Juliana Heluy do Prado

Uso de Visão Computacional para reconhecer marcas de maquiagem e tipos de pele de Fitzpatrick, de acordo com a escala Taylor Hyperpigmentation.

PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de Graduação em Ciência da Computação

Rio de Janeiro
Junho de 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Juliana Heluy do Prado

Uso de Visão Computacional para reconhecer marcas de maquiagem e tipos de pele de Fitzpatrick, de acordo com a escala Taylor Hyperpigmentation.

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao Programa de Ciência da Computação, do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do titulo de Bacharel em Ciência da Computação.

Orientador : Prof. Marcelo Gattass
Co-orientador: Luiz Fernando Trindade Santos

Rio de Janeiro
Junho de 2022

Juliana Heluy do Prado

Ficha Catalográfica

Heluy do Prado, Juliana

Uso de Visão Computacional para reconhecer marcas de maquiagem e tipos de pele de Fitzpatrick, de acordo com a escala Taylor Hyperpigmentation. / Juliana Heluy do Prado; orientador: Marcelo Gattass; co-orientador: Luiz Fernando Trindade Santos. – 2022.

44 f: il. color. ; 30 cm

Projeto Final - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2022.

Inclui bibliografia

1. Informática – Teses. 2. Visão Computacional. 3. Indústria da Beleza. 4. Reconhecimento de Tom de Pele. 5. Reconhecimento de Logomarcas. 6. YOLOv5. 7. Rede Neural Convolucional. 8. Detecção de Objetos. 9. Detecção de rosto. 10. Roboflow. 11. K-Means. I. Gattass, Marcelo. II. Fernando Trindade Santos, Luiz. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Agradecimentos

Ao meu orientador Professor Marcelo Gattass e meu co-orientador Luiz Fernando Trindade Santos por estarem sempre disponíveis e me orientando da melhor forma possível.

Ao Marcelo Taranto que me apoiou e salvou meu futuro, a Fátima Amaral que é como uma mãe pra mim e ao meu tio Fred que é o melhor professor de matemática que eu já tive.

Ao meu namorado, que é o meu porto seguro. Ao meu irmão Victor que me fez gostar de computadores, as minhas amigas que são minha família e aos meus amigos da faculdade que tive o prazer de conhecer.

Dedico este projeto ao meu pai que me ensinou a ser quem eu sou.

Abstract

Heluy do Prado, Juliana; Gattass, Marcelo (Advisor); Fernando Trindade Santos, Luiz (Co-Advisor). **Using Computer Vision to recognize makeup brands and types of Fitzpatrick skin, according to the Taylor Hyperpigmentation scale.**. Rio de Janeiro, 2022. 44p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The beauty industry is an ever growing market, and it isn't as explored by technology as it could be when it comes to the company-consumer relationship. Currently the biggest means of communication is YouTube, with influencers reviewing every product. However, the search for personalized content is still scarce. Considering this, the objective of this Final Project is to train the object detection computer vision model, YOLOv5, to recognize logos. Roboflow will also be used to annotate and split the database. Also, considering the scope of the project, a Convolutional Neural Network will be implemented to detect the face and recognize the skin using K-Means. Also, the Fitzpatrick skin type will be defined, according to the Taylor Hyperpigmentation scale.

Keywords

Computer Vision; Beauty Industry; Skin Color Recognition; Logo Recognition; YOLOv5; Convolutional Neural Network; Object Detection; Face Detection; Roboflow; K-Means.

Resumo

Heluy do Prado, Juliana; Gattass, Marcelo; Fernando Trindade Santos, Luiz. **Uso de Visão Computacional para reconhecer marcas de maquiagem e tipos de pele de Fitzpatrick, de acordo com a escala Taylor Hyperpigmentation.**. Rio de Janeiro, 2022. 44p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A indústria da beleza é um dos mercados mais quentes do momento, e é muito pouco explorado pela tecnologia quando se trata da relação empresa-consumidor. Atualmente o maior meio de comunicação é o YouTube com vídeos de reviews feitos por influenciadores. Porém a busca por conteúdo personalizado ainda é escasso. Considerando isso, o objetivo desse Projeto Final é treinar o modelo de visão computacional de detecção de objetos, YOLOv5, para reconhecer logomarcas. Será utilizado também, o Roboflow para anotar e dividir o banco de dados. Também, considerando o escopo do projeto, será implementada uma Rede Neural Convolucional que faça detecção de rosto, e reconheça a pele utilizando K-Means. E também, o tipo de pele Fitzpatrick será definido, de acordo com a escala de Taylor Hyperpigmentation.

Palavras-chave

Visão Computacional; Indústria da Beleza; Reconhecimento de Tom de Pele; Reconhecimento de Logomarcas; YOLOv5; Rede Neural Convolucional; Detecção de Objetos; Detecção de rosto; Roboflow; K-Means.

Sumário

Lista de figuras

Lista de Códigos

Lista de Abreviaturas

YOLO – You Only Look Once

CNN – Convolutional Neural Network

CBL – Challenge Based Learning

MTCNN – Multi-task Cascaded Convolutional Networks

RGB – Red, Green and Blue

IoU – Intersection Over Union

AP – Average Precision

TP – True Positives

FN – False Negatives

mAP – mean Average Precision

PR – Precision Recall

1
Introdução

A indústria da beleza sempre existiu. Ela esteve mencionada, por exemplo, em obras como Hamlet, ordenando o rosto da Ophelia, numa época em que a maquiagem ainda era vista como um instrumento para mulheres consideradas vulgares na cultura Europeia. Assim como Egito antigo, com o delineado ao redor dos olhos dos mais afortunados. Essa indústria, além de antiga, se alimenta das relações que os seres humanos têm com a beleza e a vaidade. Não por acaso ela é uma das maiores indústrias do mundo, avaliada em 511 bilhões de dólares em 2021, e projetada a valer 805 bilhões em 2023(9) de acordo com a pesquisa Global Cosmetics Products Market(10). E o Brasil é o quarto maior mercado global de beleza(4). Apesar das vendas terem sido afetadas durante a pandemia de 2020 e 2021, essa projeção continua sendo considerada devido à constatação de que este mercado possui um grande espaço a ser explorado: o espaço digital(9, 10).

Os consumidores da indústria da beleza sempre usaram o meio digital - mais especificamente o YouTube, que teve, por exemplo, 169 bilhões de visualizações de vídeos de beleza em 2018(11)- e o Instagram, utilizado em pesquisas sobre seus produtos de interesse(12). Desta forma, os consumidores conseguem ver os efeitos imediatos dos produtos em peles sem filtros estéticos ou edições de imagem. Graças à quantidade considerável de mais de 88 bilhões de vídeos de cosméticos disponíveis no YouTube(12), o consumidor pode optar por ouvir uma pessoa que tem o mesmo tom de pele que ele. Porém, é importante ressaltar que pessoas com tons de pele mais escuros sentem dificuldade em achar vídeos de influenciadores com o mesmo tom de pele(13).

Como se não bastasse, os consumidores digitais de beleza(14) procuram por *reviews* de criadores de conteúdo com menos seguidores, por acreditarem que eles seriam mais honestos e menos propensos – ou comprometidos – com a

sugestão de produtos patrocinados(15). Pela natureza da plataforma Youtube, os consumidores conseguem conversar e tirar dúvidas com micro criadores, mas não com criadores de conteúdo com maior número de seguidores. Porém, menos de 25% os vídeos de maquiagem do YouTube usam o nome da marca(12)porque os criadores consideram isso uma propaganda gratuita. Justamente por isso as grandes empresas, tais como a L´ORÉAL, Unilever e Estée Lauder, entre outras, possuem dificuldade para explorar esse mercado(15), pois vender produtos em capas de revista com modelos super produzidas e completamente editadas é uma estratégia ultrapassada.

Algumas empresas desse setor notaram a lacuna entre elas mesmas e seus consumidores, e optaram por pagar influencers com nomes conhecidos no mundo digital para fazerem propaganda, tais como Madison Beer, patrocinada pela Morphe(16) e a Kim Kardashian, patrocinada por diversas marcas(17), entre várias outras. Mas os consumidores já demonstraram notar quando os criadores de conteúdo estão sendo pagos por essas grandes empresas para vender um produto. Em uma pesquisa realizada pela Vettes, 58% de 520 mulheres entrevistadas relataram que não comprariam, ou estariam inseguras em comprar, um produto que foi pago por uma empresa para influenciadores divulgarem(15). Em suma, os consumidores decidiram ouvir influenciadores ao invés de propaganda(15).

Afinal, a uma forma de saber se o produto ficaria bom no consumidor seria buscando o mesmo produto em uma pessoa parecida. Atualmente, 90% dos consumidores buscam por autenticidade(18) antes de decidirem qual marca comprar. Por isso eles buscam por conteúdo criado sem segundas intenções, tais como vídeos sem patrocínio. Isso, entretanto, cria um dilema pois, um criador de conteúdo se estabelece e fica conhecido falando a verdade sobre produtos. Mas ele prefere não mencionar o nome da marca no seu vídeo justamente por não ser patrocinado, tornando-o difícil de achar.

Porém, escondido nesse dilema se encontra uma oportunidade: a opção

do criador de conteúdo lucrar com qualquer compra feita por causa deles, e ainda manter a liberdade de falar bem – ou mal – de qualquer marca em seus vídeos, usando tecnologia do estado da arte da Visão Computacional e Machine Learning para detectar os produtos presentes, e permitindo que o criador de conteúdo ganhe um percentual sobre qualquer compra feita com seus *reviews*. As duas partes, entretanto, estão perdendo oportunidades: tanto os micro influenciadores quanto as empresas de cosméticos.

Considerando todos esses pontos, o objetivo desse Projeto Final é implementar um algoritmo de Visão Computacional que, dado um vídeo feito por um criador de conteúdo, ele reconheça as marcas dos produtos exibidos pelo apresentador, e também o seu tipo de pele Fitzpatrick(19) de acordo com a escala de Taylor Hyperpigmentation(20).

O reconhecimento do tipo de pele Fitzpatrick ajuda a categorizar os vídeos baseados no tom de pele do criador de conteúdo, já que pessoas com tons de pele mais escuras sentem dificuldade em achar vídeos de criadores de conteúdo com o mesmo tom de pele que eles próprios(13), e o reconhecimento das marcas dos produtos utilizados nos vídeos garante maior facilidade aos consumidores a acharem *reviews* honestos dos produtos que eles procuram, além de permitir que o influenciador possa ganhar alguma compensação pelos seus vídeos.

Esse documento está estruturado da seguinte forma: no Capítulo 2 nós apresentamos a visão geral do contexto deste projeto, e como está a situação atual do uso da tecnologia na indústria da beleza, assim como trabalhos análogos. No Capítulo 3, explicamos em ordem cronológica o desenvolvimento do projeto. No Capítulo 4 apresentamos os resultados obtidos, e por fim, no capítulo 5 concluímos o projeto apresentando melhorias para o futuro.

2
Visão Geral

Para que o projeto estivesse com objetivos claros e concisos, o framework do CBL(Challenge Based Learning) (21) foi aplicado. Esse framework, criado pela Apple (21), é excelente para afunilar a proposta até que todos os seus aspectos iniciais estejam coerentes e ordenados. Inicialmente, uma Big Idea é definida, tornado claro a que nicho o projeto pertence.

Na sequência definimos a Essential Question: ela serve para afunilar mais um pouco, dentro do nicho decidido anteriormente, qual problema será tratado. Por fim, o Challenge é apresentado dando nome ao desafio que será abordado.

Observando a figura 2.1 , vemos que ficou definido como:

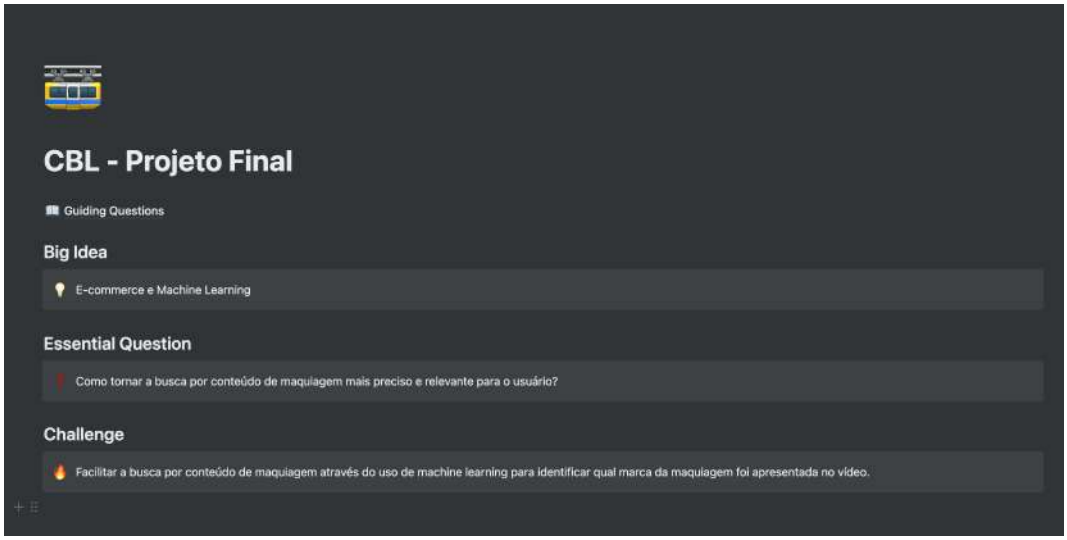


Figura 2.1: Challenge Based Learning feito através da ferramenta de edição; Notion(1)

Big Idea : E-commerce e Machine Learning e Visão Computacional.

Essential Question : Como tornar a busca por conteúdo de maquiagem mais preciso e relevante para o usuário?

Challenge : Facilitar a busca por conteúdo de maquiagem através do uso de machine learning e técnicas de visão computacional para identificar qual marca de maquiagem foi apresentada no vídeo.

Estando com a ideia bem definida, era preciso buscar por projetos similares e análogos para entender o que a proposta oferece de novo, que não exista ainda no mercado. Um checklist foi feito com todas as características que o projeto oferecerá e, na medida que cada produto análogo foi estudado, os checklists foram ticados de acordo com o que eles proporcionam ao usuário. Esse checklist foi feito no Notion (1) e pode ser visto na figura 2.2.

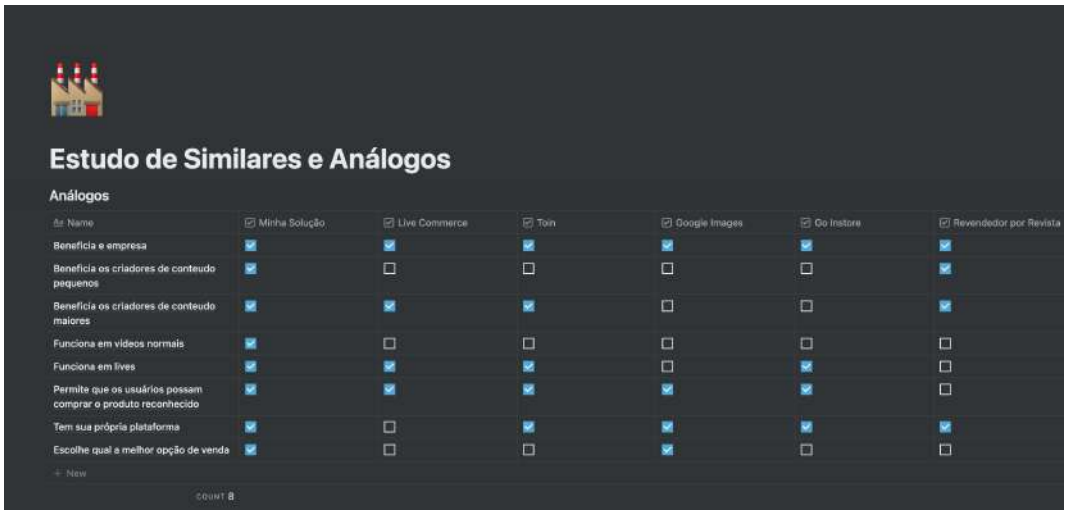


Figura 2.2: Estudo de Similares e Análogos feito através da ferramenta de edição; Notion (1).

Como pode ser observado, há poucos produtos similares ou análogos à proposta sugerida. Seguem abaixo algumas das soluções estudadas, que trazem elementos parecidos com o que foi sugerido, mas não englobam toda a proposta:

Criadores de Conteúdo - Alguns criadores de conteúdo colocam os produtos utilizados e os minutos que eles aparecem nas descrições dos vídeos, mas esses criadores costumam ser patrocinados pelas marcas para fazer o mesmo.

Live Commerce - Live commerce está em alta, vai crescer ainda mais, mudando a forma como o e-commerce opera atualmente (22). A L´ORÉAL é uma das marcas que atualmente está investindo em live commerce. Por exemplo: houve um evento no Brasil, no dia 23 de Setembro de 2021, que oferecia 50% de desconto nos

produtos que fossem apresentados durante a live (23). Essas lives não usam softwares para detectar quais elementos estão presentes, mas costumam ser programados com antecedência, com uma pessoa chave hosteando o evento, disponível simultaneamente, com alguma plataforma de e-commerce aberta com os produtos que foram pré-selecionados (24) listados e colocados disponíveis para compra.

Google Images - Uma forma de pesquisar o produto usando uma foto. Os resultados retornados são sites de compra que possuem pouca informação sobre como o produto se comporta na pele, além de ser pouco personalizado para o cliente.

Filtros - Apps que permitem que o usuário veja como a maquiagem ficaria na pele utilizando filtros de realidade aumentada.

- O Aplicativo da Sephora (4) chamado Virtual Artist (25) atualmente não está disponível no Brasil, e utiliza filtros para simular a maquiagem no rosto do usuário.
- Um outro aplicativo parecido é o da L´ORÉAL chamado Virtual Try On(26) que tem a mesma funcionalidade. Os dois aplicativos não são mais avançados que os filtros de maquiagem do Instagram, TikTok e Snapchat (27) mas o Virtual Try On já foi muito bem aceito na China com milhões de downloads após o lançamento (28).

Impressora de batom - A marca Yves Saint Laurent desenvolveu uma impressora (29) capaz de criar 4 mil tonalidades de batons diferentes. A impressora é controlada por um aplicativo que permite imprimir cores pre-selecionadas, cores escolhidas pelo usuário através de um espectro de cores, e também utilizar fotos *uploaded* pelo usuário e recriando as cores da foto.

Os produtos listados não apresentam todos os pontos que a solução deste Projeto oferece, já que nenhuma delas se aproveita da fonte mais procurada pelos consumidores quando se trata de avaliações e respostas honestas sobre produtos de beleza. Os aplicativos da Sephora e L´ORÉAL, ambos são reconhecidos como propaganda pelos próprios usuários, e utilizam tecnologias existentes de pouca inovação, já usadas em filtros de redes sociais conhecidas (27). Além disso, o filtro não mostra como a maquiagem ficaria realmente na pele, pois não tem o elemento de oxidação, sudorese, e textura que uma pele real possui.

Já o Live commerce, apesar de promissor, cai na mesma armadilha que os aplicativos das grandes empresas, pois uma Live precisa de uma personalidade conhecida para *hostear* o evento, caso contrário não chamaria ao público. Isso torna a Live apenas numa grande propaganda patrocinada pela empresa por trás do evento. O mesmo se aplica aos links incluídos nas descrições dos vídeos com os nomes dos produtos usados, pois esses links são reconhecidos como patrocínio pelos consumidores.

Da mesma forma, o Google Image é uma ferramenta que reconhece o produto, lista produtos similares visualmente e onde comprá-los, mas essa alternativa não apresenta como o produto ficaria no rosto - ele é apenas uma lista de preços.

Quanto à impressora de batom da Yves Saint Laurent, esse produto é o mais próximo ao projeto. Apesar de ser um produto físico, ele reconhece uma cor através de uma imagem e imprime um batom. Porém, ele não endereça os outros pontos deste projeto, tais como ver através de vídeos como outros produtos (base, sombra, primer etc.) de várias marcas, não só os da Yves Saint Laurent, ficariam no usuário; oferecer recompensa aos influenciadores pelos seus reviews, categorizar maquiagens que ficariam bem no tom de pele do usuário etc. Esse produto é apenas uma maquiagem, e não exatamente uma solução.

É possível, portanto, observar que o maior avanço tecnológico nesta área utiliza técnicas gastas e saturadas, como é o caso dos aplicativos Virtual Artist e Virtual Try On. As outras alternativas não englobam tudo que este Projeto oferece, apesar de inovadoras.

3
Metodologia

A partir do estudo realizado, foi possível dar início ao desenvolvimento do projeto. Como é preciso separar em duas partes, optei por dividir cronologicamente da seguinte maneira; o reconhecimento de logomarca seria o primeiro desafio a ser enfrentado, e, em seguida, o reconhecimento de tons de pele. Portanto, este capítulo está dividido da mesma maneira como descrito acima.

3.1
Reconhecimento de Logomarcas

Para detectar logomarcas é preciso criar um algoritmo capaz de detectar e classificar as logos dos produtos que aparecem em vídeos de criadores de conteúdo. Dessa forma será possível, posteriormente, classificar e filtrar os vídeos de acordo com o que esteja presente dentro deles.

Durante a pesquisa sobre artigos científicos que estudavam o melhor algoritmo para detectar objetos, dois artigos se destacaram e foram usados como base para o desenvolvimento desta parte do projeto: Deep Learning for Logo Recognition (30) e You Only Look Once: Unified, Real-Time Object Detection (31).

O artigo You Only Look Once: Unified, Real-Time Object Detection (31) estuda o algoritmo denominado YOLO para detecção de objetos. Um detector de objetos é desenhado para criar características de imagens de entrada, e assim, alimentar essas características em um sistema de predição que desenhe caixas ao redor desse objeto, que, por sua vez, terá a classe à qual ele pertence detectada. O modelo YOLO foi o primeiro detector de objeto a conectar o procedimento de prever caixas delimitantes com classificações em uma *differentiable network* de ponta a ponta. E, também, o YOLO treina com imagens inteiras e otimiza a eficácia de detecção.

A rede YOLO consiste em três peças-chaves:

Coluna : Uma CNN que reúne e forma características de imagens em granularidades diferentes;

Pescoço : Uma série de camadas que misturam e combinam características de imagens para entregar ao próximo passo que é a predição; e

Cabeça : Se alimenta das características entregue pelo pescoço e inicia os passos para classificar e encaixar objetos.

O modelo YOLO tem mais de cinco versões, porém o YOLOv5 foi selecionado para este projeto já que, como pode ser visto na figura 3.1, ele é mais rápido, eficaz, e fácil de usar justamente por este modelo estar na linguagem Python. Por sua vez, o YOLOv5 possui uma comunidade mais ativa e disposta a tirar dúvidas em foruns na internet. Portanto, foi utilizado o estado da arte de Machine Learning em linguagem Python, e o algoritmo de detecção de objeto em tempo real YOLOv5 (2) (You Only Learn Once [(31)]).

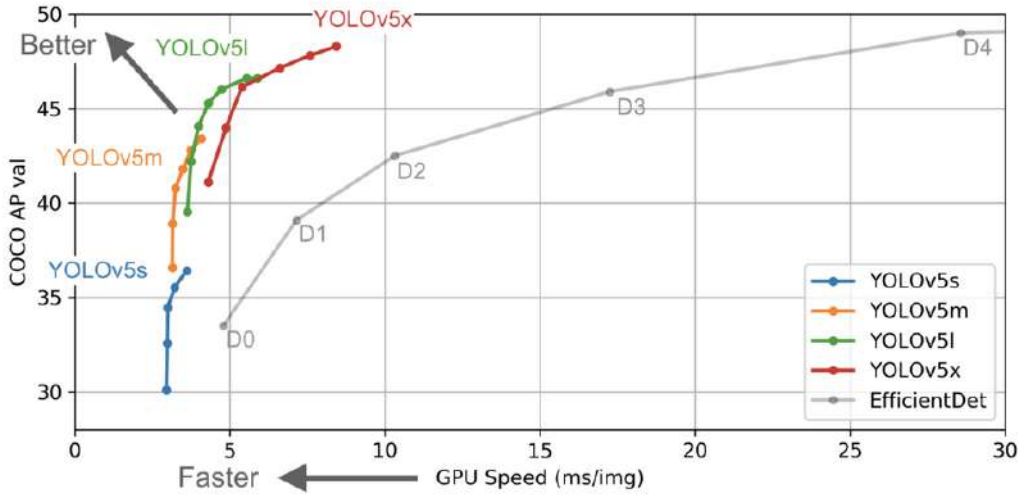


Figura 3.1: O lançamento inicial do YOLOv5 mostra um avanço no estado da arte da detecção de objeto (2).

No artigo Deep Learning for Logo Recognition (30), alguns bancos de dados foram nomeados. Foi a partir deles que seguimos com a busca por um banco de dados que melhor atenda às demandas do projeto.

Os maiores bancos de dados *open source* de logos são atualmente relacionados a marcas em geral, cobrindo até algumas marcas que produzem cosméticos, como é o caso do BelgaLogos (32), a WebLogo-2M (33) que até agora foi a que teve a maior quantidade de marcas de maquiagem contendo imagens classificadas e não classificadas de mais de 5 marcas relevantes ao projeto, e a QMUL-OpenLogo (34) com uma quantidade menor de marcas relevantes.

Como foi discutido acima e pode ser visto na figura 3.2, o banco de dados que melhor atende às necessidades deste projeto é o WebLogo-2M (33), e portanto, foi com ele que este projeto deu continuidade.

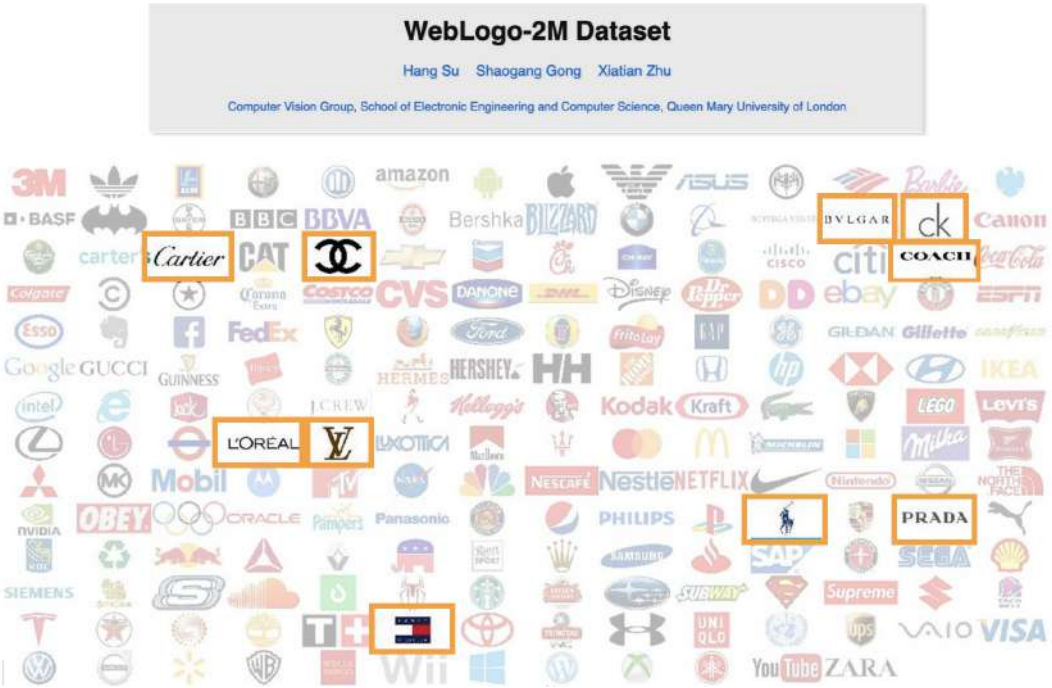


Figura 3.2: As marcas em amarelo produzem produtos de beleza e são relevantes a esse projeto.

A pasta *Evaluation/Test Dataset* foi baixada do banco de dados gratuito do WebLogo-2M e - através de um algoritmo feito em Python - foi possível selecionar onze marcas, totalizando em 321 imagens baixadas. As logos selecionadas foram: Bvlgari, Calvin Klein, Cartier, Chanel, Coach, Hermès, L'Oréal, Polo Ralph Lauren, Prada e Tommy Hilfiger. Essas são as marcas disponíveis no banco de dados WebLogo-2M que produzem cosméticos, e possuem anotações.

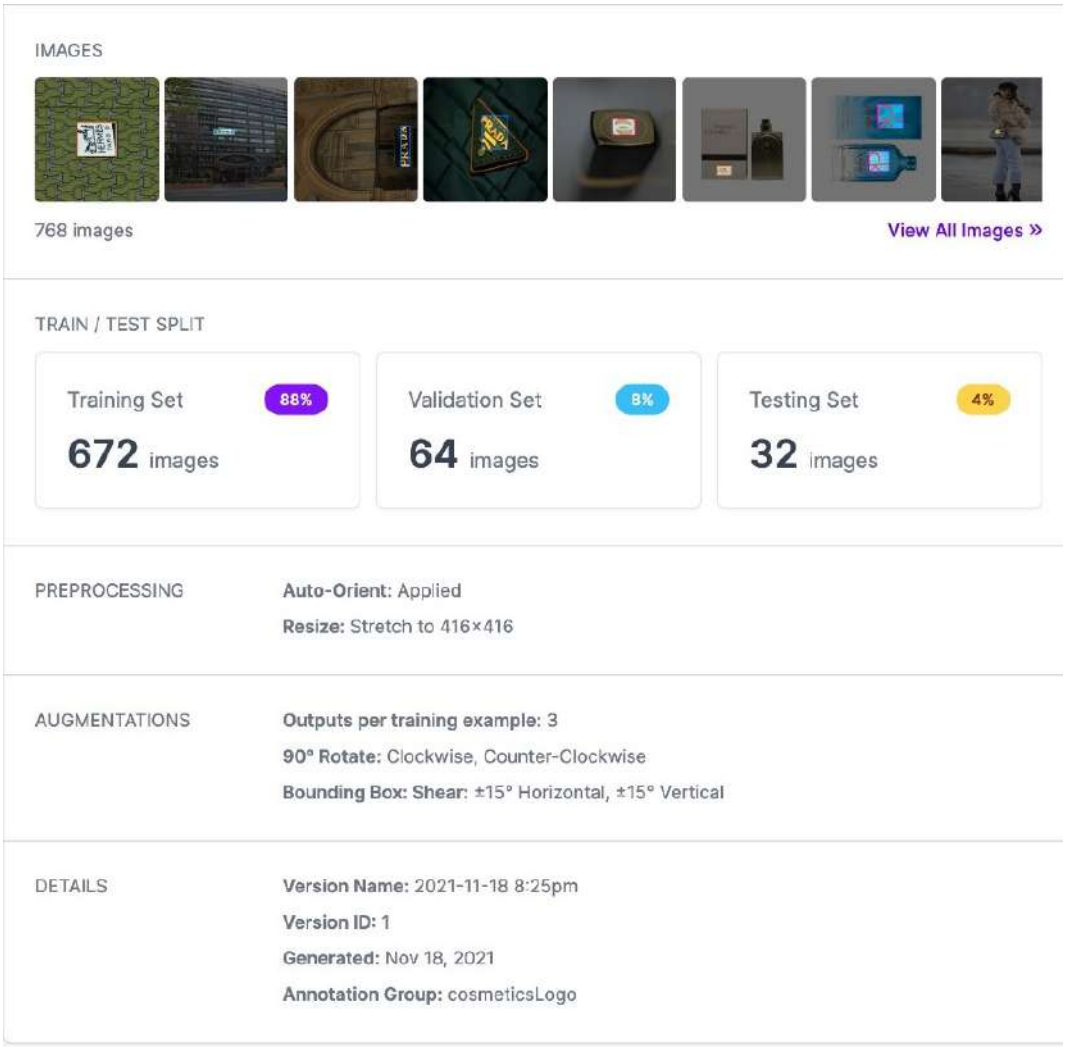


Figura 3.3: Resumo do pré-processamento das imagens do banco de dados.

A plataforma Roboflow (30) é uma ferramenta que permite que desenvolvedores consigam processar as imagens do banco de dados para utilizar em modelos personalizados. Ela foi utilizada neste projeto por ser uma ferramenta intuitiva, e que possibilitaria pré-processar o banco de dados automaticamente, incluindo a possibilidade de fazer anotações. As anotações denotam a qual classe determinada imagem pertence, para que, por sua vez, o modelo consiga aprender a classificar as imagens automaticamente.

Outro passo importante é denotar os *bounding boxes* nas imagens, permitindo que o modelo consiga aprender a fazer o mesmo. Todo esse processo, se feito manualmente, pode levar muito tempo, assim como ter erros humanos que prejudicariam o projeto. Esta plataforma também permite que o modelo

de dados seja facilmente importado dentro do Google Colab, e também divide as imagens em pastas para treinar, validar e testar.

Portanto, como mostra a figura 3.3, as imagens foram:

- pré-processadas;
- bounding boxes foram criadas;
- todas as imagens foram diminuídas para o tamanho de 416:416 pixels que é o tamanho aceito pelo YOLOv5;
- Rotação de 90 para a direita e para a esquerda; Sheer no bounding box em copias de algumas imagens para evitar *overfitting*.

Portanto, ao adicionar as *augmentations*, aumentar artificialmente a quantidade de dados gerando novos pontos de dados a partir de dados existentes, e ao todo, 768 imagens foram utilizadas. Assim, o próprio Roboflow dividiu as imagens entre *Imagens de treino* (88%), *imagens de teste* (64%) e *imagens de validação* (32%), gerando um código para acessar esse banco de dados e totalizando em 768 imagens.

As imagens foram então usadas para treinar o modelo de YOLOv5. Como as imagens já estavam com anotações do Roboflow, o modelo utiliza o nome da marca para classificar a logomarca detectada. Foi então definido os parâmetros de execução do modelo em Python:

Código 1: Parâmetros de Execução

```
1 !python train.py --img 416 --batch 16 --epochs 649 --data {dataset.
      location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights
       '' --name yolov5s_results  --cache
```

1. --img: Tamanho das imagens de entrada.
2. --batch: Determina o tamanho do lote de treino.
3. --epochs: Epoch é uma passada completa das imagens de treinamento pelo algoritmo.

4. --data: Caminho para arquivo .yaml.

5. --cfg: Configurações do modelo.

6. --weights: Caminho para arquivo dos pesos de treino do modelo.

7. --nosave: Salvar apenas o último checkpoint.

8. --cache: Cache as imagens para treinar mais rapidamente.

O tamanho das imagens era de 416x416 como mencionado anteriormente. O *batch* foi definido com 16, pois é o número recomendado pela YOLOv5. Vale à pena ressaltar que o modelo foi treinado com 649 *epochs* pois o Google Colab possui um limite de tempo de execução, e após tentativa e erro, este número foi considerado o limite antes que o Google Colab fosse travado por inatividade. Por fim, os pesos utilizados foram do próprio YOLOv5.

Os primeiros resultados obtidos foram satisfatórios, como poderá ser visto mais a fundo no capítulo 4. Exatamente por ter escolhido marcas que já produziam cosméticos, algumas imagens dos resultados eram de propagandas de produtos de beleza. Isto era um indicativo muito bom, pois o modelo estava detectando marcas em frascos e garrafas de produtos de beleza existentes.

Porém, essas imagens eram majoritariamente de propaganda, o que facilita a detecção uma vez que a iluminação e o posicionamento dos produtos estavam da melhor forma para serem vistos facilmente. Porém, como o objetivo inicial do projeto era detectar logos em vídeos de influenciadores que mostram os produtos para a câmera de formas não exatamente ideias, o modelo foi rodado em cima de vídeos similares. Esses vídeos foram separados em imagens, separando cada frame com o algoritmo em Python, e rodando o modelo em cima de cada uma das imagens.

Código 2: Separando Vídeos em Frames

```
1 import cv2
2 from google.colab import drive
3 drive.mount('/content/drive')
```

```
4 video_name = "/content/drive/My Drive/Youtube2.mp4" # or any other
      extension like .avi etc
5 vidcap = cv2.VideoCapture(video_name)
6 success,image = vidcap.read()
7 count = 0
8
9 while success:
10   cv2.imwrite("/content/sample_data/YouTube/frame%d.jpg" % count,
        image) # save frame as JPEG file
11   success,image = vidcap.read()
12   count += 1
```

O resultado foi igualmente satisfatório, e será visto em mais detalhes no capítulo 4.

3.2 Reconhecimento de Tons de Pele

O reconhecimento automático de tom de pele por via de uma imagem é um processo complexo, no qual é preciso achar pixels em uma dada região da imagem, que pertençam à pele do sujeito que se encontra na foto. Porém, para o olho humano, reconhecer a região da foto que contém um corpo humano, ou até parte dele, é automático. Na maioria das imagens é possível saber o que é pele, roupa ou cabelo de forma intuitiva. Já para um computador, essa tarefa é mais difícil: primeiro ele teria que saber o que é um humano para reconhecer e, em seguida, identificar o que foi pele para conseguir excluir o que não for pele (por exemplo roupa e cabelo ou acessórios), e depois extrair a cor entre vários pixels que melhor representariam a cor da pele dessa pessoa.

Portanto, foi preciso pesquisar para definir o processo mais eficaz de reconhecimento de tom de pele, e os artigos que foram usados como base para esta etapa do projeto foram os artigos cientificos Casual Conversations: A dataset for measuring fairness in AI (35) e Skin Cancer Classification Using K-Means Clustering (36).

O artigo Casual Conversations: A dataset for measuring fairness in AI (35) explora o dataset Casual Conversations criado para testar e aprimorar modelos de Inteligencia Artificiais para serem mais inclusivos e robustos. Apesar deste artigo não especificar um método ou um modelo para reconhecimento de tons de pele, ele foi esclarecedor sobre quão robusto o banco de dados deve ser para que o modelo deste projeto seja eficaz, e também que uma boa estratégia seria usar identificação do rosto para detectar o tom de pele.

Com base nisso, e a partir do principio de que dado um rosto, era preciso detectar o que na foto pode ser considerado pele para então extrair a cor, o artigo Skin Cancer Classification Using K-Means Clustering (36) foi iluminador. O artigo usa K-Means Clustering para detectar um melanoma em uma foto, e deu um ponto de partida para encontrar a implementação (37) do usuário do GitHub, NikolayTV, que se encaixa na proposta deste projeto e, a

partir dele, foi possível avaliar o que poderia ser feito.

Dando início à implementação, primeiro seria preciso detectar o rosto em uma foto. Portanto era preciso usar *bounding boxes* para detectar o rosto, como pode ser visto na figura 3.4.

Figura 3.4: Reconhecendo o rosto e formando *bounding boxes* ao seu redor.

Foi utilizado, então, a biblioteca MTCNN do Python(pip) desenvolvida pelo usuário do GitHub, ipacz (38), para detectar as coordenadas do canto superior esquerdo e canto inferior direito do rosto, como pode ser visto no código abaixo.

Código 3: Função que detecta as coordenadas do rosto em uma imagem

```
1  def get_face_coords_MTCNN(img):
2      detector = MTCNN()
3      results = detector.detect_faces(img)
4      if results != []:
5          b = results[0]['box']
6          x1 = int(b[0])
7          x1 = (x1 if x1>0 else 0)
8
9          y1 = int(b[1])
10         y1 = (y1 if y1>0 else 0)
11
12         x2 = int(b[0]) + int(b[2])
13         x2 = (x2 if x2<img.shape[1] else img.shape[1])
```

```
14
15          y2 = int(b[1]) + int(b[3])
16          y2 = (y2 if y2<img.shape[0] else img.shape[0])
17          return x1, y1, x2, y2
18      else:
19          return None, None, None, None
```

Com o rosto detectado, e trabalhando somente com o recorte do bounding box da imagem, era possível utilizar o K-Means para detectar a pele. Caso o rosto não esteja coberto por algo, a maior área do rosto de uma pessoa é a pele, portanto utilizamos essa lógica para seguir adiante utilizando K-Means Clustering Segmentation.

O K-Means Clustering Segmentation é um método de particionamento. Esse método agrupa objetos de forma que a variação dentro do grupo seja mínima. Desta forma, a imagem é segmentada com características de alto nível, ou seja, sem muitos detalhes. Este método funciona da seguinte maneira:

a. Inicialização de dois centros de classes randomicamente, esses centros representam centros de grupos iniciais.

b. Cálculo da distancia do histograma bin entre cada pixel da imagem e o centro da classe, e atribuição desses pixels ao centro da classe mais próxima.

c. Recalcular o novo posicionamento dos centros - isso é feito com a média dos valores do histograma bin do mesmo grupo.

d. Se o valor dos centros mudar, então repetir os passos **b.** e **c.**

Com as cores do rosto separadas em clusters de tons iguais, basta selecionar o centro do maior cluster e definir a cor deste centro como o tom de pele da pessoa. Para melhor entendimento, basta olhar a figura ilustrativa 3.5. Como foi mencionado acima, a razão pela qual o maior cluster foi escolhido foi em função do fato de que, em uma foto do rosto de uma pessoa, a maior área

é ocupada pela pele do resto. Portanto, os clusters das cores da boca, olhos, sobrancelhas etc., seriam bem menores comparado a área do rosto. Porém, como há variáveis que podem mudar esse resultado - como por exemplo uma barba - os três maiores clusters são detectados.

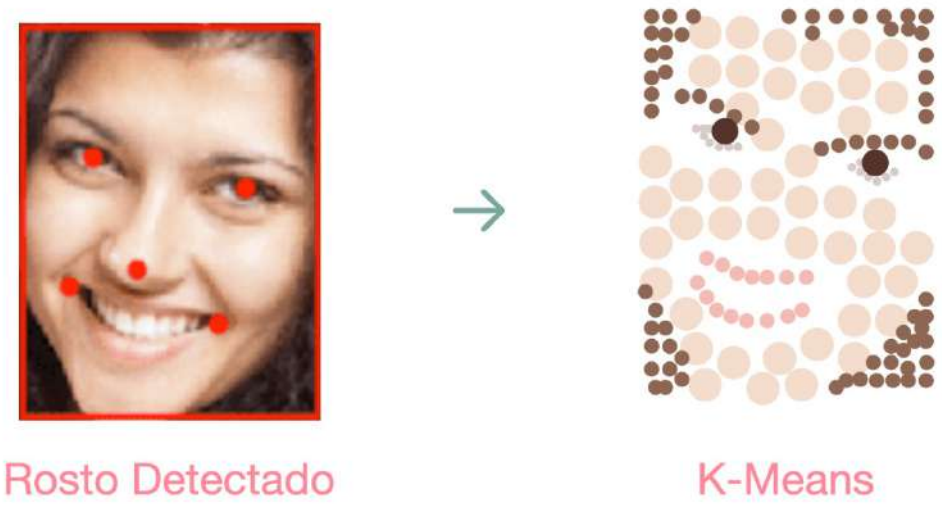


Figura 3.5: Figura ilustrativa da utilização de K-Means para clusterizar as cores do rosto.

Este método foi utilizado e testado em um banco de dados chamado UTKFace dataset disponível no Kaggle (39). Ele possui mais de 20 mil imagens de rostos com anotações sobre idade (variando de 0 a 116 anos), gênero e etnia.

O programa recebe uma imagem e retorna os tês tons principais em *grey scale*, assim como um vetor com o valor em RGB da cor principal em gray scale, e quatro imagens, a imagem do bounding box contendo o rosto da pessoa, e três quadrados contendo a cor em RGB do centro dos clusters prevalentes.

O programa também retorna uma imagem do bounding box contendo o rosto da pessoa em cor, três quadrados contendo a cor em RGB do centro dos clusters prevalentes, e seus respectivos vetores das cores em RGB. Segue a imagem 3.6, que é o retorno visual do programa de reconhecimento de tom de pele.

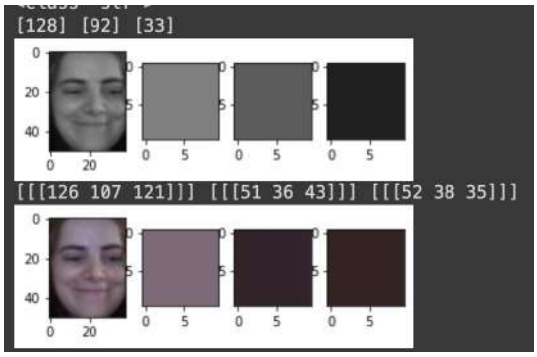


Figura 3.6: Retorno visual do programa de reconhecimento de tom de pele.

Os resultados com esse banco de dados foram satisfatórios mas não eram o suficiente para provar que o projeto estava funcionando, já que as imagens deste banco de dados vinham com o recorte do rosto pronto, e estavam muito bem iluminadas, então, elas não representavam um senário verdadeiro à proposta do projeto. Portanto, a partir deste ponto, o próximo passo era testar usando imagens de câmeras comuns.

No Google Colab, utilizando uma função própria disponível pelo sistema, foi possível acessar a webcam do Macbook Pro. Com isso, foi possível utilizar o programa para detectar o rosto e, em seguida, os três tons dos maiores clusters. E para dar continuidade ao projeto, o primeiro tom encontrado do maior cluster foi assumido como o tom de pele da pessoa.

Como pode ser visto na figura 3.7, o programa reconheceu o rosto corretamente, e detectou três cores principais. Porém, a pessoa da foto possui um tom de pele considerado branco pálido, e o tom detectado não condiz com a realidade. A foto foi tirada em um ambiente escuro, e o tom de pele detectado ficou vários tons mais escuros que o tom de pele verdadeiro.

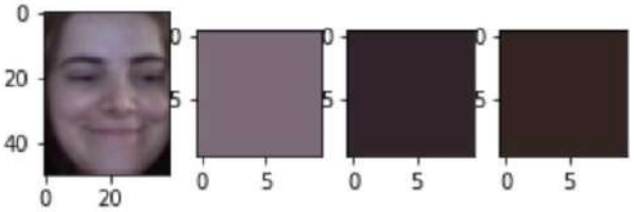


Figura 3.7: Foto da Juliana Prado, tirada usando a webcam do Macbook Pro, e reconhecida pelo programa.

O desafio verdadeiro em reconhecimento de tom de pele por meio de uma foto é a correção de luz. Caso uma foto seja tirada em um ambiente escuro, como foi feito na figura 3.7, ou em um ambiente com uma luz colorida, o programa de reconhecimento de tom de pele não deve detectar um tom que não seja o verdadeiro da pessoa.

Portanto, para dar continuidade ao projeto, foi preciso criar um algoritmo que pudesse detectar a temperatura da cor, e fazer a correção de luz antes de detectar o tom de pele. Foi a partir da implementação (40) do usuário do GitHub, Syoyo Fujita, e do artigo Color Constancy based on the Grey-Edge Hypothesis (41) que pudemos desenvolver o código abaixo.

Código 4: Correção de Luz

```
def colorCorrection(path):

  #... processando a imagem

  InputImage = cv2.imread(path)
  InputImage = InputImage.astype(np.uint8)

  outputImage = np.zeros((InputImage.shape[1-1],InputImage.shape
      [2-1],3))

  meanRedChannel = np.mean(np.mean(InputImage[::1]))
  meanGreenChannel = np.mean(np.mean(InputImage[::2]))
  meanBlueChannel = np.mean(np.mean(InputImage[::3]))

  LRed = meanRedChannel / 128
  Lblue = meanBlueChannel / 128
  LGreen = meanGreenChannel / 128

  outputImage[:, :, 0] = InputImage[:, :, 0] * (0.9 / LRed)
  outputImage[:, :, 1]  = InputImage[:, :, 1] * (0.9 / LGreen)
  outputImage[:, :, 2] = InputImage[:, :, 2] * (0.9 / Lblue)

  outputImage = outputImage.astype(np.uint8)
  outimageName = 'corrected.jpg'
```

```
24    outimageName = os.path.join(outDir, outimageName)
25    cv2.imwrite(str(outimageName),outputImage)
26    return outimageName
```

Uma forma simples de modelar o efeito de uma luz em um objeto é, assumindo que a cor refletida C = (cr; cg; cb) acontece devido a uma luz L = (lr; lg; lb) que interage com uma tinta T = (tr; tg; tb), e satisfazendo T * L = C, então, em cada pixel temos (tr * lr ; tg * lg ; tb * lb) = (cr; cg; cb). Porém, calcular L e T, dado C, é um problema mal-posto, já que, por exemplo, se temos um pixel de cor vermelha em uma imagem; R = (255; 0; 0), ele pode aparentar ter essa cor por causa de uma luz **branca** L = (1; 1; 1) em um objeto **vermelho** O = (255; 0; 0), ou, uma luz **vermelha** L = (1; 0; 0), em um objeto **branco** O = (255; 255; 255).

Para resolver esse problema podemos observar a imagem em *grey scale*. Podemos usar o *Gray World Assumption* (42) que é um método de *White Balance* que assume que qualquer imagem, em média, tem um tom cinza neutro. Lembrando que, qualquer cor (r; g; b), onde r = g = b, é considerada cinza. Portanto, esse método é verdadeiro caso tenha uma boa distribuição de cor na imagem, e, assumindo isso, a cor média refletida é reconhecida como a cor da iluminação da imagem. Portanto, podemos estimar que a cor da iluminação projetada é a cor média e compará-la com o cinza.

Assumindo que a cor média de uma imagem que esteja com uma iluminação branca L = (1; 1; 1) é (128; 128; 128), dado uma imagem, a cor da iluminação pode ser calculada como L = (mediaR/128 ; mediaG/128 ; mediaB/128), onde mediaR, mediaG e mediaB são as médias dos valores dos canais de RGB da imagem.

Portanto, a cor verdadeira de um pixel pode ser dada por cr = ir x 128/mediaR; cg = ig x 128/mediaG; cb = ib x 128/mediaB; onde ir; ig; e ib são os canais de RGB da imagem processada pelo programa. O Código 4: Correção de Luz recebe um caminho para a imagem *InputImage*, e retorna a

imagem corrigida usando os cálculos mencionados acima.

Infelizmente o código acima não funcionou para todos os tipos de iluminações de forma automática, foi preciso de intervenção humana para alterar os valores nas linhas 28 - 30 que dividem o componente de luz. O valor 0.9 foi o mais utilizado pois era o que se encaixava com imagens tiradas com pouca ou média iluminação. Porém, caso a imagem seja tirada em um ambiente iluminado, e a cor da pele do indivíduo seja branca, esses valores fazem com que a escala Azul do RGB da imagem estoure. Esse resultado será observado em mais detalhes no Capítulo 4.

Para finalizar esta parte do projeto, foi preciso classificar os tons de pele encontrados utilizando a escala de Taylor Hyperpigmentation. A escala de Taylor Hyperpigmentation foi desenvolvida para conseguir avaliar, de forma visual, o tom de pele e monitorar os avanços da hiperpigmentação após o tratamento(20). Essa ferramenta possui quinze cartões de plástico coloridos nos tons de pele variados que se encaixam à indivíduos de tons de pele Fitzpatrick I até VI (20). Cada cartão possui gradações de tons de pele que representam os níveis de hiperpigmentação (20). Exemplos destes cartões podem ser vistos na figura 3.8.

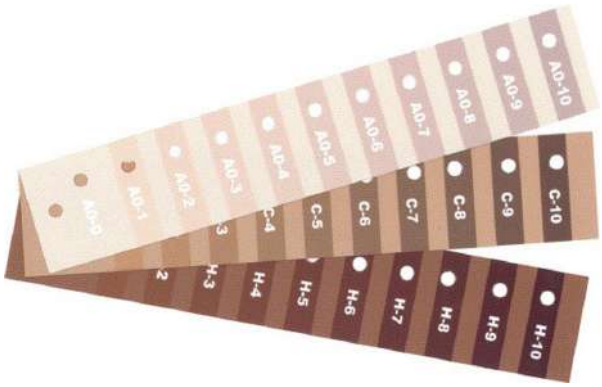


Figura 3.8: Cartões que possuem gradações de tons de pele que representam os níveis de hiperpigmentação.

Porém, a escala principal de Taylor Hyperpigmentation separa os tons de pele em cinco principais tons que se encaixam nos cinco tons de pele Fitzpatrick(43), por isso, a escala utilizada foi dividida em cinco tons básicos,

como pode ser visto na figura 3.9: Light, Light Medium, Medium, Medium Deep e Deep. Esses nomes foram escolhidos por serem comuns no mundo da indústria da beleza para tipos de base.

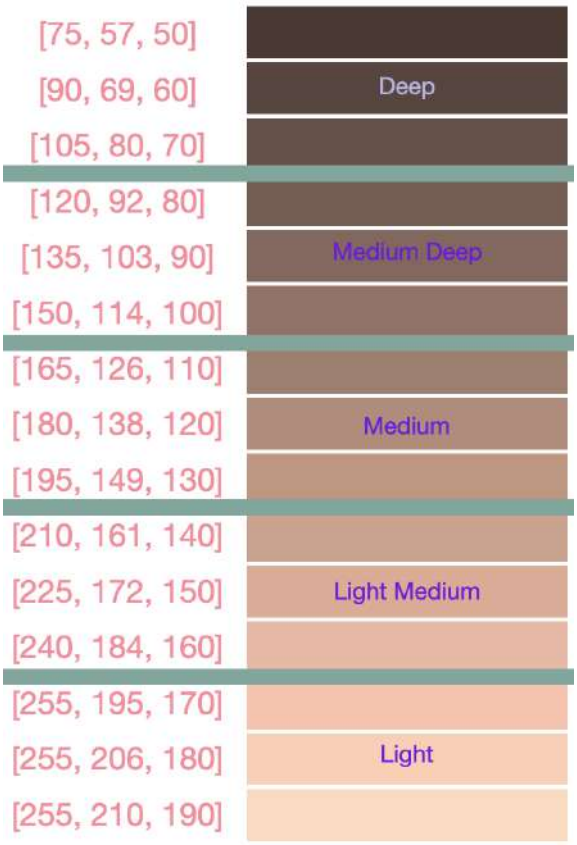


Figura 3.9: Escala utilizada pra classificar os tons de pele.

Portanto, os tons de pele forma classificados utilizando a escala da figura 3.9 em cima dos valores de RGB do primeiro tom de pele detectado e retornado pelo programa. Os resultados do reconhecimento do tom de pele foram como esperados mas apenas com intervenção humana para determinar a temperatura da luz na imagem inserida no programa. Este método não é o ideal para a solução final, e será revisitado no futuro. Uma análise mais aprofundada dos resultados desta seção podem ser vistos no capítulo 4.

4
Resultados

4.1
Reconhecimento de Logomarcas

Depois do desenvolvimento, foi avaliada a performance do modelo YOLOv5 de reconhecimento de logomarcas, usando seis métricas principais:

- *Precision*;
- *Recall*;
- mAP (*mean Average Precision*), quando o IoU (*Intersection Over Union*) está em 0.5 (50%) e 0.95 (95%);
- *Objectivess*;
- *Classification* e
- *Box*.

Modelos de detecção de objeto fazem predições a partir de *bounding boxes* e rotulações. Para cada *bounding box* é calculada a sobreposição entre a *bounding box* prevista e a *bounding box* verdadeira. Isso é calculado usando o IoU da forma descrita na figura 4.1.

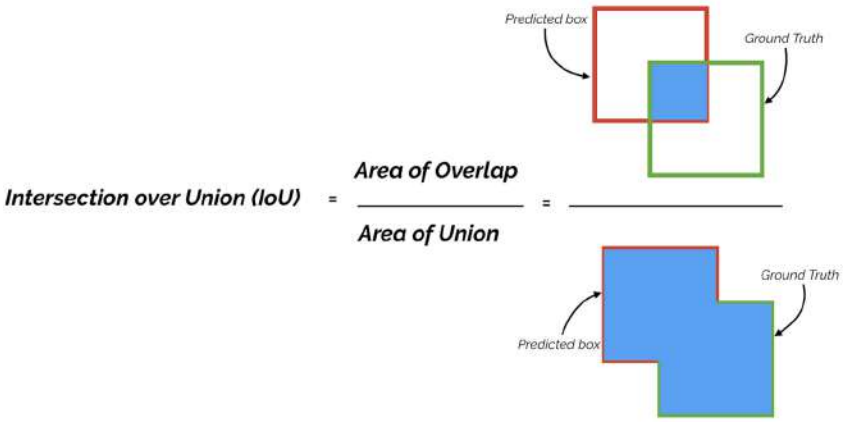


Figura 4.1: Cálculo do IoU (*Intersection Over Union*) (3)

A partir do IoU, é possível calcular a *Precision* e o *Recall*, dependendo dos limites iniciais usados para o IoU. Por exemplo, observando a figura 4.2,

dado um limite inicial de 0.4 do IoU, e o valor do IoU de uma predição for maior, sendo igual a 0.6, então ele é considerado um **Verdadeiro Positivo**, da mesma forma, se o valor do IoU de uma predição for menor, sendo igual a 0.2, então ele é considerado um **Falso Positivo**.

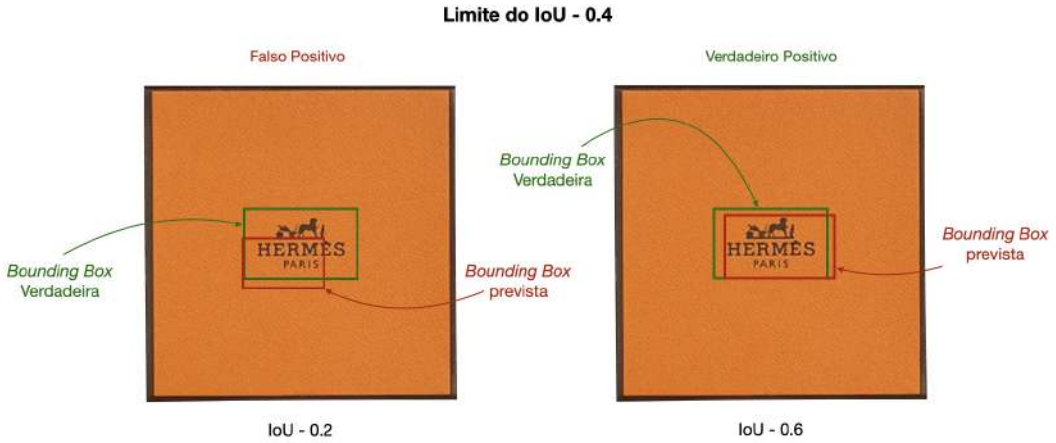


Figura 4.2: Exemplo de limite inicial de IoU na logo da Hermès (4).

A *Precision* calcula a acurácia das predições, ou seja, quantas predições que o modelo fez que estavam de fato corretas. A precisão, portanto, informa a habilidade do modelo de identificar apenas os objetos relevantes. Um modelo que produz nenhum falso positivo teria uma precisão de 1.0, porém, o valor ainda seria 1.0 se tivessem *bounding boxes* que não foram detectadas, mas que deveriam ter sido. Portanto, a precisão se calcula (44) da seguinte forma:

$$Precision = \frac{TP}{TP + FP} \tag{4-1}$$

Onde TP = *True Positives* (Verdadeiro Positivo) - o que é predito como sendo positivo, e estava correto - e FP = *False Positives* (Falso Positivo) - o que é predito como sendo positivo e estava incorreto.

Já o *Recall* é a habilidade que o modelo possui de achar todas as *bounding boxes* verdadeiras, ou seja, a proporção de positivos que foram identificados corretamente. Um modelo que não produz nenhum falso negativo - não houve nenhuma *bounding box* que não foi detectada mas deveria ter sido - tem uma proporção de 1.0. Porém, mesmo que o modelo esteja detectando de forma

exagerada e *bounding boxes* estejam sendo detectadas erroneamente, o recall ainda seria 1.0. Desta forma, o *Recall* é calculado (44) da seguinte forma:

$$Recall = \frac{TP}{TP + FN} \tag{4-2}$$

Onde TP = *True Positives* (Verdadeiro Positivo) - o que é predito como sendo positivo, e estava correto - e FN = *False Negative* (Falso Negativo) - onde o modelo falhou em detectar um objeto que estava lá.

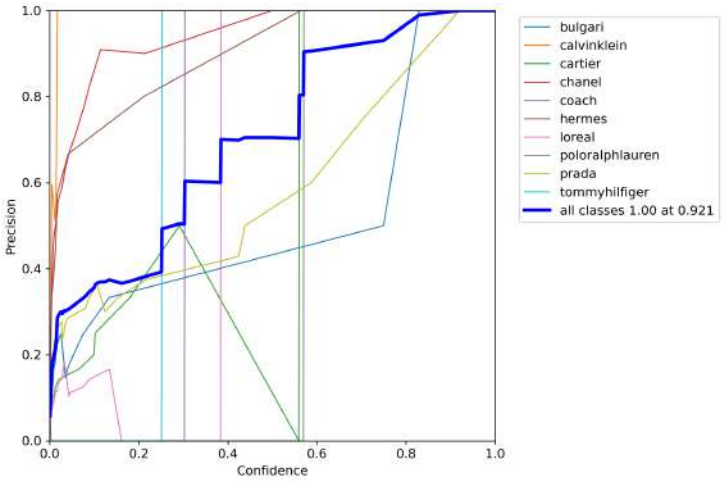


Figura 4.3: Valores de *Precision* de todas as classes do modelo YOLOv5 personalizado.

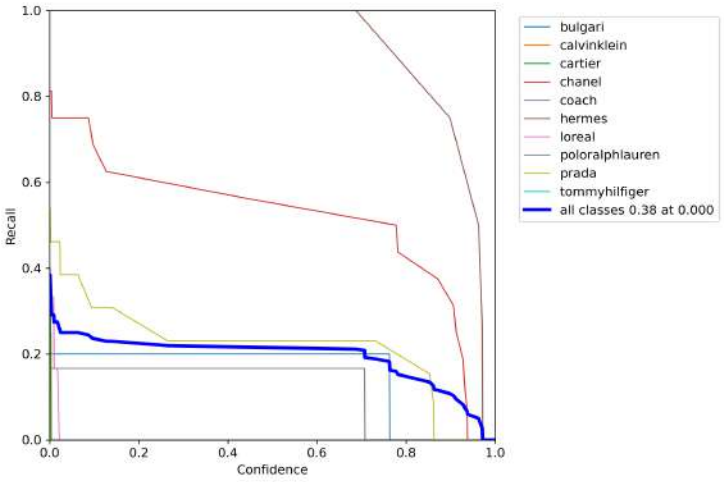


Figura 4.4: Valores de *Recall* de todas as classes do modelo YOLOv5 personalizado.

Como pode ser visto nas figuras 4.3 e 4.4, normalmente na medida em que o valor da confiança cresce, o valor de *Precision* aumenta e o valor de

Recall diminui (45). Ao determinar o valor máximo de F1 é possível balancear, de forma ótima, os valores de *Precision* e *Recall*. Os valores de F1 são calculados da seguinte forma (45):

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{4-3}$$

Onde P = *Precision* e R = *Recall* (44).

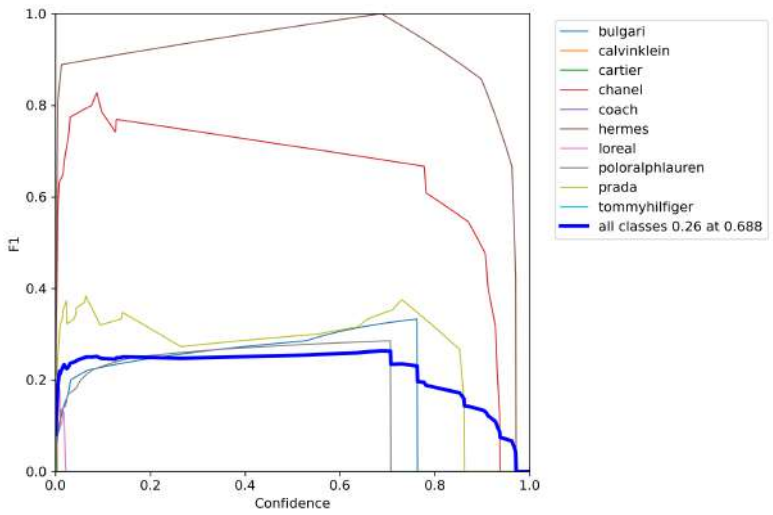


Figura 4.5: Curva de valores de F1 do modelo YOLOv5 personalizado.

A curva de valores de F1 do modelo YOLOv5, figura 4.5, mostra que o melhor valor de confiança para este caso é de 0.7, e se observarmos os valores de *Recall* e *Precision*, podemos confirmar nas figuras 4.4 e 4.3 respectivamente, que este valor é realmente o ideal. A partir do ponto de confiança 0.7 os valores de *Recall* começam a decair e os valores de *Precision* aparentam estar em seu máximo. Este valor de confiança será utilizado quando o modelo for retreinado.

Também, é possível observar em todas as figuras acima, tanto de *Recall*, figura 4.4, quanto de *Precision*, figura 4.3 e F1, figura 4.5, que algumas classes possuem quedas súbitas nos valores do eixo y. Isso se dá pela quantidade de fotos que existem nessas classes. Na figura 4.6 é possível ver a distribuição das fotos entre as classes. Existe uma diferença de mais de 137 fotos entre a classe com maior quantidades de fotos, na figura com o nome de "8", e a classe com a menor quantidade de fotos, na figura com o nome de "7".

Essa má distribuição de quantidade de fotos afeta os resultados do

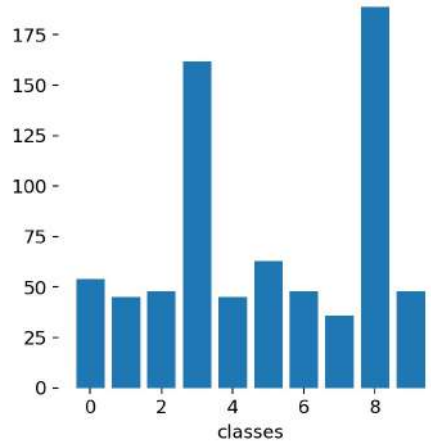


Figura 4.6: Distribuição de fotos por classe.

modelo. É possível observar a influencia da falta de imagens na *confusion matrix* do modelo personalizado, figura 4.7. Fica claro que o modelo possui mais confiança com as duas classes Hermès e Cartier em comparação com as demais. O modelo detectou com 100% de confiança de que as logos nas fotos não pertenciam a uma classe, quando na verdade pertenciam sim, isso pode ser visto na linha horizontal com a legenda *background FN*.

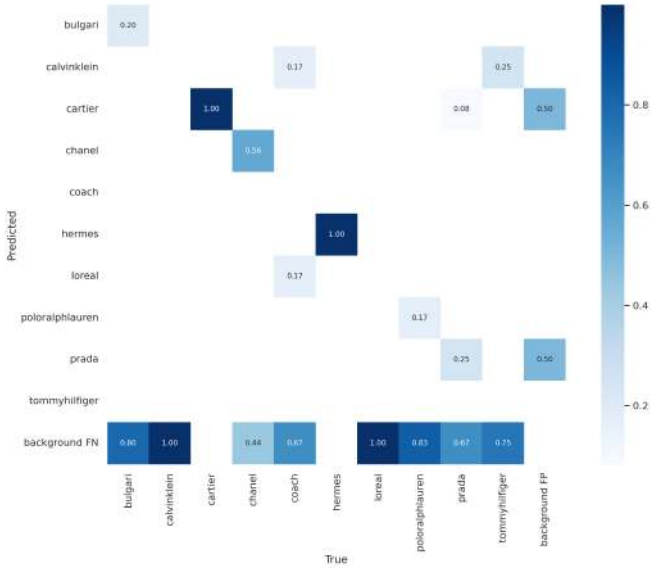


Figura 4.7: *confusion matrix*

A relação entre *Precision* e *Recall* se resumem ao limite do IoU: quanto maior o limite, menor o risco da detecção exagerada de objetos. Mas aumenta o risco de detecções perdidas(46). Portanto, essas duas métricas são muito

importantes para definir a performance do modelo, e é possível gerar o gráfico de PR (*Precision Recall*), com a *Precision* no eixo y e *Recall* no eixo x, como pode ser visto na figura 4.8.

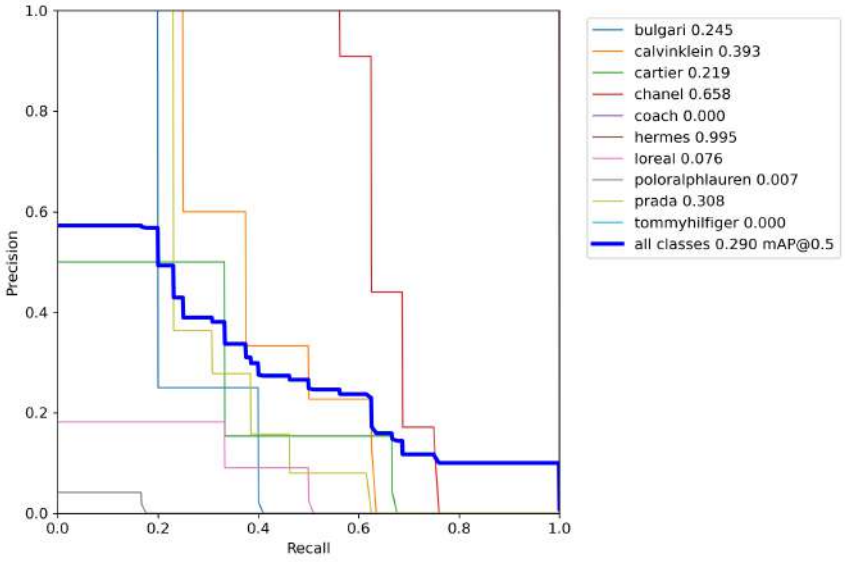


Figura 4.8: Gráfico de mAP do modelo personalizado YOLOv5.

A partir desse gráfico, é possível pegar a média da área embaixo da curva, gerando a AP (*Average Precision*). Quanto maior a curva na parte direita do gráfico de *Precision Recall*, maior a área, e maior a AP(3).

Já a métrica mAP é a média de todas as APs, sobre todas as classes e/ou sobre todos os limites de IoU (47). Para este modelo, o mAP foi calculado sobre todas as classes usando o limite fixo de IoU em 0.5 e também, começando de um limite de IoU em 0.5, e em passos de tamanho de 0.5, chegando até 0.95 (44). Resultou em uma média de todas as APs sobre todas as classes, sobre dez valores de IoUs diferentes. Vale a pena ressaltar que para muitos modelos, PR e mAP são considerados a mesma coisa.

Como pode ser visto no gráfico 4.8, o valor de mAP encontrado em um limite de IoU de 0.5 foi de 0.29. Esse valor baixo não é o ideal para um modelo de detecção de objetos, mas é o esperado quando o banco de dados não possui um número adequado de imagens para todas as classes. Portanto, essas classes estão trazendo o mAP para baixo.

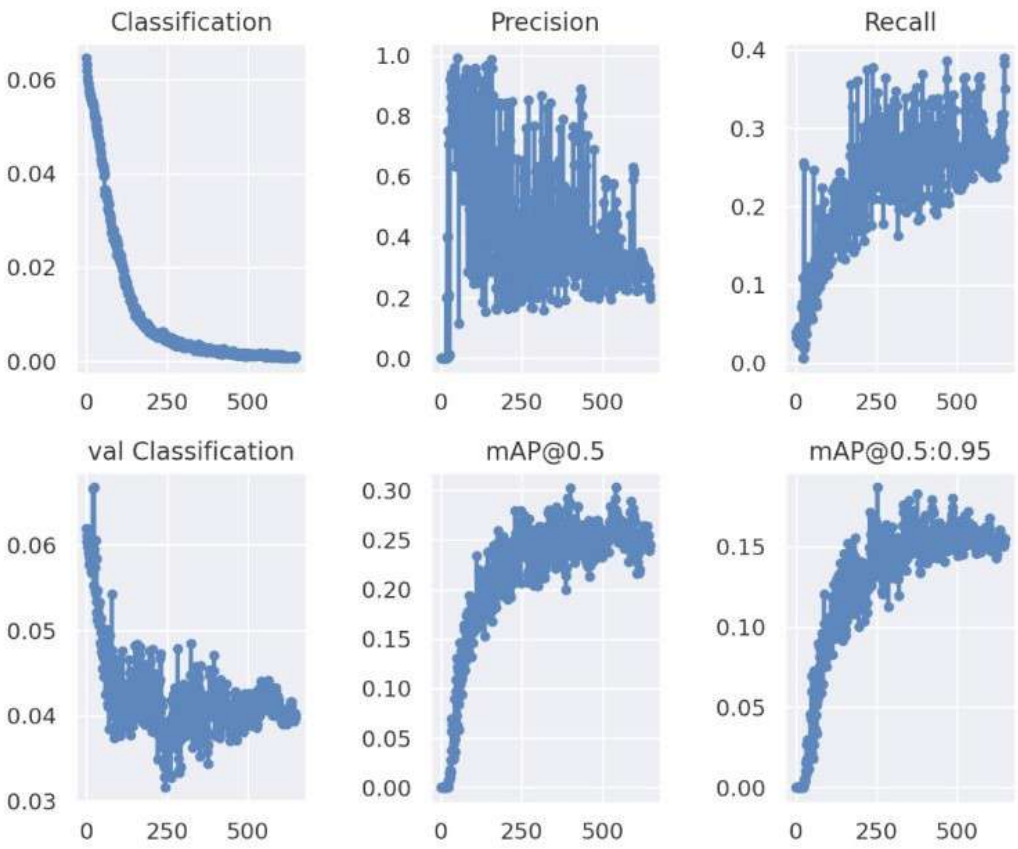


Figura 4.9: Gráficos de Precisão, Recall e mAP do processo de treinamento do YOLOv5.

Levando em consideração as métricas mencionadas, os resultados deste modelo foram satisfatórios, como pode ser visto nas figuras 4.9. Atualmente, o modelo está com 80% de precisão para algumas classes. Como foi dito acima, para a detecção de objeto, a função de perda é normalmente usada para descrever a diferença entre o valor previsto e seu valor verdadeiro do modelo. As funções de perda utilizadas no modelo YOLOv5 são incluídas em três partes; *bounding box regression loss*, *confidence loss* e *classification loss*.

A função de *bounding box regression loss* determina quando um *bounding box* é detectado corretamente e, também, quando existe um objeto dentro dele. O modelo atual está com a perda de *bounding box* próximo a 0, com 0.02 (na figura 4.10 com o nome *Box*). Já a função de *confidence loss* determina a confiança que o modelo tem de que existe um objeto dentro do *bounding box* e, também, a confiança em detectar quando não existe um objeto dentro

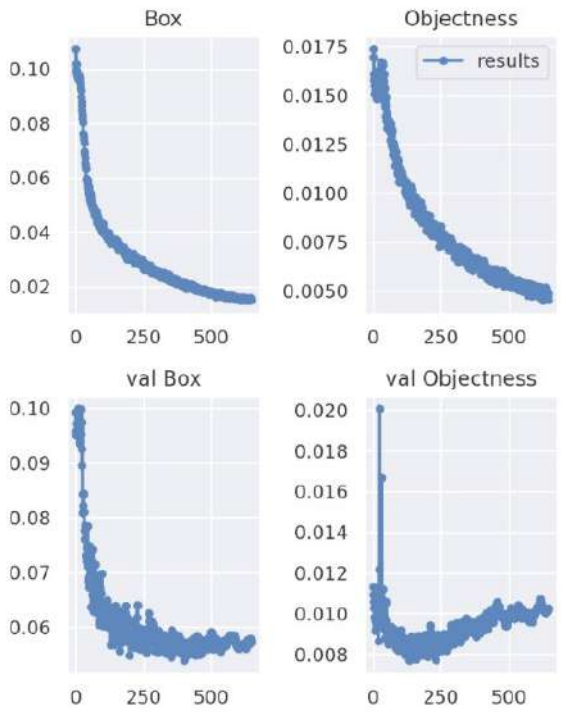


Figura 4.10: Resultados do modelo de YOLOv5.

do *bounding box*. Este modelo está com a perda de confiança próximo a 0, com 0.0050 (na figura 4.10 com o nome de *Objectness*). Por fim, a função de *classification loss* determina a confiança de que o modelo detectou um objeto pertencendo a classe correta. Este modelo está com uma *classification loss* próxima a 0, abaixo de 0.01 (na figura 4.10 com o nome de *Classification*).

Os resultados foram, em partes, positivos, mas foi preciso rodar em um vídeo do YouTube para ter a certeza de que o modelo se comportaria da mesma maneira em uma situação mais compatível com a realidade do projeto. Vídeos do YouTube não têm uma iluminação padrão, e há movimentos constantes que podem distorcer a imagem. O vídeo escolhido foi o do influenciador Jeffree Star, que no vídeo, utiliza maquiagem da marca Chanel. Esta marca foi escolhida por ter obtido resultados medianos de precisão, como pode ser visto no *confusion matrix* da figura 4.7.

Os resultados do modelo ficaram satisfatórios: a detecção está dentro dos padrões descritos acima, chegando a confianças de 0.81. Seguem frames do vídeo utilizado, e é possível observar que houve uma detecção correta da

marca Chanel na embalagem da base utilizada por Jeffree 4.11.

Figura 4.11: Frames do vídeo do YouTube.

A foto superior esquerda apresenta uma distorção da marca Chanel por causa do ângulo em que a embalagem aparece no vídeo (posição da câmera). Mesmo assim a precisão está em 81%. Portanto, com resultados satisfatórios que atendem às necessidades deste projeto, foi possível seguir para a segunda etapa do projeto.

4.2
Reconhecimento de Tons de Pele

Os primeiros resultados do programa de reconhecimento de tom de pele foram satisfatórios, como pode ser visto na figura 4.12.

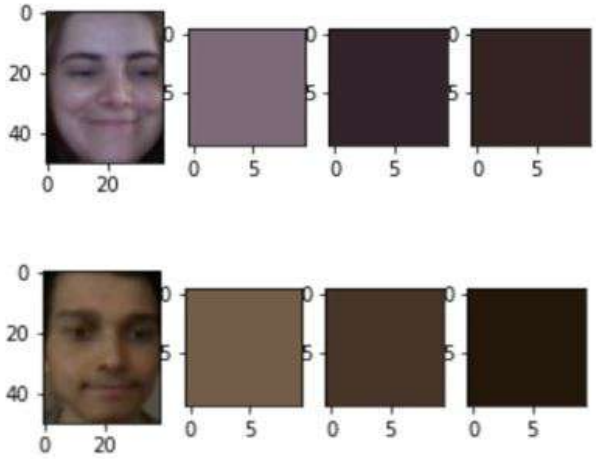


Figura 4.12: Primeiros testes realizados.

Porém, quando testado em luzes diferentes, foi possível notar que o tom de pele reconhecido mudava consideravelmente. A figura 4.13 foi tirada usando dois tipos de luzes diferentes, uma luz difusa do céu, e uma luz amarela de uma lâmpada de LED.

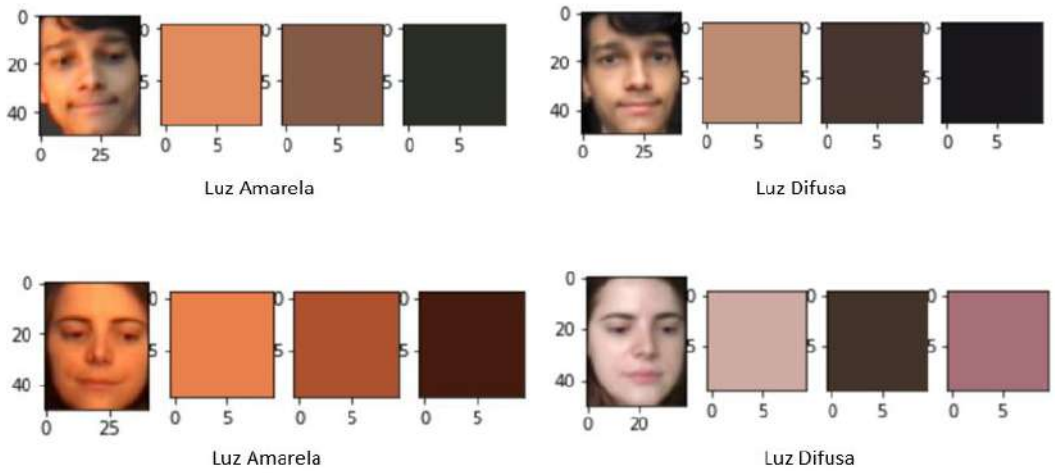


Figura 4.13: Dois tipos de luzes diferentes: luz difusa do céu, e luz amarela de uma lâmpada de LED.

Com esses resultados, e alguns testes em uma pessoa com o tom de pele mais claro, foi possível observar que era preciso ter alguma correção de cor

antes de passar a imagem pelo programa. Portanto, como foi visto no capítulo anterior, as imagens foram passadas pelo Código 4, e o resultado inicial foi extremamente satisfatório, como pode ser visto na imagem 4.14.

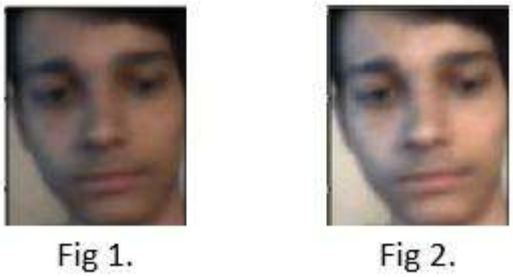


Figura 4.14: A Fig 1 está sem a correção, Fig 2 está com a correção.

Porém, ao testar o mesmo código em uma pessoa com o tom da pele mais claro, a correção de cor estourava o canal azul da imagem. Isso interferia na cor reconhecida, já que, onde havia mais brilho no rosto dessa pessoa, era transformado em uma cor azulada, vide a imagem 4.15. Infelizmente, não foi possível deixar o código automatizado: as imagens, portanto, foram corrigidas de forma manual.

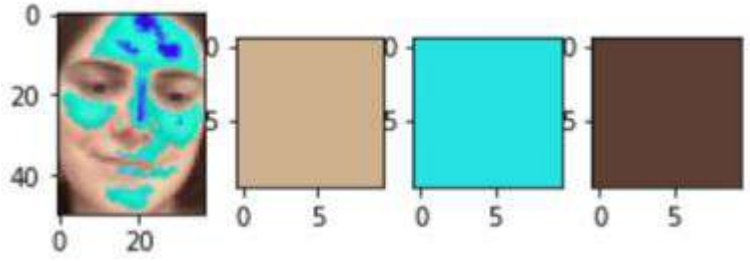


Figura 4.15: Imagem com uma iluminação branca em uma pessoa com tom de pele claro.

A partir desses resultados foi preciso classificar os tons de pele encontrados utilizando a escala de Taylor Hyperpigmentation em cima dos valores de RGB do primeiro tom de pele detectado e retornado pelo programa, como foi descrito no capítulo anterior.

As fotos utilizadas sofreram correções de luz manualmente, porém, o ambiente em que as pessoas sujeitas aos testes estavam era bem iluminado

com luz branca difusa do céu. Também, foi certificado de que todos tirassem as fotos olhando diretamente para a câmera de forma a não criar sombras em seus rostos. Foram tiradas algumas fotos consecutivas para ter certeza de que o algoritmo estava retornando os mesmos valores.

A classificação foi bem sucedida e ficou satisfatória, reconhecendo o tom de pele e classificando ele de forma que a pessoa sujeita ao teste ficou satisfeita idem. Foi feito um teste em quatro tipos de pele diferentes, alguns exemplos se encontram abaixo nas imagens 4.16.

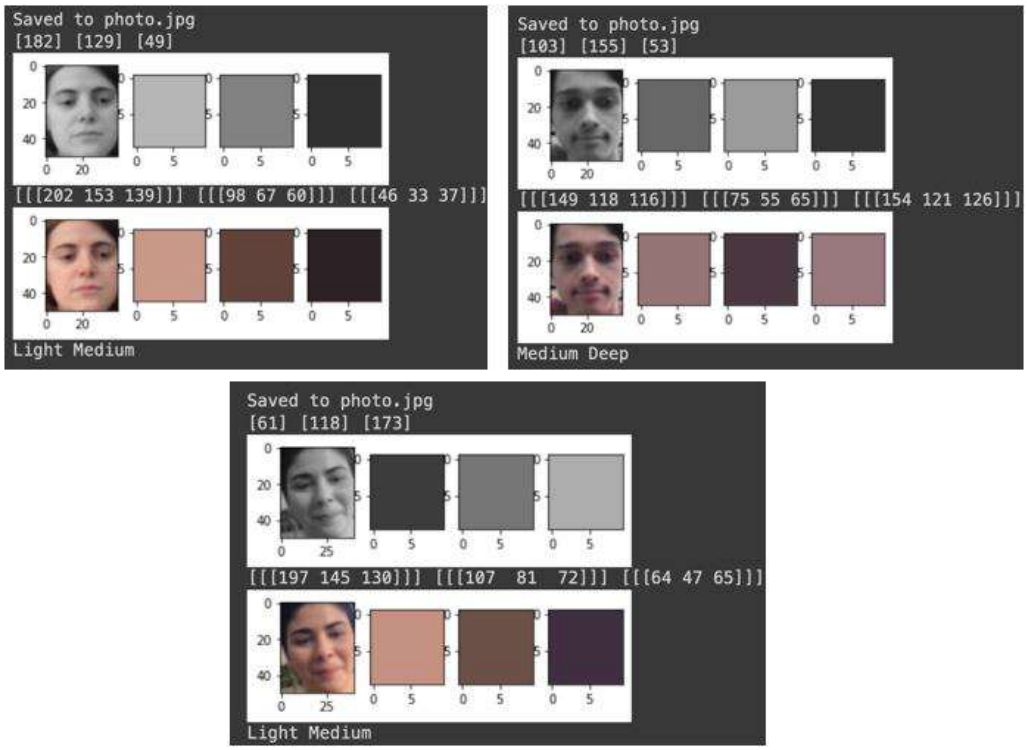


Figura 4.16: Classificação do tom de pele de três usuários.

Mas, como em alguns momentos para uma das usuárias testadas, - na figura 4.16 a usuária de baixo - o algoritmo retornava valores que variavam entre dois tons de pele; Light Medium e Medium o algoritmo foi rodado em um vídeo que foi gravado no ambiente em que a pessoa sujeita estava quando testou a primeira vez. Ao rodar o algoritmo em cada frame do vídeo, foi feito uma média total das classificações. Os resultados podem ser vistos na figura 4.18

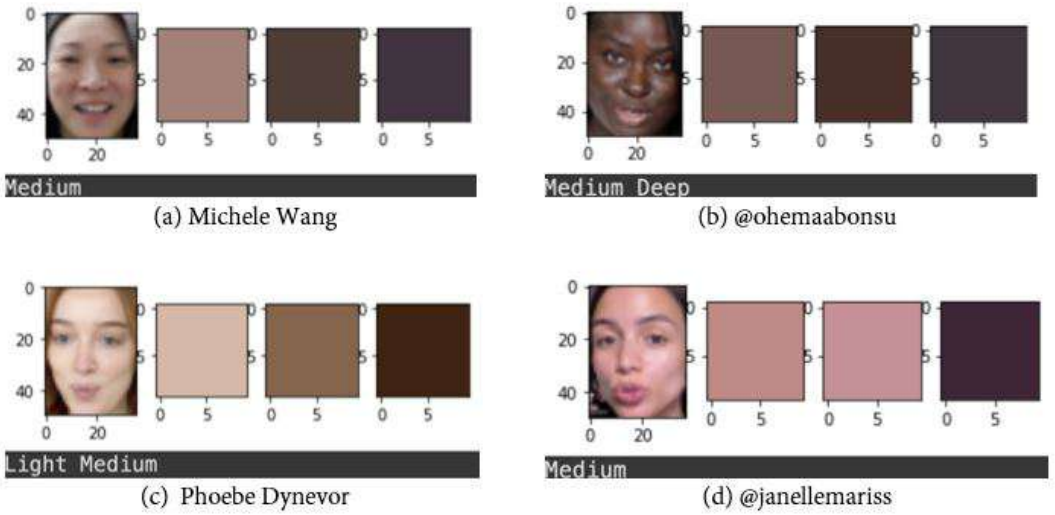
É possível observar que o algoritmo ficou dividido entre duas classes, Light Medium e Medium. Apesar de 0.240674% das classificações terem

	Tom de Pele	%
0	Light	0
1	Light Medium	49.6992
2	Medium	49.4986
3	Medium Deep	0.240674
4	Deep	0.561572

Figura 4.17: Média de classificações de tom de pele do vídeo.

sido de Medium Deep e 0.561572% de Deep, foi feita uma busca pelos frames que obtiveram essas classificações e foi possível ver que o algoritmo de reconhecimento de face detectou um rosto em lugares impróprios - como por exemplo em seu cabelo - e, portanto, esses resultados não foram considerados. Quanto a oscilação entre tons de pele, isso se deu ao fato da pessoa mexer o rosto gerando uma sombra que deixava o rosto dela um tom mais escuro.

Porém, o sistema do objetivo final deste projeto não cogita que a detecção de tom de pele seja feita através de vídeos, mas sim de fotos tiradas em ambientes claros com luz branca e com o rosto posicionado olhando diretamente para a câmera. O tom de pele classificado quando a usuária olhava diretamente para a câmera foi a que à deixou satisfeita.



(a) Michele Wang (b) @ohemaabonsu (c) Phoebe Dynevor (d) @janellemariss

Figura 4.18: Classificação do tom de pele de quatro *influencers*: (a) Michele Wang (5), (b) Ohemaa (6), (c) Phoebe Deynevor (7) e (d) Janelle Mariss (8)

Com os tons de pele sendo classificados da forma correta, foi preciso testar em *influencers* para ter a certeza de que o algoritmo se comportaria da mesma maneira em uma situação mais compatível com a realidade do projeto. Recortes de quatro vídeos foram utilizados como pode ser visto na figura 4.18.

A classificação utilizando fotos estáticas serão a forma principal de classificar o tom de pele do usuário do sistema. Tanto para *influencers* quanto para usuários comuns. Já que o tom de pele é um valor estático que, caso mude com o tempo, como por exemplo com pessoas com Vitiligo, basta tirar uma nova foto em um ambiente iluminado para fazer a re-classificação. Portanto, não foi preciso rodar o algoritmo em um vídeo.

5
Conclusão e Trabalhos Futuros

O desenvolvimento descrito acima serve como base para a proposta final do projeto de desenvolver um programa capaz de detectar o tom de pele do usuário e recomendar marcas de maquiagem que possuem produtos que se encaixem no tom de pele detectado. Assim como detectar o tom de pele dos influenciadores e os produtos utilizados em seus vídeos, para facilitar a busca e possibilitar que eles lucrem com os produtos que forem comprados por meio dos seus reviews.

Com o desenvolvimento acima, as logomarcas foram detectadas em produtos de beleza com sucesso em vídeos que se encaixam nas premissas da minha proposta. Também, vale à pena ressaltar que, apesar da precisão estar em 80%, algumas logomarcas estão com resultados inferiores e insatisfatórios devido ao baixo número de imagens para treino e teste.

Para trabalhos futuros será preciso achar um banco de dados com um número maior de imagens para melhorar a classificação de algumas marcas. Também seria ideal trabalhar com um banco de dados com um número maior de marcas, pois apenas 10 marcas de produtos de beleza foram utilizadas para treinar o modelo. Ao conseguir novas marcas e retreinar o modelo, o valor máximo de F1 encontrado será utilizado para balancear, de forma ótima, os valores de *Precision* e *Recall*.

Também, seria interessante que o programa fosse capaz de detectar o produto e não só a marca. Para isso, teria que existir um banco de dados de produtos de beleza, algo que ainda não pôde ser encontrado. Com sites de comércio tais como a Sephora (4) que possui inúmeros produtos de beleza, seria possível montar um banco de dados através do próprio, ou até mesmo em uma parceria com a loja. Também, para este ponto do projeto seria interessante ter um sistema de recomendação para os usuários.

Quanto a detecção de tons de pele, os resultados foram satisfatórios para um código inicial, e servirá como base. Para trabalhos futuros, é preciso automatizar a correção de luz, para que o programa detecte sozinho a temperatura da iluminação da imagem. Esse passo é essencial para que a detecção de tons de pele seja feita corretamente. Utilizar o método de *Gray World Assumption* não foi a melhor forma de solucionar este problema como pode ser visto pelos resultados do capítulo anterior. Será preciso utilizar outro método que possa ser treinado a detectar como as cores refletem em luzes diferentes.

Em suma, os trabalhos desenvolvidos atenderam às necessidades da proposta inicial e servirão para dar continuidade e ter melhorias no futuro. A utilização do YOLOv5 foi ideal para a detecção de logomarcas por sua rapidez e facilidade de treino e retreino, e o algoritmo de detecção de tom de pele está completo e detecta não só a pele como também a cor que está disponível na foto.

6
Referências bibliográficas

1 NOTION. **Notion**. Disponível em: <https://www.notion.so/>.

2 JOCHER, G. **GitHub**. Disponível em: <https://github.com/ultralytics/yolov5>.

3 COCHARD, D. **mAP : Evaluation metric for object detection models**. Disponível em: <https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472>.

4 HERMèS URLDATE = 2022-06-11, d. . . l. . e. **Hermès**. Disponível em: <https://www.hermes.com/us/en/>.

5 MICHELE Wang. Disponível em: <https://www.youtube.com/c/MicheleWang1>.

6 OHEMAABONSU. Disponível em: <https://www.youtube.com/channel/UCYaW8i4ULhVoFPjC8uSRmsQ>.

7 BRIDGERTON'S Phoebe Dynevor on Dry Skin Care Casual Makeup | Beauty Secrets | Vogue. Disponível em: <https://www.youtube.com/watch?v=58OxTUPGdAs>.

8 JANELLEMARISS. Disponível em: <https://www.tiktok.com/@janellemariss?lang=en>.

9 YAMAMORA, f. **As expectativas para a indústria da beleza global**. Disponível em: <https://cosmeticinnovation.com.br/as-expectativas-para-a-industria-da-beleza-global>.

10 THE world of beauty in 2020. Disponível em: <https://www.loreal-finance.com/en/annual-report-2020/cosmetics-market-2-1-0/>.

11 L, C. **Statista**. Disponível em: <https://www.statista.com/statistics/294655/youtube-monthly-beauty-content-views/>.

12 SURGEONS, d. **Digital Surgeons**. Disponível em: <https://www.digitalsurgeons.com/thoughts/strategy/how-youtube-has-drastically-changed-the-beauty-industry/>.

13 NEWYORK Times. Disponível em: <https://www.nytimes.com/2018/11/30/style/dark-skin-black-beauty-bloggers-instagram-youtube.html>.

14 EUROMONITOR International. Disponível em: <https://www.euromonitor.com/the-digital-beauty-consumer/report?utm_source=cosmoprofnorthamerica&utm_medium=article&utm_campaign=EV_20_07_28_CHI_USA_Cosmoprof\%20North\%20America/>.

15 GERDEMAN, D. **Forbes**. Disponível em: <https://www.forbes.com/sites/hbsworkingknowledge/2019/12/13/how-influencers-are-making-over-beauty-marketing/?sh=399ba1fe1203>.

16 DONELSON, g. **Tagger**. Disponível em: <https://www.taggermedia.com/blog/15-beauty-brands-using-influencer-marketing>.

17 MULSHINE, M. **Digital Surgeons**. Disponível em: <https://observer.com/2014/10/a-brief-history-of-kim-kardashians-endorsement-deals/>.

18 MMI. Disponível em: <https://www.mmi-analytics.com/blog/the-future-of-ecommerce-9-essential-beauty-industry-statistics>.

19 OAKLEY, P. A. **DermNet NZ**. Disponível em: <https://dermnetnz.org/topics/skin-phototype>.

20 TAYLOR MD; STéPHANIE ARSONNAUD; JANUSZ CZERNIELEWSKI, M. f. t. H. S. S. G. S. C. The taylor hyperpigmentation scale: A new visual assessment tool for the evaluation of skin color and pigmentation. p. 5.

21 APPLE. Challenge based learning: A classroom guide. p. 40.

22 ARUN, A. **McKinsey Digital**. Disponível em: <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/its-showtime-how-live-commerce-is-transforming-the-shopping-experience>.

23 INOVAçãO, N. **Mercado e Consumo**. Disponível em: <https://mercadoeconsumo.com.br/2021/09/23/loreal-brasil-fara-primeira-live-commerce-reunindo-marcas-do-grupo/>.

24 SENDBIRD. **Medium**. Disponível em: <https://medium.com/@Sendbird/live-commerce-is-here-is-your-company-ready-to-take-advantage-b8443387bfd>.

25 LTD ephora D. S. P. **Sephora**. Disponível em: <https://www.sephora.my/pages/virtual-artist>.

26 PARIS, L. **L´ORÉAL**. Disponível em: <https://www.lorealparisusa.com/virtual-try-on-makeup>.

27 MATNEY, L. **TechCrunch**. Disponível em: <https://techcrunch.com/2020/11/12/loreal-rolls-out-a-line-of-virtual-makeup/>.

28 PARIS, L. **L´ORÉAL Paris**. Disponível em: <https://www.lorealparisusa.com/beauty-magazine/makeup/makeup-looks/makeup-genius-changes-makeup-application-forever>.

29 SCHMIDT, L. **Maquiagem futurista: conheça a impressora de batom da Yves Saint Laurent**. Disponível em: <https://mundoconectado.com.br/noticias/v/23831/maquiagem-futurista-conheca-a-impressora-de-batom-da-yves-saint-laurent>.

30 REDMON JOSEPH AMD DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. p. 10.

31 BIANCO, S. et al. Deep learning for logo recognition. p. 10.

32 MEEL, V. **Viso AI**. Disponível em: <https://viso.ai/deep-learning/yolor/>.

33 SU, H. **WebLogo-2M Dataset**. Disponível em: <https://weblogo2m.github.io/>.

34 SU, H. **QMUL-OpenLogo: Open Logo Detection Challenge**. Disponível em: <https://qmul-openlogo.github.io/>.

35 HAZIRBAS, C. et al. Casual conversations: A dataset for measuring fairness in ai. p. 5.

36 ANAS, M.; GUPTA, R. K.; AHMAD, S. Skin cancer classification using k-means clustering. p. 4.

37 SKIN-COLOR extraction and classification. GitHub. Disponível em: <https://github.com/NikolayTV/Skin_color_classifier/blob/master/LICENSE>.

38 MTCNN. GitHub. Disponível em: <https://github.com/ipazc/mtcnn>.

39 UTKFACE. Disponível em: <https://www.kaggle.com/datasets/jangedoo/utkface-new>.

40 COMPUTE Color Correction Matrix (CCM). GitHub. Disponível em: <https://github.com/lighttransport/colorcorrectionmatrix>.

41 WEIJER, J. van de; GEVERS, T. Color constancy based on the grey-edge hypothesis. p. 5.

42 COLOR Constancy: Gray World Algorithm. Disponível em: <https://www.codeproject.com/Articles/653355/Color-Constancy-Gray-World-Algorithm>.

43 TRAINING, P. **THE IMPORTANCE OF PHOTO TYPE RISK ASSESSMENT**. Disponível em: <https://pastiche-training.com/the-importance-of-photo-type-risk-assessment>.

44 YOHANANDAN, S. **mAP (mean Average Precision) might confuse you!** Disponível em: <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2>.

45 LEBIEDZINSKI, P. **A Single Number Metric for Evaluating Object Detection Models**. Disponível em: <https://towardsdatascience.com/a-single-number-metric-for-evaluating-object-detection-models-c97f4a98616d>.

46 ARLEN, T. C. **Understanding the mAP Evaluation Metric for Object Detection**. Disponível em: <https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3>.

47 TAN, R. J. **Breaking Down Mean Average Precision (mAP)**. Disponível em: <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52#1a59>.