes não anotadas pelo DF1 são descartadas. Desta forma, as anotações finais de BB consistem na combinação das anotações novas (YOLO) e originais (DF1).

Algorithm 1 Melhoria das caixas delimitadoras

```
for Every image do
    Load YOLO detections and DF1 labels                ▷ Current image
    maximum = max{IoU(YOLO detections, DF1 labels)}    ▷ Best IoU
    if maximum ≥ 0.5 then
        Save new annotation                            ▷ Replaces old one
    else
        Keep original annotation
    end if
end for
```

Figura 24 – Exemplos de novas anotações (vermelho)

Fonte: Autor

Figura 25 – Exemplos de anotações (DF1) “largas”

Fonte: Autor

Figura 26 – Exemplos de anotações (DF1) com perda de informação

Fonte: Autor

A combinação das anotações já existentes do DF1 com as anotações provindas da rede treinada é descrita em pseudocódigo pelo algorítimo 1. Para cada imagem, obtém-se o par de predições de BB da YOLO e as anotações do DF1, calcula-se a IoU (Intersection over Union) entre cada par de anotação. Compara-se o maior valor de IoU com um valor de *threshold*, em caso maior ou igual (264773 casos), a predição da YOLO é considerada como um *matching* e correta, passando a ser a nova anotação. Em caso contrario (7961 casos), mantém-se a anotação anterior (original do DF1). Desta forma obtém-se novas anotações conservadoras de BB, i.e, a nova anotação possui um valor mínimo de IoU com a anotação antiga, assim, pode-se atribuir o mesmo rótulo de classe.

Tabela 3 – Mapeamento das super-classes

Classe (ID)	**Super-Classe**
Blazer (0), Blouse (1), Cardigan (2) Hoodie (5), Jacket (6), Sweater (13) Sleeveless Shirt (14), T-Shirt (15)	Top
Coat (3), Dress (4), Jumpsuit (7) Kimono (8), Pair of Pants (9), Romper (10)	Full + Pants
Shorts (11)	Shorts
Skirt (12)	Skirt

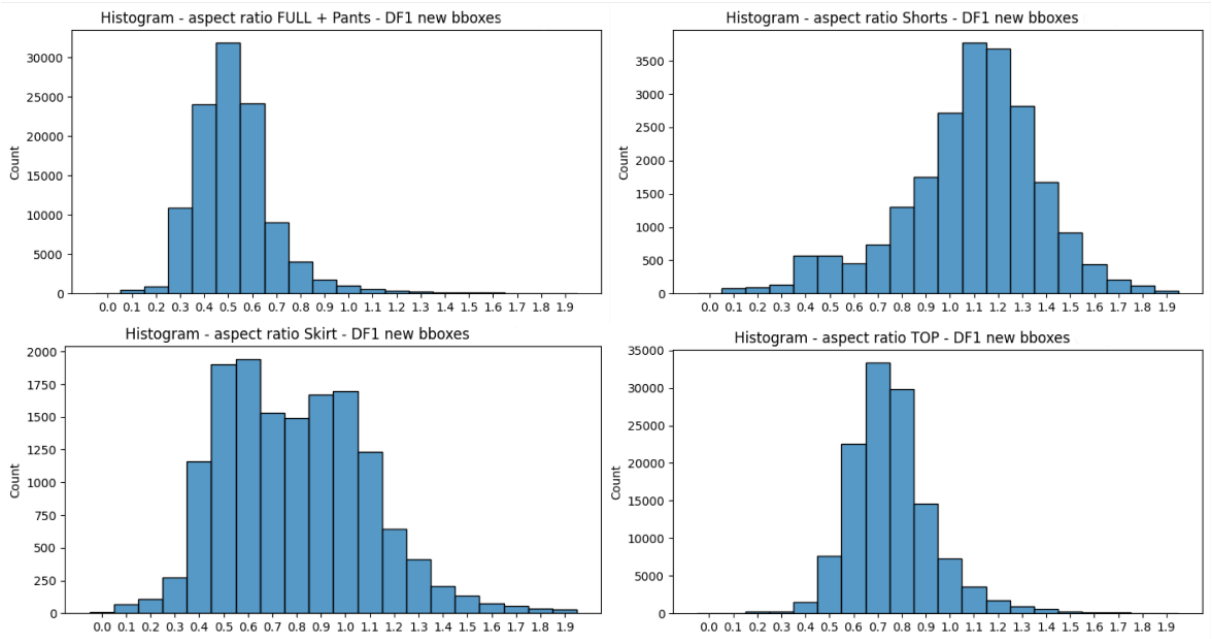
Figura 27 – Exemplos de anotações ainda incorretas

Fonte: Autor

Figura 28 – Calculo do AR e Resolução

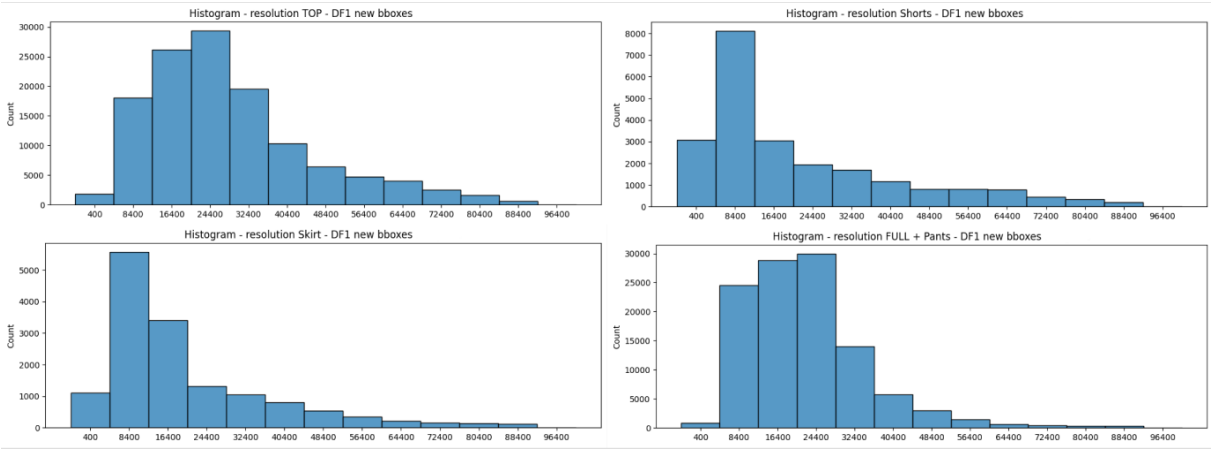
Fonte: Autor

Figura 29 – Histograma do AR para cada super-classe



Fonte: Autor

Figura 30 – Histograma de resoluções para cada super-classe



Fonte: Autor

Analisando-se algumas amostras, constatou-se anotações de BB errôneas, tanto para a BB antiga, quanto para a nova, uma vez que, o método conservador apresentado para a criação de novas anotações parte do pré-suposto de que apenas pequenas/médias (parâmetro controlado pelo *threshold* da IoU) correções são necessárias, desta forma, não corrigindo anotações totalmente incorretas, como exemplos vistos na Figura 27. Para melhorar ainda mais a qualidade dos dados, utilizou-se a criação de regras (algorítimo 2) baseadas na resolução das imagens e *aspect-ratio*. O *aspect-ratio* é obtido pela razão entre a largura e altura de uma imagem, enquanto a resolução corresponde ao produto (Figura 28). Conforme a Tabela 3, foram determinadas 4 super-classes (Junção de classes), Full + Pants, Shorts, Skirt e Top, agrupadas conforme a similaridade do *aspect-ratio* (AR) entre as classes (Figura 29). Com isso foi possível determinar um intervalo de AR esperado para cada super-classe. Assim realizou-se uma filtragem de forma a eliminando dados com anotações incorretas (IoU baixa ou nula quando comparado com uma anotação ideal), seguido da eliminação de imagens com resolução muito abaixo da média para cada super-classe (Figura 30). Novamente, analisou-se 100 imagens aleatórias de cada classe, comparando a anotação antiga com a nova. Concluiu-se que as novas anotações são em sua maioria superiores a anterior, não constatou-se anotações de BB incorretas após a filtragem.

Algorithm 2 Regras

$r\_thr \leftarrow 8000$ ▷ Resolution threshold

▷ Define conditions (class) : *rule*

$conditions \leftarrow \{$

$(0, 1, 2, 5, 6, 13, 14, 15) : (r \geq r\_thr$ **and** $0.45 < ar < 1.05)$, ▷ Top

$(3, 4, 7, 8, 9, 10) : (r \geq r\_thr$ **and** $0.25 < ar < 0.85)$, ▷ Full + Pants

$(11,) : (r \geq r\_thr$ **and** $0.35 < ar < 1.65)$, ▷ Shorts

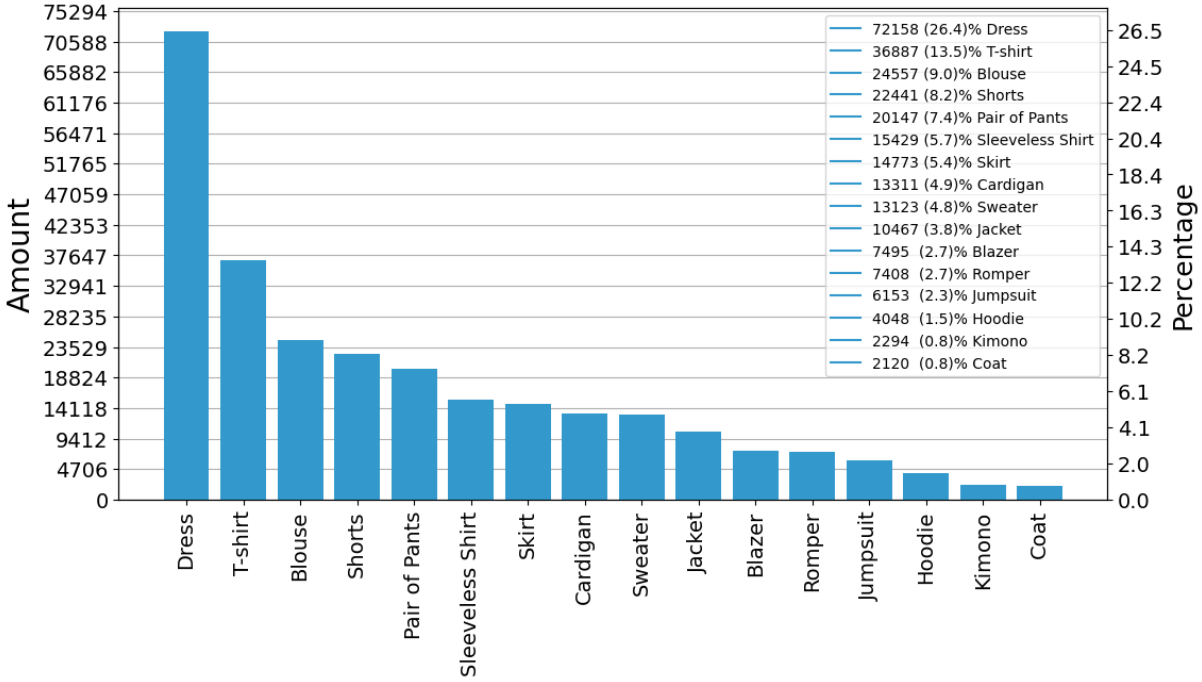
$(12,) : (r \geq r\_thr$ **and** $0.25 < ar < 1.35)$ ▷ Skirt

$\}$

return True if the condition is satisfied, else False

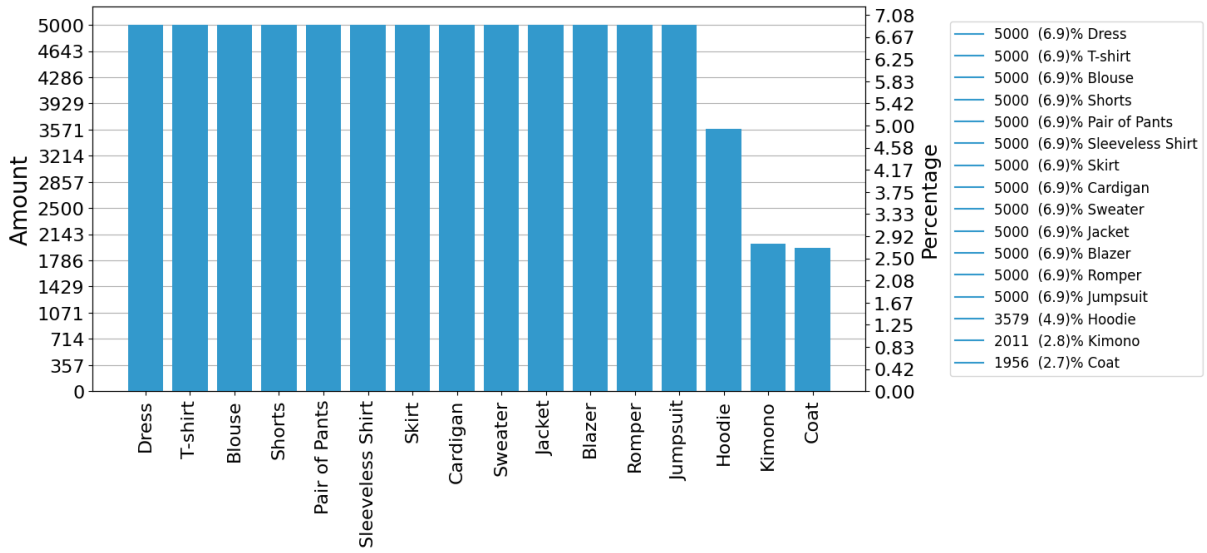
O histograma de classes resultante dos processos descritos acima pode ser visto na Figura 31 e contém um total de 272811 imagens. Os dados foram divididos em subconjuntos de treino (Figura 32), validação e testes, utilizando-se um processo aleatório. De forma a evitar um grande desbalanceamento entre classes, para o conjunto de treino, limitou-se o número de exemplos de cada classe em 5000, totalizando 72546 imagens para treinamento. Optou-se por criar conjuntos balanceados para validação e testes, ambos possuem 3200 imagens cada, sendo 200 imagens por classe. Com isso foram utilizadas um total de 78946 imagens, dividas nestes 3 conjuntos.

Figura 31 – Histograma de classes da totalidade dos dados



Fonte: Autor

Figura 32 – Distribuição do conjunto de treinamento



Fonte: Autor

3.2.1 Comparação entre anotações de *bounding boxes*

De forma a verificar se os procedimentos realizados na seção 3.2 convertem-se em ganhos de acurácia, um experimento utilizando-se a **ResNet18** foi conduzido. A rede foi treinada utilizando-se o conjunto de dados desenvolvido com as anotações originais e com as anotações novas. Ambos os modelos foram otimizados e são testados em um mesmo conjunto de dados de teste. Os resultados da Tabela 4 demostram um ganho de **2.66** pontos percentuais em acurácia.

Tabela 4 – ResNet18 - Acurácia vs anotações novas vs originais

Anotações de BB	**Acurácia (%)**
Originais (DF1)	69.37
Novas (YOLO+DF1)	**72.03**

3.3 EXPERIMENTOS

Os experimentos conduzidos visam comparar a acurácia de modelos pré-treinados para a classificação de imagens no conjunto de dados de moda desenvolvido, concentrando-se na comparação entre modelos convencionais pré-treinados no conjunto de dados ImageNet-1k, e o modelo CLIP, que utiliza aprendizado contrastivo para representações visuais. As arquiteturas (Tabela 5) e pesos utilizados para *transfer learning* estão disponíveis em *PyTorch Vision Models*[1] e *Open CLIP Repository*[2]. Para o treinamento de todos os modelos, a GPU NVIDIA GeForce RTX 3090 foi utilizada.

3.3.1 Treinamento das Arquiteturas Convencionais

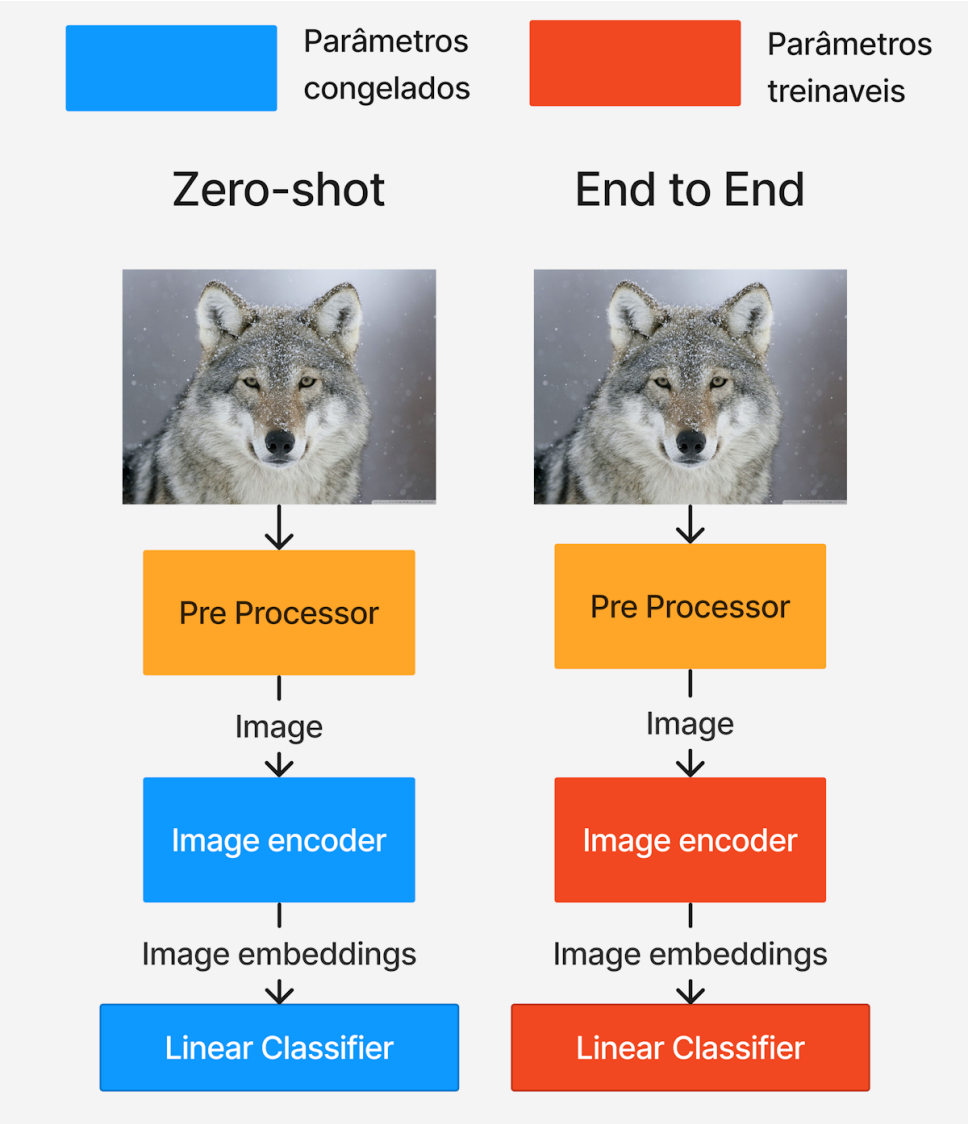
Para o treinamento ponta a ponta (*end to end*) das arquiteturas convencionais, a ultima camada de classificação, contendo as 1000 classes do ImageNet-1k é removida, dando espaço para uma nova camada linear, com dimensão de saída igual ao numero de classes (16). Para não prejudicar as demais camadas do modelo com as grandes valores de gradientes da camada linear (que ainda não possui nenhum aprendizado), duas épocas de aquecimento (*warmup*) são utilizadas. O *warmup* consiste em congelar todas as camadas do modelo, exceto a camada de classificação e realizar o treinamento por algumas épocas. Durante o treinamento, as métricas acurácia e sensibilidade são calculadas e registradas. A otimização de hiperparâmetros é conduzida para cada arquitetura, empregando uma busca sistemática no espaços de hiperparâmetros (Figura 35) relacionados ao otimizador, função de perda e técnicas de *data augmentation* (aumento de dados de forma artificial) em um subconjunto dos dados de treinamento. Este procedimento resultou na obtenção dos modelos finais, cada um caracterizado por seus hiperparâmetros otimizados (Tabela 9). A métrica principal de avaliação adotada foi a acurácia.

[1] Pytorch models
[2] Open CLIP

Tabela 5 – Arquiteturas

Arquitetura	**Nº de parâmetros**	**Conjunto**	**Fonte dos pesos**
ResNet-18	11.7M	ImageNet-1K	pytorch.org
EfficientNet-B2	9.1M	ImageNet-1K	pytorch.org
EfficientNet v2 S	21.5M	ImageNet-1K	pytorch.org
ViT-B/16 (Pytorch)	86.6M	ImageNet-1K	pytorch.org
ViT-B/16 (OpenAI)	86.2M	-	OpenAI

Figura 33 – Modelo de classificação utilizando a arquitetura CLIP



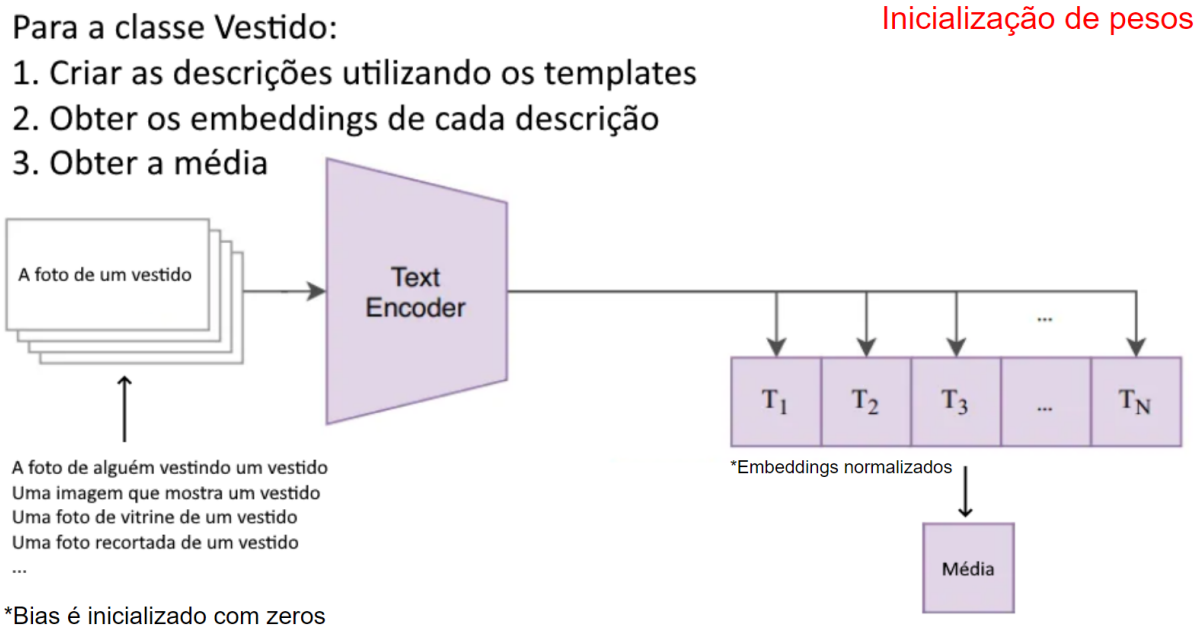
Fonte: Autor

3.3.2 Experimentos com CLIP Zero-shot

Para os modelos treinados com o método CLIP, utiliza-se o codificador de imagens como extrator de *features* e uma camada linear (de classificação) é construída seguindo a

mesma abordagem anterior, porém, para sua inicialização, o próprio modelo pré-treinado é utilizado, obtendo-se uma media das *features* de cada classe seguindo a metodologia utilizada por (WORTSMAN et al., 2022).

Figura 34 – Inicialização de pesos da camada de classificação para o modelo CLIP



Fonte: Autor

Para a obtenção dos *embeddings* (*features*) de cada classe, utiliza-se *templates* de descrição. Os *templates* de descrição são frases onde a palavra referente a classe de interesse é inserida, de forma a utilizar-se o codificador de texto pré-treinado para obter-se um vetor de *embeddings*. Para cada classe, utiliza-se a média dos *embeddings* de cada *template* para se obter uma representação não enviesada da classe (WORTSMAN et al., 2022). Desta forma obtém-se um modelo de classificação *zero-shot*, conforme a Figura 33. Na Figura 34 pode-se visualizar um exemplo dos passos seguidos para a obtenção dos pesos iniciais para a classe “Vestido”.

3.3.3 Treinamento do modelo CLIP

Para o treinamento *end to end* do modelo CLIP, como ilustrado na Figura 33, os parâmetros do codificador de imagens e da camada linear descrita anteriormente são descongelados e otimizados. Utiliza-se a abordagem convencional para o treinamento de classificadores de imagens. Duas épocas de *warmup* são utilizadas. Durante o treinamento, as métricas acurácia, e sensibilidade são calculadas e registradas. A otimização de hiperparâmetross é conduzida conforme descrito anteriormente na seção 3.3.1. Este procedimento resultou na obtenção do modelo CLIP *end to end* final, caracterizado por seus hiperparâmetros otimizados (Tabela 9).

3.3.4 Aumento de dados e espaço de otimização

Três arquivos de configuração para *data augmentation* são utilizados, baixo, médio e alto, contendo as transformações apresentadas na Tabela 6. Os valores de cada parâmetro para cada nível podem ser vistos na Tabela 7. Os hiperparâmetros de cada modelo foram otimizados seguindo o método de busca Bayesiano, implementado na biblioteca *Weights and Biases* com os parâmetros listados na Figura 35.

Tabela 6 – Transformações e parâmetros

Transformação	**Parâmetros**
Random Horizontal Flip	Probabilidade
Random Affine	Porcentagem de translação escala, rotação probabilidade
Color Jitter	Brilho, Contraste Saturação, HUE Probabilidade
Random Perspective	Escala de distorção Probabilidade
Gaussian Blur	Limite do Blur Limite do sigma Probabilidade
Random Adjust Sharpness	Alfa, Iluminação Probabilidade

Tabela 7 – Transformações e valores

Transformação	**Baixo**	**Médio**	**Alto**
Random Horizontal Flip	Médio	Alto	Alto
RandomAffine	0.05 [0.9, 0.99] [0.9, 0.99] [-10, 10] 0.15	0.05 [0.9, 0.99] [0.9, 0.99] [-15, 15] 0.35	0.05 [0.9, 0.99] [0.9, 0.99] [-20, 20] 0.35
Color Jitter	0.1, 0.1, 0.05 0.05, 0.2	0.1, 0.1, 0.05 0.05, 0.2	0.1, 0.1, 0.05 0.1, 0.2
Random Perspective	[0.25, 0.3], 0	[0.10, 0.25], 0.1	[0.20, 0.25], 0.2
Gaussian Blur	[2,3], [0.1, 1.1], 0	[2,3], [0.1, 1.1], 0.05	[3,5], [0.1, 1.1], 0.1
Random Adjust Sharpness	[0.1, 0.3] [0.7, 1.0], 0	[0.1, 0.3] [0.7, 1.0], 0.2	[0.2, 0.5] [0.5, 1.0], 0.2

Figura 35 – Espaço de busca de hiperparâmetros

```yaml
parameters:
  warmup_epochs:
    value: 2
  warmup_lr:
    value: 0.002
  batch_size:
    value: 128
  epochs:
    value: 30
  hyps:
    values:
      - hyps_low
      - hyps_medium
      - hyps_high
  max_lr:
    distribution: uniform
    max: 2e-04
    min: 5e-06
  min_lr:
    value: 2e-07
  momentum:
    distribution: uniform
    max: 0.999
    min: 0.9
  patience:
    value: 4
  weight_decay:
    distribution: uniform
    max: 0.001
    min: 0.0001
```

Fonte: Autor

4 RESULTADOS

Neste capítulo, são apresentados os resultados referentes a cada sequência de experimentos, realizados com as arquiteturas da Tabela 5 em relação aos objetivos estabelecidos. A Tabela 8 resume os resultados obtidos para o conjunto de testes.

Tabela 8 – Resultados de teste para o conjunto de dados proposto

Arquitetura	**Ajuste**	**Pré-treino**	**Acurácia (%)**
Aleatório	-	-	06.25
ResNet-18	*end to end*	classificação	72.03
EfficientNet-B2	*end to end*	classificação	78.40
EfficientNet v2 S	*end to end*	classificação	79.96
ViT-B/16 (Pytorch)	*end to end*	classificação	<u>80.97</u>
ViT-B/16 (OpenAI)	*zero-shot*	contrastivo	68.16
ViT-B/16 (OpenAI)	*end to end*	contrastivo	**83.97**

Os resultados, apresentados na Tabela 8, destacam o desempenho superior do modelo **ViT-B/16 (OpenAI)** em comparação com outras configurações, atingindo uma acurácia de **83.97%**. Este resultado sugere que partir de modelos pré-treinandos com base na metodologia CLIP proporcionou melhorias na capacidade de classificação, ficando **3.00** pontos percentuais acima do segundo melhor modelo **(ViT-B/16 (Pytorch))**. A versão *zero-shot* do modelo CLIP, mesmo sem ter sido treinada diretamente nas classes de interesse, apresentou uma acurácia de **68.16%**, chegando próximo a **ResNet-18** com treinamento *end to end* **(72.03%)**. A arquitetura **EfficientNet v2 S** obteve um desempenho muito próximo ao modelo que utiliza *Vision transformers* **(ViT-B/16 (Pytorch))**, sendo sua diferença de apenas **1.01** pontos percentuais.

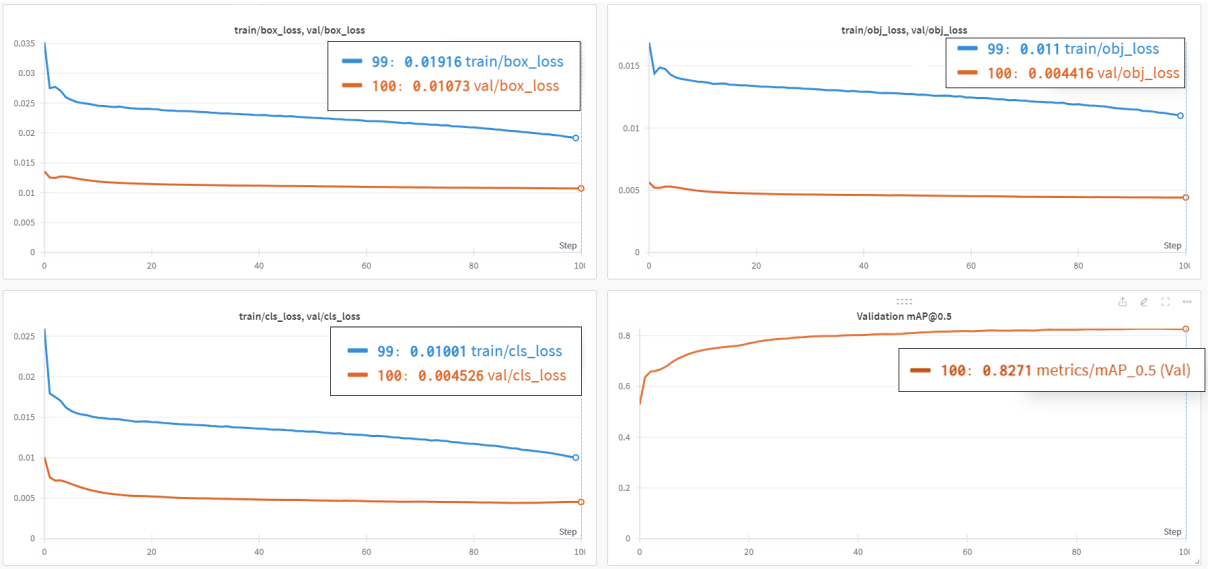
Tabela 9 – Melhores hiperparâmetros

Hiperparâmetro	**ResNet18**	**EN B2**	**EN v2 S**	**ViT-B/16 (Pytorch)**	**ViT-B/16 (OpenAI)**
Nível de aumento	Médio	Alto	Alto	Médio	Alto
Taxa de aprendizado máxima	$6.33 \cdot 10^{-05}$	$8.67 \cdot 10^{-05}$	$3.93 \cdot 10^{-05}$	$6.74 \cdot 10^{-05}$	$2.00 \cdot 10^{-06}$
Momento	0.9521	0.9021	0.9125	0.9182	0.9105
Decaimento	$8.35 \cdot 10^{-04}$	$6.00 \cdot 10^{-04}$	$5.09 \cdot 10^{-04}$	$6.73 \cdot 10^{-04}$	$1.55 \cdot 10^{-04}$

4.1 COMPARAÇÃO ENTRE MODELOS

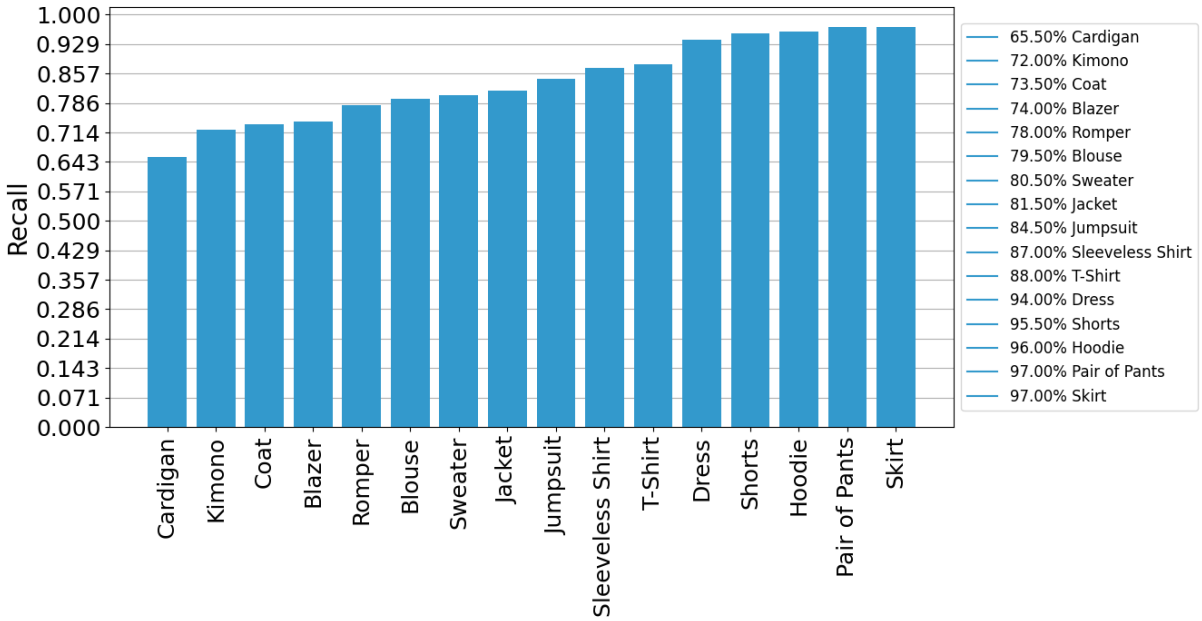
Nesta seção uma analise de erro é realizada, visando obter-se *insights* das limitações dos dados e do modelo. Esta analise se limita ao modelo **ViT-B/16 (OpenAI)**, que obteve os melhores resultados.

Figura 36 – Resultados - treinamento da YOLOv5m



Fonte: Autor

Figura 37 – Sensibilidade por classe | ViT-B/16 (OpenAI)

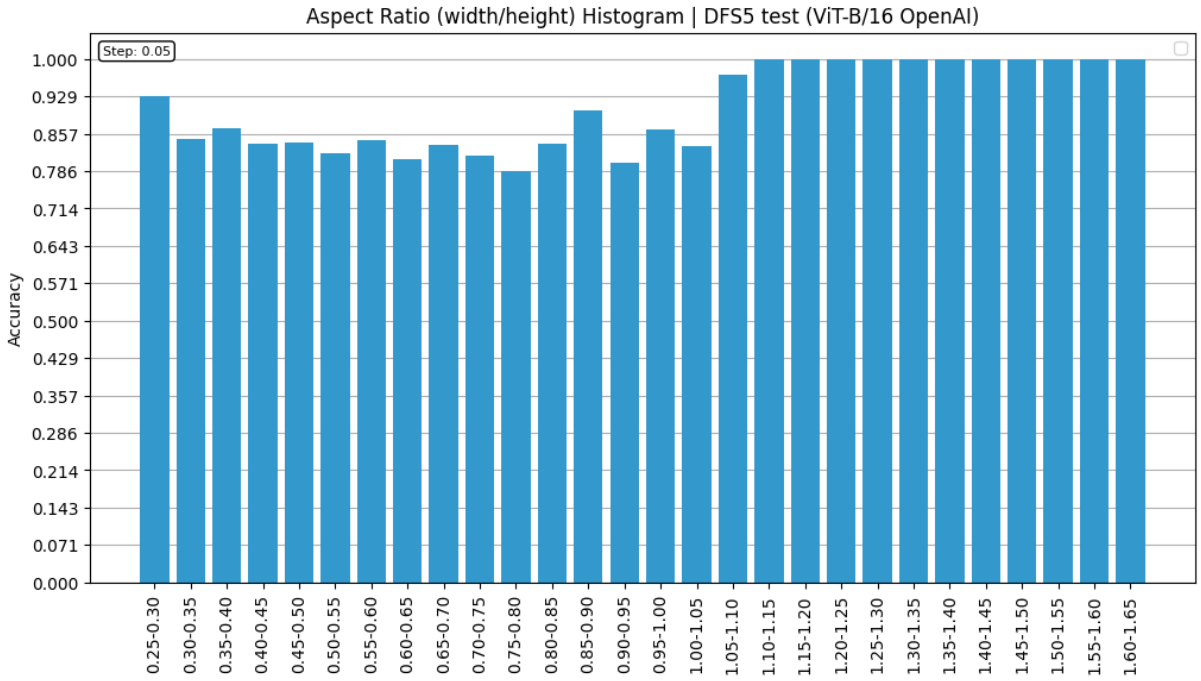


Fonte: Autor

Analisando a acurácia em função de intervalos de AR, na Figura 38, verifica-se que não existe uma grande relação entre estas duas grandezas. Para um AR superior a 1.10, o modelo obteve acurácia igual a 1. Verificando-se a Figura 29 percebe-se que estas amostras (AR superior a 1) são em sua maioria das classes *Shorts* e *Skirt*. Segundo a matriz de confusão na Figura 41, o modelo obteve um ótimo desempenho nestas classes, sendo condizente com a Figura 38. A alta sensibilidade para estas duas classes em especial, pode estar relacionada com seu AR, e o fato de serem classes mais fáceis de se identificar,

gerando menos confusão, por possuírem características consideravelmente diferentes das demais.

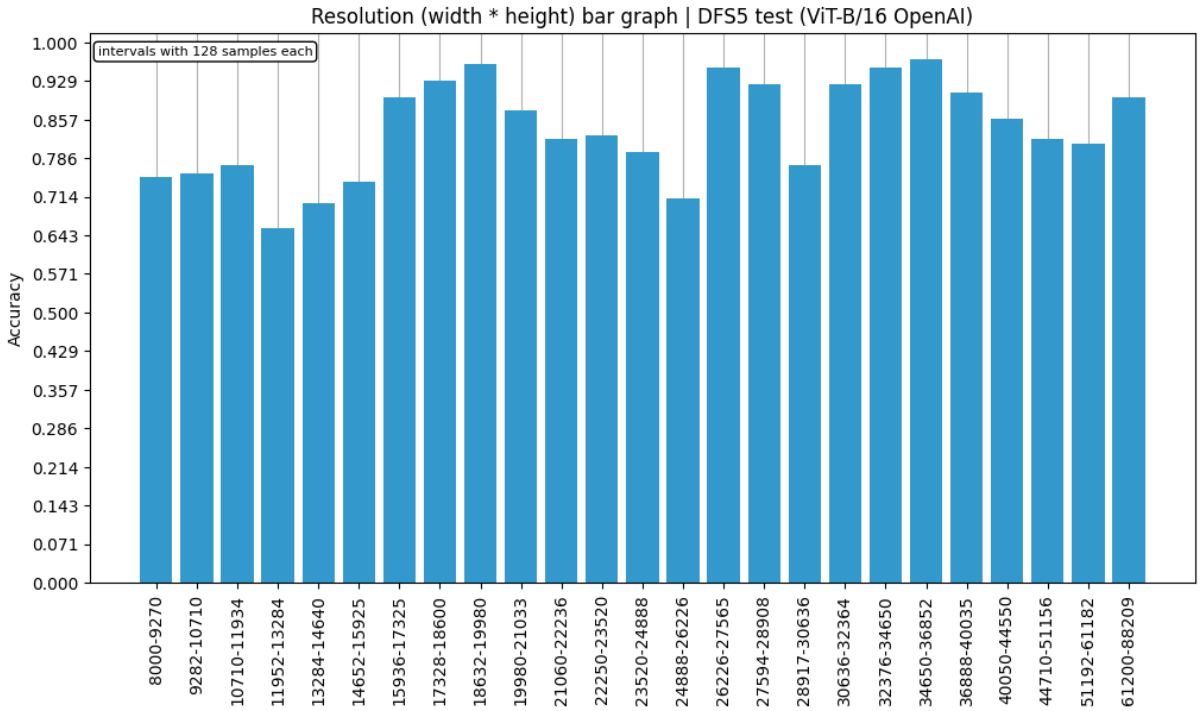
Figura 38 – Gráfico de barras de AR | ViT-B/16 (OpenAI)



Fonte: Autor

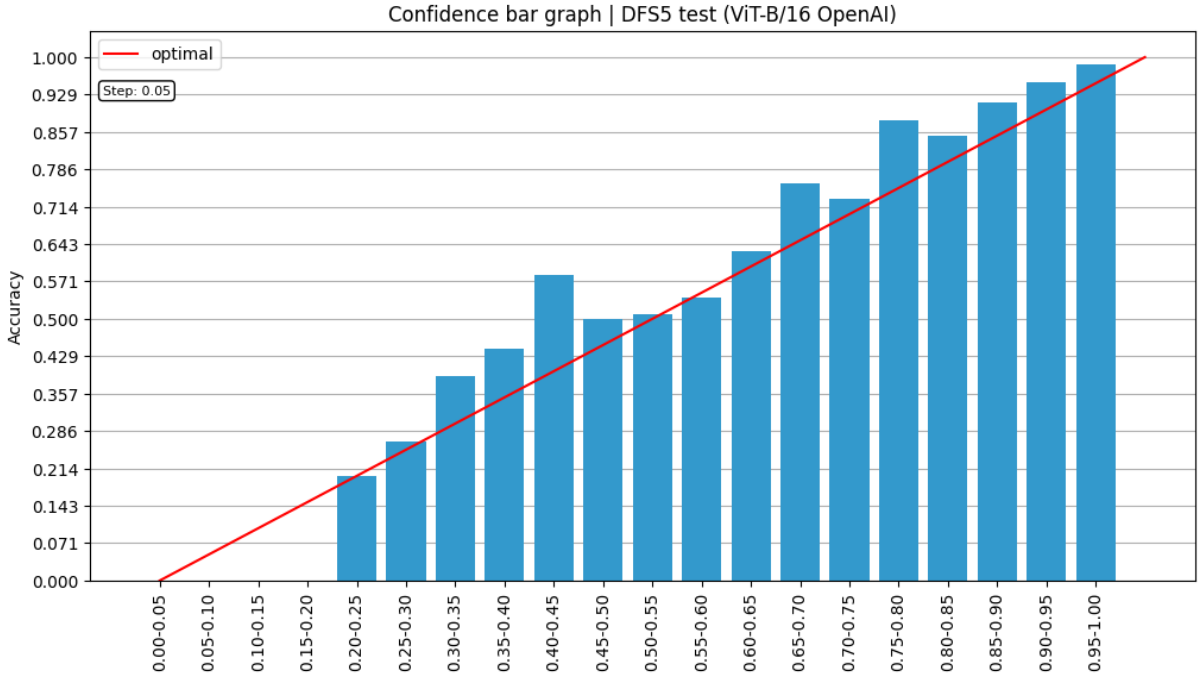
Ao treinar um modelo de classificação de imagens, é possível de que a resolução das imagens (antes da aplicação de *resize* para treinamento) possam influenciar a acurácia. De forma a verificar esta condição, a Figura 39 foi criada. Pela Figura 39 percebe-se que a resolução das imagem não estão relacionadas de forma considerável a acurácia, o que pode ser devido a filtragem realizada no pré-processamento do conjunto de dados, evitando-se imagens com resolução muito abaixo da média. A Figura 40 mostra o gráfico de barras do intervalo de confianças e acurácia. Como esperado, com o aumento da confiança do modelo, registra-se uma acurácia maior.

Figura 39 – Gráfico de barras de resolução | ViT-B/16 (OpenAI)



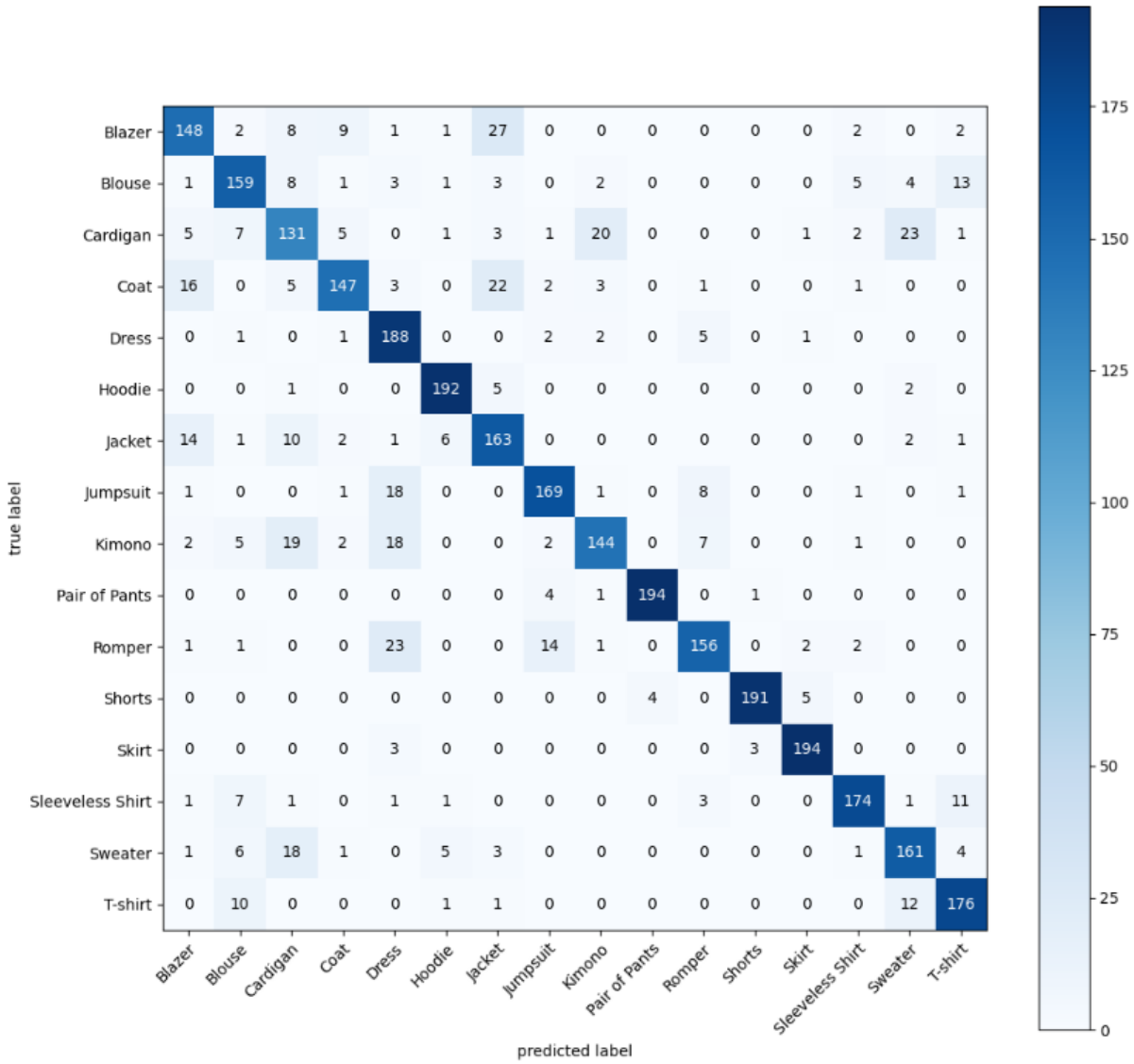
Fonte: Autor

Figura 40 – Gráfico de barras de confianças | ViT-B/16 (OpenAI)



Fonte: Autor

Figura 41 – Matriz de confusão | ViT-B/16 (OpenAI)



Fonte: Autor

4.1.1 Analise visual e qualitativa das classes com pior desempenho

Esta analise qualitativa e visual se limita as 5 classes com menor sensibilidade (*recall*), a sensibilidade para cada classe pode ser vista na Figura 37. As imagens destas classes onde houveram erros (predição diferente da anotação) são analisadas visualmente, realizando-se comentários quando necessário. Para esta analise, os erros do modelo são considerados levando em conta o conhecimento do autor, casos que geram dúvida (indefinidos) não serão classificados, uma vez que um conhecimento mais aprofundado no campo da moda é necessário para a correta definição das classes e classificação destes exemplos. Denomina-se amostras incorretas, as amostras que foram classificadas com uma classe diferente de sua anotação, e erros de anotação amostras onde é possível deliberar

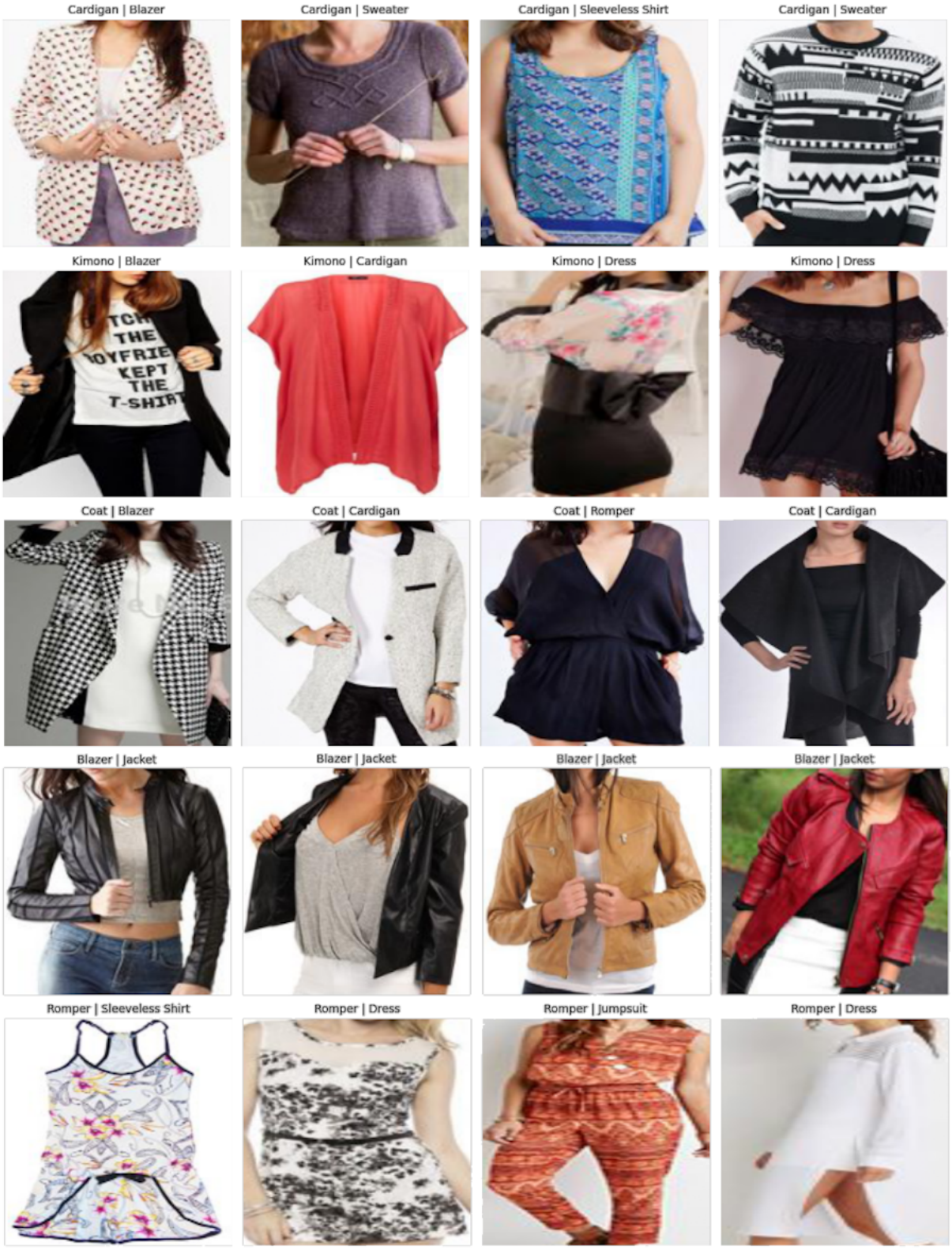
com facilidade que o modelo acertou na atribuição da classe, e o erro se deve a anotação incorreta da amostra.

A Tabela 10 apresenta de forma resumida a quantidade de amostras incorretas, erros do modelo, erros de anotação e amostras indeterminadas. Percebe-se que, dentro das amostras incorretas (predição diferente da anotação), a proporção de erros de anotações é grande, isso demostra que o conjunto de dados de testes contém uma quantidade de ruido significativa. O fato do modelo ter acertado amostras incorretamente anotadas, levanta a hipótese de que apesar do conjunto de dados ser ruidoso em relação aos rótulos de classe, o modelo consegue extrair bons vetores de atributos. O número de erros que podem ser facilmente identificados é consideravelmente baixo, evidenciando que o modelo aprendeu a lidar com os casos mais comuns de cada classe. A Figura 42 possui exemplos de erros de anotação para cada classe analisada. Na Figura 43 pode-se visualizar exemplos de erros do modelo (onde tem-se certeza de que a anotação esta correta).

Tabela 10 – Analise de erro

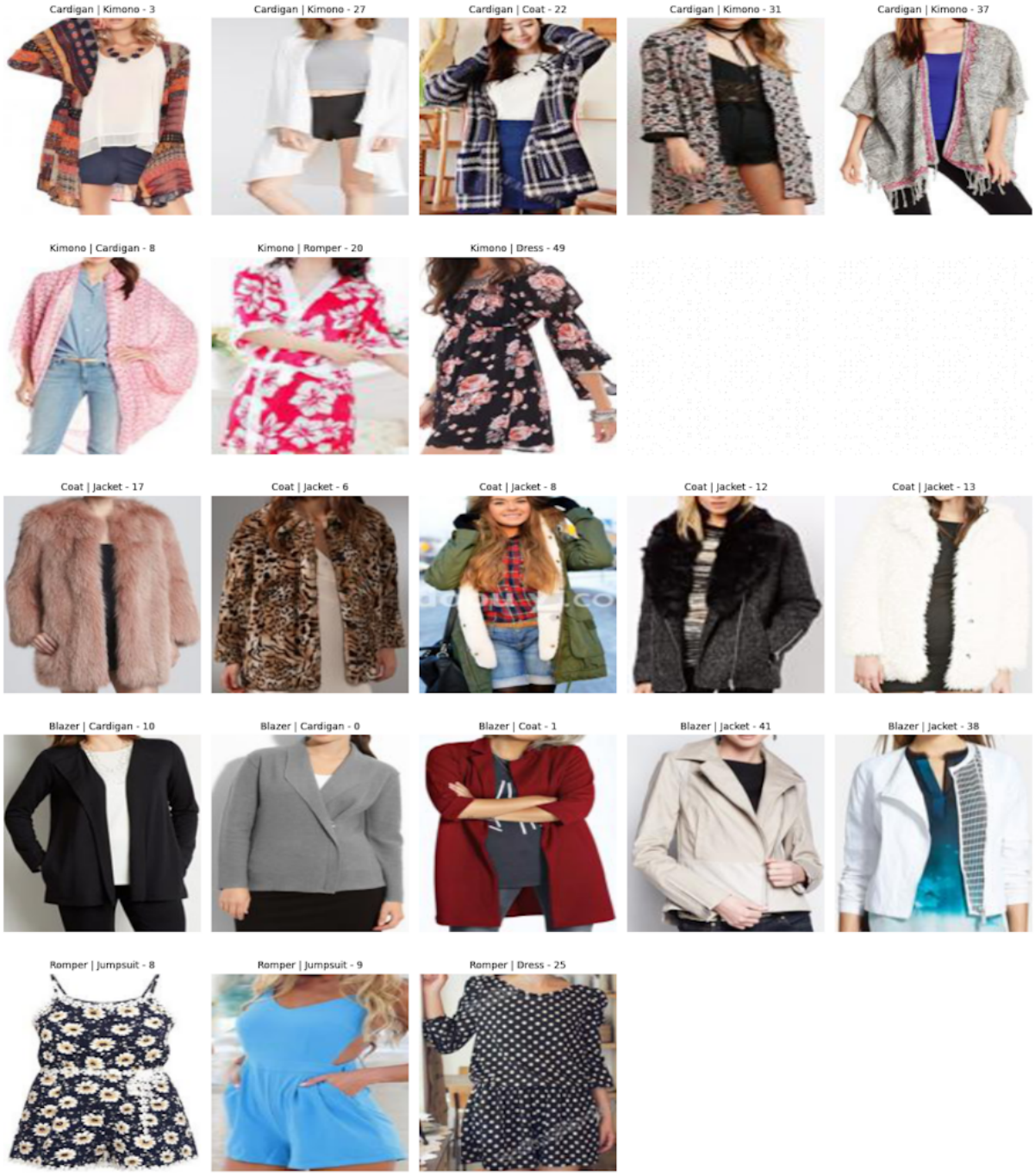
Classe	**Incorretas**	**Erros do modelo**	**Erros de anotação**	**Indeterminadas**
Cardigan	**69**	9 (13.04%)	35 (50.72%)	25 (36.23%)
Kimono	56	3 (05.35%)	36 (64.28%)	17 (30.35%)
Coat	53	5 (09.43%)	26 (49.05%)	22 (41.50%)
Blazer	52	7 **(13.46%)**	11 (21.15%)	34 **(65.38%)**
Romper	44	3 (06.81%)	29 **(65.90%)**	12 (27.27%)

Figura 42 – Exemplos de erros de anotação (Label | Predição) | ViT-B/16 (OpenAI)



Fonte: Autor

Figura 43 – Exemplos de erros do modelo (Label | Predição) | ViT-B/16 (OpenAI)



Fonte: Autor

5 CONCLUSÃO

Neste trabalho, foi realizada uma comparação entre diferentes arquiteturas de redes neurais convolucionais e o método CLIP para a tarefa de classificação de imagens de moda. Foram utilizados os conjuntos de dados Deep Fashion e Deep Fashion 2, que contêm alta diversidade de itens de moda. Os procedimentos realizados para obter-se um conjunto de dados adequado, contribuíram para o aumento da acurácia dos modelos. Constatou-se que o conjunto de dados de testes é ruidoso em relação aos rótulos de classe, porém, o modelo ViT-B/16 (OpenAI) foi capaz de extrair boas representações das classes, uma vez que encontrou-se um grande número de exemplos onde apesar da anotação incorreta, o modelo classificou corretamente a amostra. As analises de *aspect ratio*, resolução e confiança não demostram nenhuma inconformidade que possa ter prejudicado o aprendizado do modelo. Foram avaliadas as métricas de acurácia e sensibilidade. Os resultados mostraram que o método CLIP superou as arquiteturas pré-treinadas de forma convencionais em todas as métricas analisadas. O modelo CLIP também demonstrou um desempenho considerável de predição *zero-shot*, ou seja, de classificar imagens sem ter visto exemplos das classes de interesse. Isso indica que modelos CLIP podem aprender conceitos visuais ricos a partir de descrições em linguagem natural, sem depender de um conjunto fixo de classes, se tornando um ótimo ponto de partida para utilização de transferência de aprendizado para modelos de classificação de imagens de moda.

5.1 TRABALHOS FUTUROS

Como possíveis trabalhos futuros, sugere-se:

- Treinar outras arquiteturas pré-treinadas com o método CLIP e pré-treinadas no ImageNet, de forma a verificar se a superioridade do método CLIP se mantém para diferentes arquiteturas de modelos.
- Comparar o método CLIP com outros métodos do estado da arte para classificação de imagens.
- Realizar um estudo sobre o impacto dos *templates* no resultado do treinamento com modelos CLIP.
- Obter a ajuda de um especialista na área da moda para a rotulação e definição das classes de forma menos ambígua possível.

REFERÊNCIAS

AN, X. et al. Unicom: Universal and compact representation learning for image retrieval. *arXiv preprint arXiv:2304.05884*, 2023. Disponível em: <https://arxiv.org/abs/2304.05884>. Citado na página 18.

ANWAR, A. *What is Average Precision in Object Detection Localization Algorithms and how to calculate it?* 2022. Acesso em 12/12/2023. Disponível em: <https://aqeel-anwar.medium.com/>. Citado na página 34.

BAELDUNG. *Intersection Over Union for Object Detection*. 2023. Acesso em: 18/12/2023. Disponível em: <https://www.baeldung.com/cs/object-detection-intersection-vs-union>. Citado na página 35.

CHAGAS, E. T. D. O. Deep learning e suas aplicações na atualidade. *In: Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano, v. 4*, p. 05–26, 2019. Disponível em: <https://www.nucleodoconhecimento.com.br/administracao/deep-learning>. Citado na página 20.

CHENG, W.-H. et al. Fashion meets computer vision: A survey. *In: Association for Computing Machinery*, New York, NY, USA, v. 54, n. 4, jul 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3447239>. Citado na página 17.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. *In: 2009 IEEE conference on computer vision and pattern recognition. Ieee*, p. 248–255, 2009. Disponível em: <https://ieeexplore.ieee.org/document/5206848>. Citado 3 vezes nas páginas 17, 22 e 23.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. Disponível em: <https://arxiv.org/abs/2010.11929>. Citado 3 vezes nas páginas 17, 29 e 30.

GE, Y. et al. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, p. 5337–5345, 2019. Disponível em: <https://arxiv.org/abs/1901.07973>. Citado 2 vezes nas páginas 42 e 43.

GUSTINELI, M. A survey on recently proposed activation functions for deep learning. *arXiv preprint arXiv:2204.02921*, 2022. Disponível em: <https://arxiv.org/abs/2204.02921>. Citado 2 vezes nas páginas 23 e 24.

HE, K. et al. Deep residual learning for image recognition. *In: Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770–778, 2016. Disponível em: <https://arxiv.org/abs/1512.03385>. Citado 3 vezes nas páginas 17, 25 e 26.

HOSSIN, M.; SULAIMAN, M. A review on evaluation metrics for data classification evaluations. *In: International journal of data mining knowledge management process, v. 5, n. 2*, p. 1, 2015. Disponível em: <https://www.researchgate.net/publication/275224157_A_Review_on_Evaluation_Metrics_for_Data_Classification_Evaluations>. Citado na página 30.

JAIN, V.; WAH, C. Computer vision in fashion trend analysis and applications. *In: Journal of Student Research, v. 11, n. 1*, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:251520637>. Citado na página 22.

JAISWAL, A. et al. A survey on contrastive self-supervised learning. *In: Technologies, v. 9, n. 1*, p. 2, 2021. ISSN 2227-7080. Disponível em: <https://www.mdpi.com/2227-7080/9/1/2>. Citado na página 22.

KUKIL. *Intersection over Union (IoU) in Object Detection & Segmentation*. 2022. Acesso em: 16/12/2023. Disponível em: <https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/>. Citado na página 33.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *In: nature, v. 521, n. 7553*, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Citado 2 vezes nas páginas 17 e 24.

LI, Z. et al. A survey of convolutional neural networks: Analysis, applications, and prospects. *In: IEEE transactions on neural networks and learning systems*, 2021. Disponível em: <https://arxiv.org/abs/2004.02806>. Citado na página 24.

LIANG, J. et al. Two-terminal fault location method of distribution network based on adaptive convolution neural network. *In: IEEE Access, v. 8*, v. 8, p. 54035–54043, 2020. Disponível em: <https://ieeexplore.ieee.org/document/9035471>. Citado na página 25.

LIU, Z. et al. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. *In: Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 1096–1104, June 2016. Disponível em: <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Liu_DeepFashion_Powering_Robust_CVPR_2016_paper.pdf>. Citado na página 42.

RADFORD, A. et al. Learning transferable visual models from natural language supervision. *In: International conference on machine learning*, p. 8748–8763, 2021. Disponível em: <https://arxiv.org/abs/2103.00020>. Citado 8 vezes nas páginas 8, 9, 18, 36, 37, 38, 39 e 40.

REDMON, J. et al. You only look once: Unified, real-time object detection. *In: Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 779–788, 2016. Disponível em: <https://arxiv.org/abs/1506.02640>. Citado 2 vezes nas páginas 28 e 29.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *In: International journal of computer vision, v. 115*, p. 211–252, 2015. Disponível em: <https://arxiv.org/abs/1409.0575>. Citado na página 23.

SANTANA, M. Deep learning: do conceito às aplicações. 2018. Acesso em: 06/11/2023. Disponível em: <https://medium.com/data-hackers/deep-learning-do-conceito-%C3%A0s-aplica%C3%A7%C3%B5es-e8e91a7c7eaf>. Citado 2 vezes nas páginas 20 e 21.

SWAPNA K, E. *Convolution Neural Networks*. 2020. Acesso em: 11/12/2023. Disponível em: <https://developersbreach.com/convolution-neural-network-deep-learning>. Citado na página 24.

TAN, M.; LE, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *In: International conference on machine learning. PMLR*, p. 6105–6114, 2019. Disponível em: <https://arxiv.org/abs/1905.11946>. Citado 3 vezes nas páginas 17, 26 e 27.

TORREY, L.; SHAVLIK, J. Transfer learning. *In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global*, p. 242–264, 2010. Citado na página 27.

VASWANI, A. et al. Attention is all you need. *In: Advances in neural information processing systems, v. 30*, 2017. Disponível em: <https://arxiv.org/abs/1706.03762>. Citado 2 vezes nas páginas 29 e 30.

WEISS, K.; KHOSHGOFTAAR TAGHI, M.; WANG, D. A survey of transfer learning. *In: Journal of Big data, v. 3, n. 1*, p. 1–40, 2016. Disponível em: <https://doi.org/10.1186/s40537-016-0043-6>. Citado na página 27.

WORTSMAN, M. et al. Robust fine-tuning of zero-shot models. *In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 7959–7971, 2022. Disponível em: <https://arxiv.org/abs/2109.01903>. Citado na página 53.

YOSINSKI, J. et al. How transferable are features in deep neural networks? *In: Advances in neural information processing systems, v. 27*, 2014. Disponível em: <https://arxiv.org/abs/1411.1792>. Citado na página 27.

ZOU, Z. et al. Object detection in 20 years: A survey. *In: Proceedings of the IEEE*, 2023. Disponível em: <https://arxiv.org/abs/1905.05055>. Citado na página 33.