PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

KAIKE BATISTA DA SILVA

DETECÇÃO AUTOMÁTICA DE DEFEITOS EM MANUFATURAS UTILIZANDO REDES CONVOLUCIONAIS

Goiânia
2022

KAIKE BATISTA DA SILVA

DETECÇÃO AUTOMÁTICA DE DEFEITOS EM MANUFATURAS UTILIZANDO REDES CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação, da PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS, como requisito parcial para a Obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Talles Marcelo Gonçalves de Andrade Barbosa.

Goiânia

2022

KAIKE BATISTA DA SILVA

DETECÇÃO AUTOMÁTICA DE DEFEITOS EM MANUFATURAS UTILIZANDO REDES CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação, da PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS, como requisito parcial para a Obtenção do grau de Bacharel em Ciência da Computação.

Goiânia, 15 de Junho de 2022

BANCA EXAMINADORA

_____

Profa. Kátia Kelvis Cassiano

Universidade Federal de Goiás

_____

Prof. Dr. José Olímpio Ferreira

Pontifícia Universidade Católica de Goiás

_____

Prof. Talles Marcelo G. de A. Barbosa

Pontifícia Universidade Católica de Goiás

Dedico este trabalho a minha familia, amigos e professores que estiveram comigo durante esta jornada.

RESUMO

Neste trabalho é construido um modelo de detecção utilizando o o algoritimo You Only Look Once, YOLO utilizando a plataforma colab do google. Modelos de detecção sao usados em varias areas do conhecimento. Foi feita uma revisao teoria dos conceitos envolvendo visao computacional e uma explicação do funcionamento do algoritimo YOLO. Tambem são elicitadas caracteristicas da plataforma colab que a tornam atrativa como infraestrutura para o treinamento de modelos de detecção. O modelo criado utiliza tecnicas semelhantes as usadas em trabalhos da area que buscam soluções para detecção de defeitos.

Palavras-chave: YOLO, colab, detecção, visao computacional.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

YOLO	You Only Look Once
CNN	Convolutional Neural Network
RCNN	Region based Convolutional Neural Network
NMS	Non Maximum Supression

SUMÁRIO

1 INTRODUÇÃO

Humanos dotados do sentido da visão conseguem perceber o mundo a sua volta e criar um modelo tridimensional do mesmo de forma fácil. Fazer sentido dos objetos de uma cena, detectar profundidade ou distinguir entre pessoas diferentes e nomeá-las corretamente de acordo com suas características, são capacidades atribuídas a visão. A habilidade de perceber o mundo a nossa volta de forma visual chama a atenção da comunidade cientifica, a computação, de forma mais específica, por meio de técnicas, tenta simular a capacidade visual humana, esse campo de estudo é chamado de visão computacional (SZELISKI, 2011).

São várias aplicações no campo da visão computacional, como aplicações militares de drones que precisam acompanhar um objeto em movimento, conhecido como tracking, Zhao et al (2020) propõe um modelo de tracking melhorado para resolver o problema de voltar a acompanhar o objeto quando este fica ocluso na cena e reaparece em outro local.

A fotogrametria é o campo que se ocupa de gerar um modelo em três dimensões a partir de uma ou mais imagens de uma cena, também conhecido como mapa de profundidade, isso possibilita por exemplo criar ambientes virtuais tridimensionais para testar aplicações (MASSON, 2021).

Outra tarefa é a de detecção de imagens, classificar e informar a posição do objeto em uma imagem, como mostrado em Rath (2021), que utiliza a versão 5 do algoritmo YOLO para fazer detecção de vários tipos de veículos em uma imagem.

A tarefa de detectar imagens demanda recurso computacional. Os algoritmos têm melhor performance conforme o hardware utilizado. Em comparação, GPU's (*Graphics Processing Unit*) mais potentes, mais memória ou mais núcleos de processamento e arquiteturas mais recentes, conseguem processar mais imagens por segundo (DEEP..., 2021).

Além de considerações em relação ao hardware também há de ser feita escolhas em relação ao software utilizado para criação de um modelo de detecção, hoje existem várias ferramentas. A linguagem de programação python se destaca pela sua simplicidade e suporte de frameworks e bibliotecas especificas para essa função (PYTHON..., 2020).

Em python pode se utilizar frameworks como o tensorflow, que abstraem boa parte do processo de codificação, por vezes removem a necessidade de se escrever qualquer código (RATH, 2021). Outro framework conhecido é o DarkNet, escrito na linguagem C (REDMON, 2021).

A infraestrutura necessária para construir um sistema de detecção, (hardware e software), é fornecida por várias empresas, algumas delas são, Amazon Web services, Microsoft Azure, Google cloud e IBM Cloud (KAUR et al, 2020). Também como infraestrutura a plataforma colab é uma máquina virtual disponibilizada pelo google através de uma interface gráfica acessada pela internet. No colab se tem acesso a memória ram, CPU e uma GPU adequada para o treinamento de um modelo de detecção. (OLÁ..., 2022).

Dada a tarefa de treinar um modelo de detecção, a infraestrutura de hardware e software necessárias, neste trabalho será feita a busca que ajude a compreender e responder como a plataforma colab pode ajudar no processo de construção de um modelo de detecção de imagens?

Construir um modelo de detecção passa por hardware especializado como GPU's (DEEP..., 2021). Uma medida de desempenho é o *throughput* (Taxa de transferência), este mede a quantidade de imagens que o modelo consegue processar por segundo. Maior *thoughput* significa aumentar a velocidade de processamento do modelo (*What*... , 2022). Uma comparação feita em GPU... , 2022, mostra o *throughput* em diferentes GPU's. A GPU RTX 2070 tem a colocação mais baixa na métrica de *throughpu*, a mais alta colocação foi da GPU A100. Em uma comparação de preços A GPU RTX 2070 pode ser encontrada por R$2.764,68 (PLACA..., 2022) enquanto a A100 pode ser encontrada por $13,999.00 (NVIDIA... , 2022). A tendência é que GPU's mais potentes custem mais caro.

O estudo feito pelo Instituto Semesp desenha o perfil do estudante de ensino superior no brasil. A conclusão foi que boa parte dos estudantes de graduação trabalham, 61,8% nas instituições privadas e 40,3% em instituições públicas. Desses 72% tem uma renda média de até 2 salários mínimos (PEDUZZI, 2020). Portanto acesso a um hardware adequado para treinamento de um modelo de detecção pode estar inacessível para boa parte dos estudantes, visto o valor de uma GPU.

Além da barreira de infraestrutura, desenvolver um modelo de detecção passa por uma série de conhecimentos específicos de certas tecnologias. Em resumo, uma linguagem de programação é necessária, a comunidade tem utilizado o python com maior frequência (PYTHON..., 2020). Desenvolver os algoritmos em python puro pode ser extremamente difícil. Frameworks foram criados para auxiliar nesse processo, como o tensorflow (ABADI et al, 2015).

O desenvolvedor pode utilizar um serviço externo como infraestrutura, como por exemplo a plataforma do google cloud que tem como serviço o vertex AI. neste serviço o usuário pode fazer o treinamento utilizando o hardware do google cloud (Vertex... ,

2022). Para trabalhar com o vertex AI o usuário precisa entender de Docker e configurar um container para utilizar os serviços (CONTAINERS... , 2022). Também é necessário se familiarizar com a plataforma do google cloud e aprender a utilizar os comandos da janela de comandos personalizada do google cloud (CLOUD... , 2022). Outros serviços de cloud também fornecem plataformas de infraestrutura que podem servir para treinar modelos de detecção. Cada plataforma possui também suas próprias camadas de complexidade.

Neste trabalho através da construção de um modelo de detecção será analisado como o colab fornece solução para a acessar uma infraestrutura de treinamento dos modelos de detecção e como ele lida com a complexidade da configuração de um ambiente de um modelo de detecção.

1.1 OBJETIVO

1.1.1 Objetivo Geral

Compreender como a plataforma colab pode ajudar no processo de construção de um modelo de detecção de imagens.

1.1.2 Objetivos Específicos

- identificar e avaliar características e recursos da plataforma colab para o estudo da detecção de objetos em imagens
- verificar tecnologias de experimentação no colab (yolo)
- documento de referência para trabalhos futuros que tenham escopo de detecção de objeto em imagens

2 FUNDAMENTAÇÃO TEÓRICA

2.1 VISAO COMPUTACIONAL

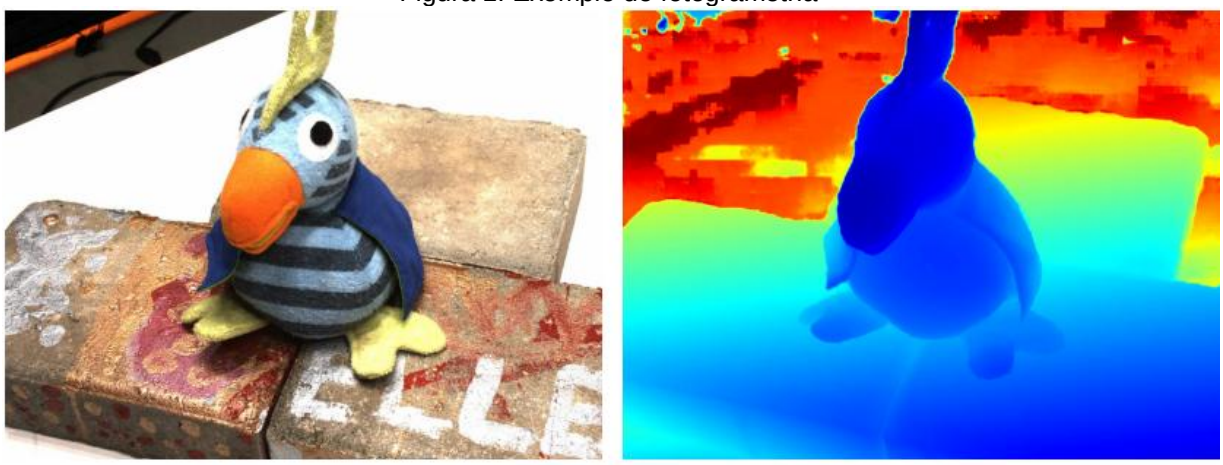
O ser humano, dotado do sentido da visão, consegue realizar várias tarefas. Descrever uma cena e nomear os objetos contidos nela, entender a profundidade ou distância em que objetos se encontram e acompanhar um veículo em movimento, para calcular a melhor oportunidade de atravessar a rua. As habilidades da visão humana chamam a atenção da ciência, mais especificamente da ciência da computação (SZELISKI, 2011).

Na subárea da visão computacional são construídos modelos computacionais

que realizem tarefas relacionadas a visão humana como as citadas (BARELLI, 2018). É um campo que relaciona conhecimentos da matemática e da computação e serve de ferramenta para construção de aplicações nas mais diversas áreas, na saúde (BACCIU, LISBOA, VELLIDO, 2022), em carros que se dirigem sozinhos (BEHERE, TÖRNGREN, 2015).

Uma aplicação da visão computacional e a fotogrametria. Essa área da computação tenta dizer qual a profundidade de objetos em uma cena. Nesse tipo de aplicação uma imagem é entregue a um modelo e este através de cores quentes e frias mostra a profundidade da cena. O que está mais próximo fica com cores mais frias e o mais longe fica com a cor mais quente (MASSON, 2021). Essa dinâmica pode ser vista na Figura 1 abaixo:

Figura 1: Exemplo de fotogrametria



(a) Imagem de referência. (b) Mapa de profundidade.

Fonte: MASSON, 2021

Outra área é a área do tracking. A tarefa de tracking se trata de acompanhar um objeto em movimento (MILANO, BARROZO, 2022). Um exemplo de aplicação em (ZHAO *et al*, 2020), tenta resolver o problema de oclusão no *tracking*. Nesse problema um objeto em movimento pode ficar atras de algo e se tornar ocluso na cena. A tarefa é que mesmo após a oclusão do objeto conseguir acompanhá-lo. Na Figura 2 temos um drone acompanhando uma pessoa de camiseta branca. Em um dado momento a pessoa fica oclusa embaixo da tenda. O drone mesmo após a oclusa consegue detectar a pessoa de camiseta branca.

Figura 2 drone detecta objeto apos oclusão

Fonte: ZHAO *et al*, 2020

Outra aplicação de visão computacional é a detecção. Na tarefa de detecção geralmente um objeto é identificado na cena por um retângulo conhecido como *bounding box*. O *bounding box* geralmente contém o nome da classe que identifica o objeto e fornece uma probabilidade de que a classe em questão seja realmente a classe do objeto (KING, 2022).

No trabalho de RATH, 2021, é implementado um algoritmo que identifica tipos de carros. Na Figura 3 vemos o resultado da detecção, os *bounding boxes* fornecem a localização dos veículos e a classe a que pertencem, como por exemplo *car* ou *truck* e um número entre 0 e 1, que significa a probabilidade de pertencer a classe.

Figura 3 detecção de tipos de veiculos em imagem

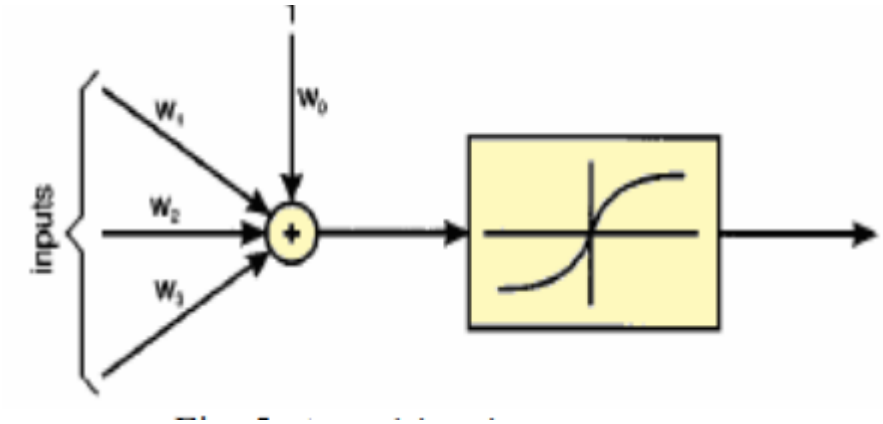
Fonte: RATH, 2021.

2.2 REDES NEURAIS

As raízes teóricas da visão computacional estão na teoria dos *multilayer perceptrons*. Um *perceptron* é uma abstração de um neurônio humano. A abstração matemática do *perceptron* é a de um grafo, as conecções são arestas que contêm pesos que multiplicam os valores que passam por elas. Os valores que chegam são então acumulados e na saída do vértice é aplicada uma função que modifica os valores (MCCULLOCH, PITTS, 1942).

Na Figura 4 temos a ilustração de um *perceptron*. Os valores que chegam, ou *inputs*, são multiplicados pelos pesos w e em seguida acumulados. Na saída o valor passa por uma função de ativação. Na figura 4 a função é chamada sigmoide. Esta função transforma valores reais em valores também reais entre 0 e 1 (HAYKIN, 2006).

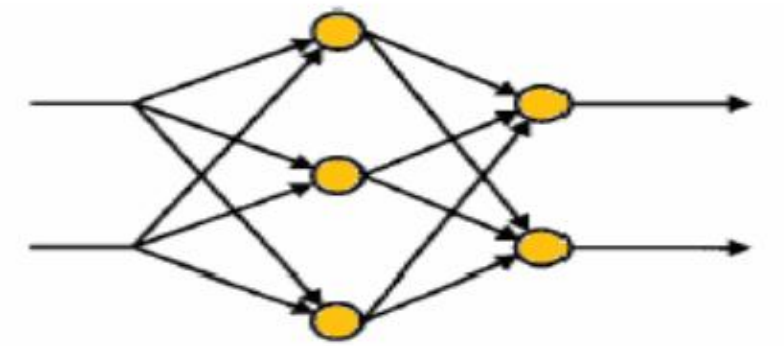
Figura 4 Representação de um perceptron



Fonte: POPESCU et al, 2009.

Os *perceptrons* podem ser organizados em redes. Os valores são processados por camadas sucessivas e após treinamento podem aprender a compreender padrões complexos. Essas redes são chamadas de *feedfoward networks* pelo fluxo dos dados que entram na primeira camada e vão sendo processados camada após camada (POPESCU *et al*, 2009). A Figura 5 mostra um exemplo de vários perceptrons organizados em uma rede *feedfoward*.

Figura 5 Representação de uma rede de perceptrons



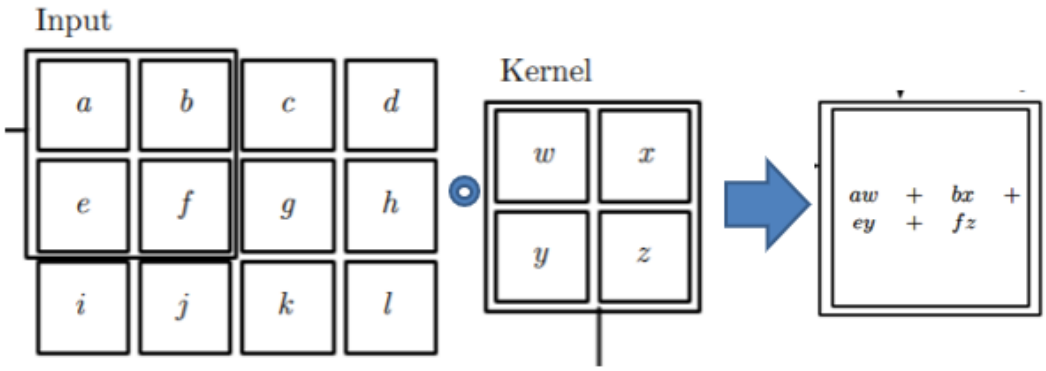
Fonte: POPESCU et al, 2009.

As redes de *perceptrons* podem ser organizadas em termos de suas operações e ligações entre os *perceptronsI*. Redes de perceptrons também são conhecidas como redes neurais. Alguns tipos são as *convolutional neural networks* (IZADYYAZDANABADIA *et al*, 2018), *Recurrent Neural Networks* (PEREIRA, 2020) entre outros. O campo da visão computacional vai se utilizar dessas redes para construir modelos de aprendizado.

Redes neurais convolucionais são assim conhecidas porque utilizam a operação de convolução. Uma operação de convolução é uma operação entre duas matrizes, a matriz entrada ou *input* e a matriz *kernel* ou filtro. O kernel desliza sobre a matriz de entrada e faz uma multiplicação ponto a ponto e soma os valores multiplicados. O resultado se torna uma entrada na matriz de saída (ANTON, RORRES, 2012).

Na Figura 6 temos um esquema que exemplifica a operação de convolução.

Figura 6 Operação de convolução

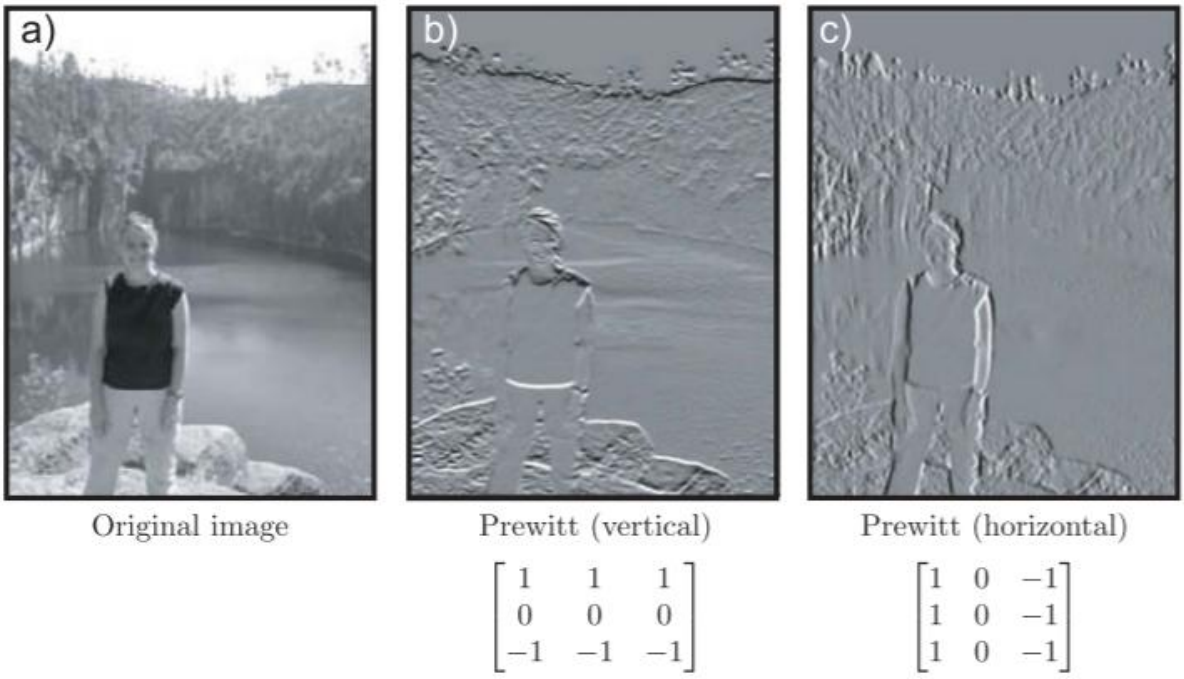


Fonte: adaptado de GOODFELLOW 2016.

Operações de convolução podem ser usadas para extrair características de uma imagem. Por exemplo na Figura 7, após passar a imagem a) por um kernel de convolução, b) destaca as bordas horizontais da imagem enquanto o kernel em c)

destaca as bordas verticais.

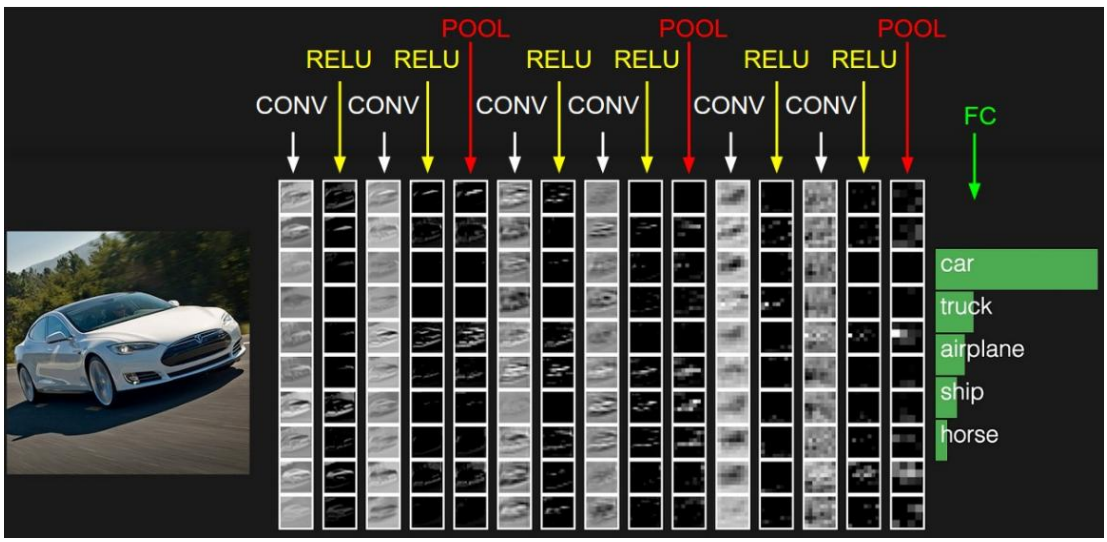
Figura 7 Convolução aplicada em imagem



Fonte: Prince 2012

Uma rede neural convolucional recebe como entrada uma imagem e aplica várias operações de convolução em suas camadas. Ao final a saída da rede pode identificar a classe da imagem fornecida (cs231N... , 2022). A Figura 8 exemplifica esse processo.

Figura 8 Imagem sendo processada por rede convolucional



Fonte: cs231N... , 2022.

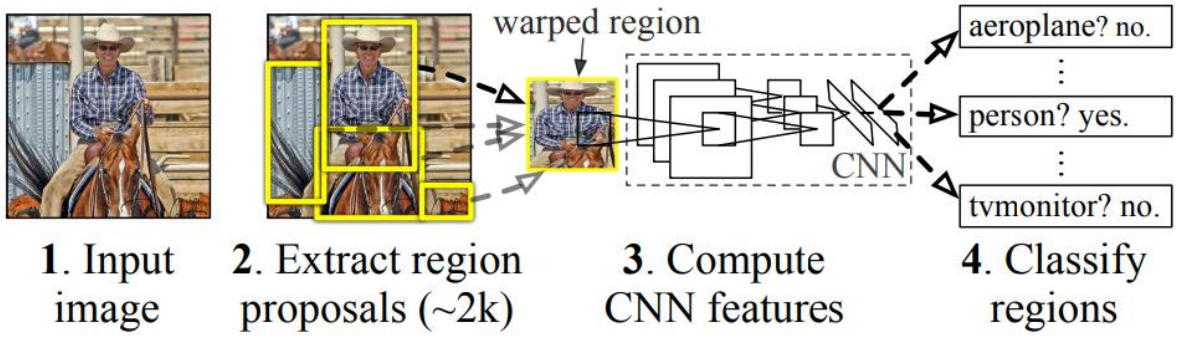
2.3 DETECÇÃO

A tarefa de detecção se trata de identificar as classes e a posição de objetos em uma cena. Alguns detectores passam por dois estágios, outros por apenas um estágio.

Um detector de dois estágios é o *Region based convolution neural network* (RCNN). Em GIRSHICK, 2014, é fornecida as etapas de execução desse detector. A primeira etapa é a de fornecimento da imagem, a imagem passa por uma etapa em que são eleitas regiões que podem conter objetos a serem classificados. As regiões propostas passam por uma rede convolucional na terceira etapa e por fim na 4 etapa é fornecida a classificação para a região.

A Figura 9 ilustra o funcionamento de um detector RCNN.

Figura 9 Passos de execução de uma RCNN

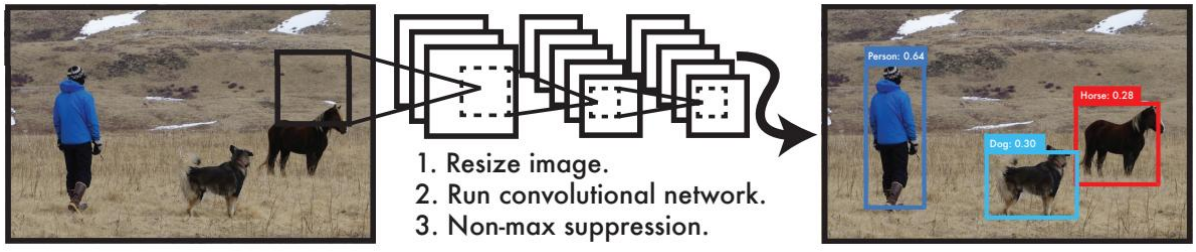


Fonte: GIRSHICK, 2014.

Outro tipo de detector é o detector que não passa pela etapa de propor regiões. O detector You Only Look Once, ou YOLO, tem um funcionamento diferente. Nele a imagem entra direto na rede convolucional e a saída é a previsão de todos os *bounding boxes* de objetos contidos na imagem. O detector YOLO não precisa passar por várias áreas da imagem várias vezes, por isso o nome *you only look once*, ou você so olha uma vez (REDMON, 2016).

Na Figura 10 temos uma ilustração do funcionamento de um detector YOLO.

Figura 10 Passos de execução de um modelo de detecção YOLO



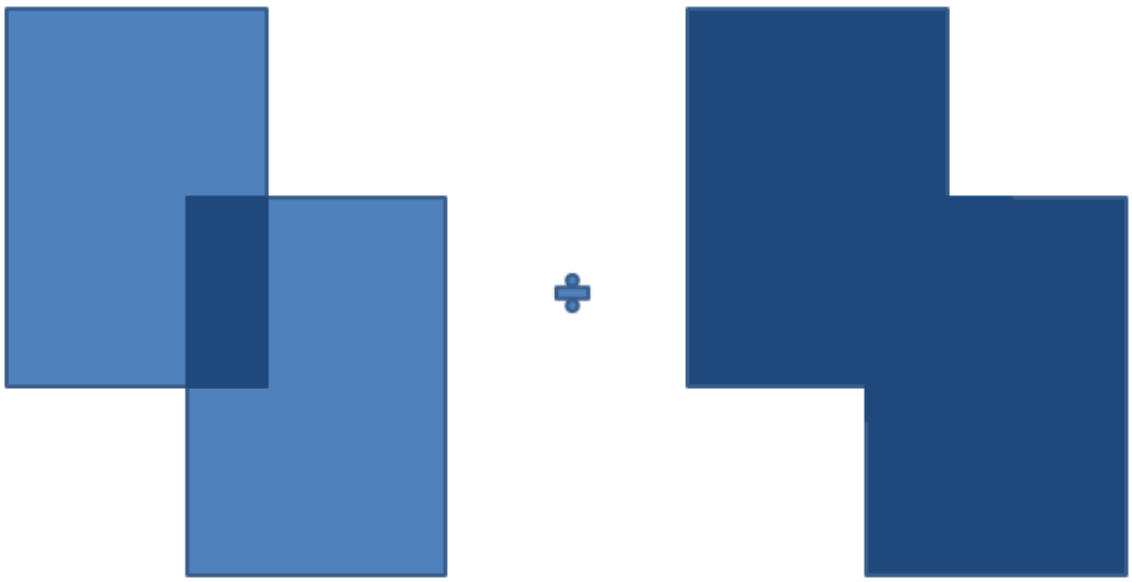
Fonte: REDMON, 2016.

Um conceito importante para a detecção de objetos é o conceito de Intersecção

pela união, também conhecido como IOU. A divisão da área da intersecção de dois *bounding box* pela área da união desses dois *bounding boxes* resulta em um número entre 0 e 1. Esse número é usado para interpretar se um *bounding box* esta sobre outro. Se o valor for 0 significa que os bounding boxes não se intersectam em nenhum local, se for 1 significa que estão completamente um sobre o outro (MONTGOMERY, 2015).

Na Figura 11 temos uma amostra visual de como seria o IOU.

Figura 11 Exemplo visual de IOU



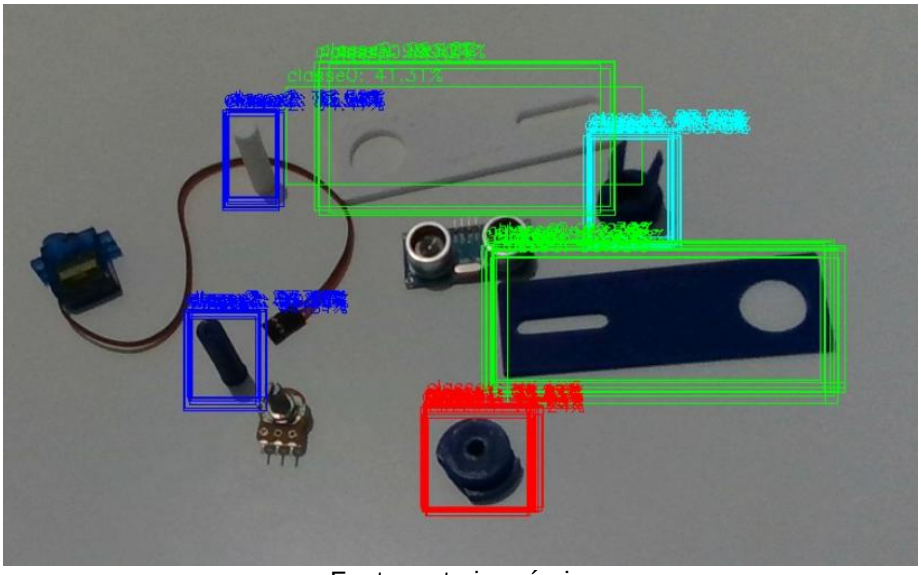
Fonte: autoria própria

O conceito de IOU é importante para a etapa final do detector YOLO. Por vezes um detector YOLO identifica mais de um *bounding box* para o mesmo objeto. É preciso remover bounding boxes repetidos para a o mesmo objeto na imagem. Para isso é usada uma técnica conhecida como Non maximum suppression, ou NMS (REDMON, 2016).

O NMS consiste em eleger para o objeto detectado o bounding box com a maior probabilidade de acerto. Eleito, esse *bounding box* é comparado com os outros que classificam o mesmo objeto. O algoritmo sabe qual deles classifica o mesmo objeto utilizando a técnica do IOU. Ao comparar um *bounding box* com o de maior probabilidade, se o IOU for maior que um determinado valor, o *bounding box* de menor probabilidade é removido (XU, HUANG, 2015).

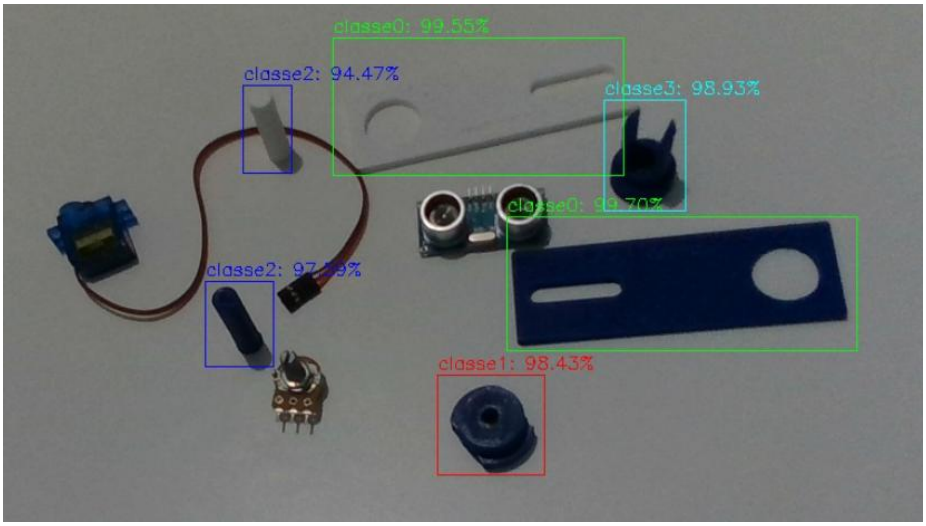
Na Figura 12 temos uma detecção feita pelo YOLO sem usar o NMS e na figura 13 com o NMS.

Figura 12 Detecção sem aplicação de NMS



Fonte: autoria própria

Figura 13 Detecção com aplicação do NMS



Fonte: autoria própria

3 PROCEDIMENTOS METODOLÓGICOS

Nesta seção serão apresentadas as ferramentas utilizadas.

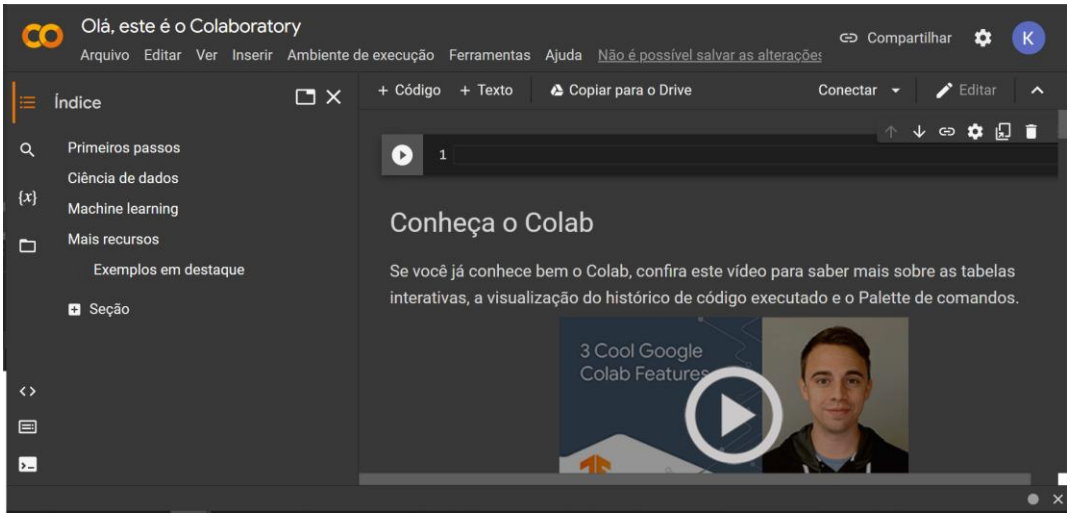
A construção do modelo de detecção YOLO foi feita na plataforma colab. A plataforma colab é uma máquina virtual disponibilizada pelo google através de uma interface web. A máquina fornecida continha 8GB de memória ram e o processador core i 7. O colab fornece tambem uma GPU, no treinamento do modelo ele forneceu

uma GPU Nvidia T4. A plataforma colab na sua versão gratuita oferece até 12 horas de execução continua (GOOGLE, 2022).

Na Figura 14 temos a interface do google colab.

.

Figura 14 Interface da ferramenta colab



Fonte: colab.

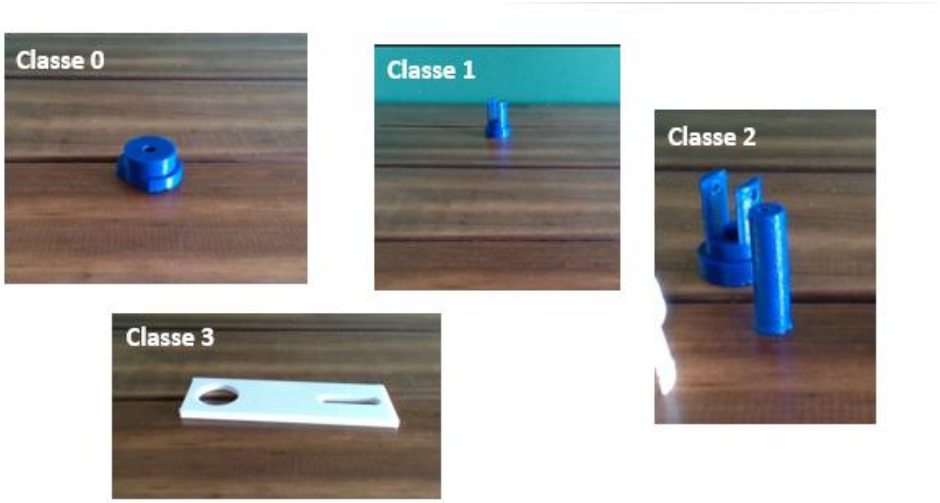
Além do colab foram utilizadas a linguagem de programação python, e o *framework dark net* que implementa o algoritmo YOLO.

O *framework darknet* foi escrito na linguagem C e não necessita que seja escrito código. Para utilizar o framework é necessário compilar os arquivos disponíveis em (CHET REDMON, 2022). No programa compilado basta fornecer os locais dos arquivos de imagens de treino e de teste.

Foi utilizado para o treinamento o *dataset* fornecido pelo grupo de mestrado em engenharia de produção da UFG de Aparecida de Goiânia.

O dataset consiste em 4 classes de peças impressas por impressão 3d. Na Figura 15 temos as 4 classes de peças.

Figura 15 Peças impressas em impressão 3D e suas respectivas



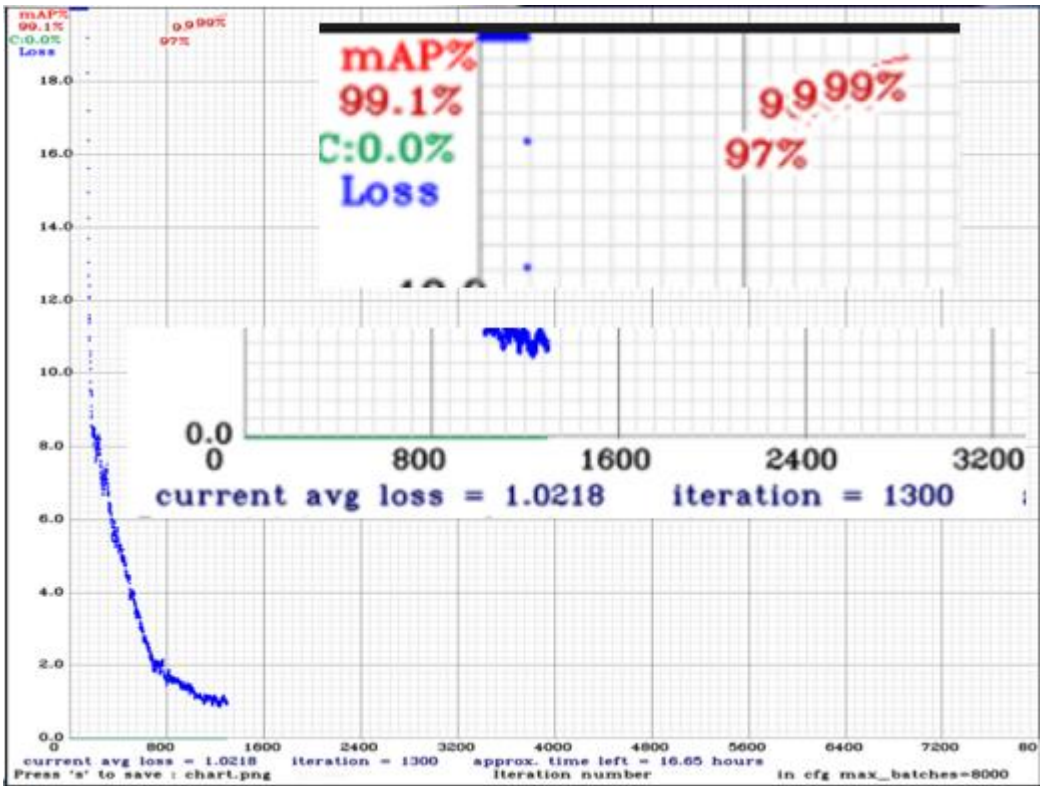
Fonte: grupo de mestrado em engenharia de produção da UFG de Aparecida de Goiânia.

4 RESULTADOS

O resultado foi que após cerca de 8 horas o colab parou de rodar por expiração do tempo da versão gratuita. Nesse tempo foi possível alcançar uma precisão média de noventa e nove virgula um por cento.

Na Figura 16 é possível ver o gráfico gerado durante o treinamento.

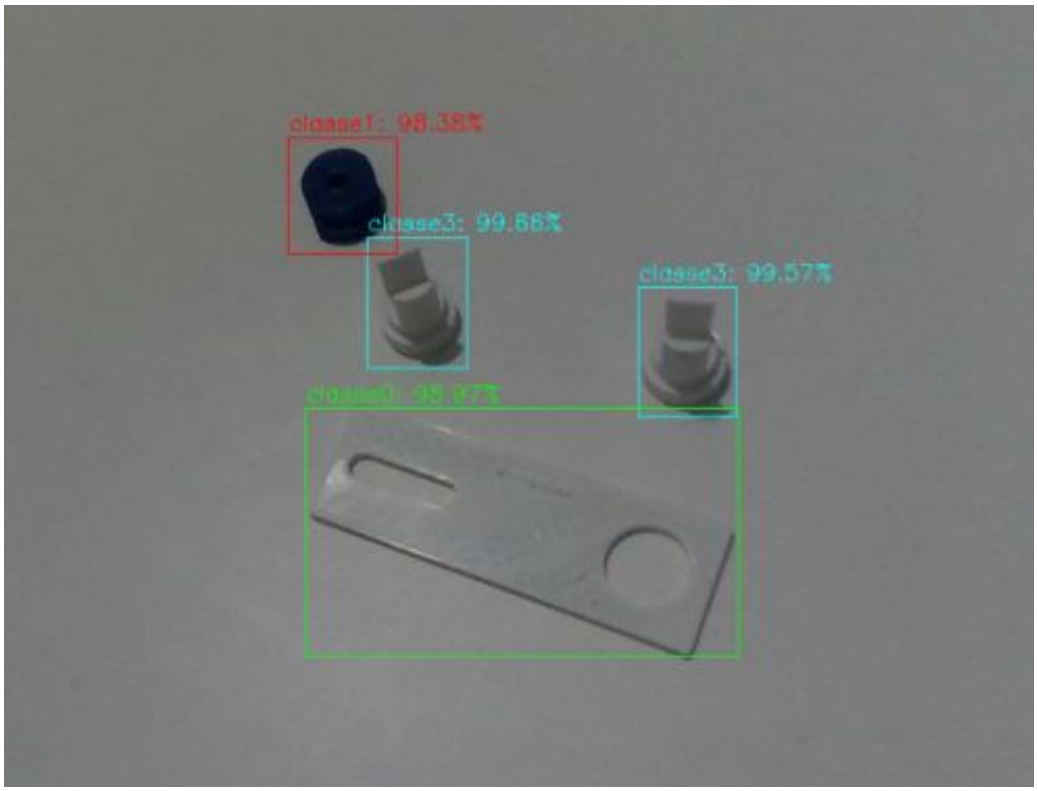
Figura 16 Grafico Evolução do treinamento.



Fonte: autoria própria.

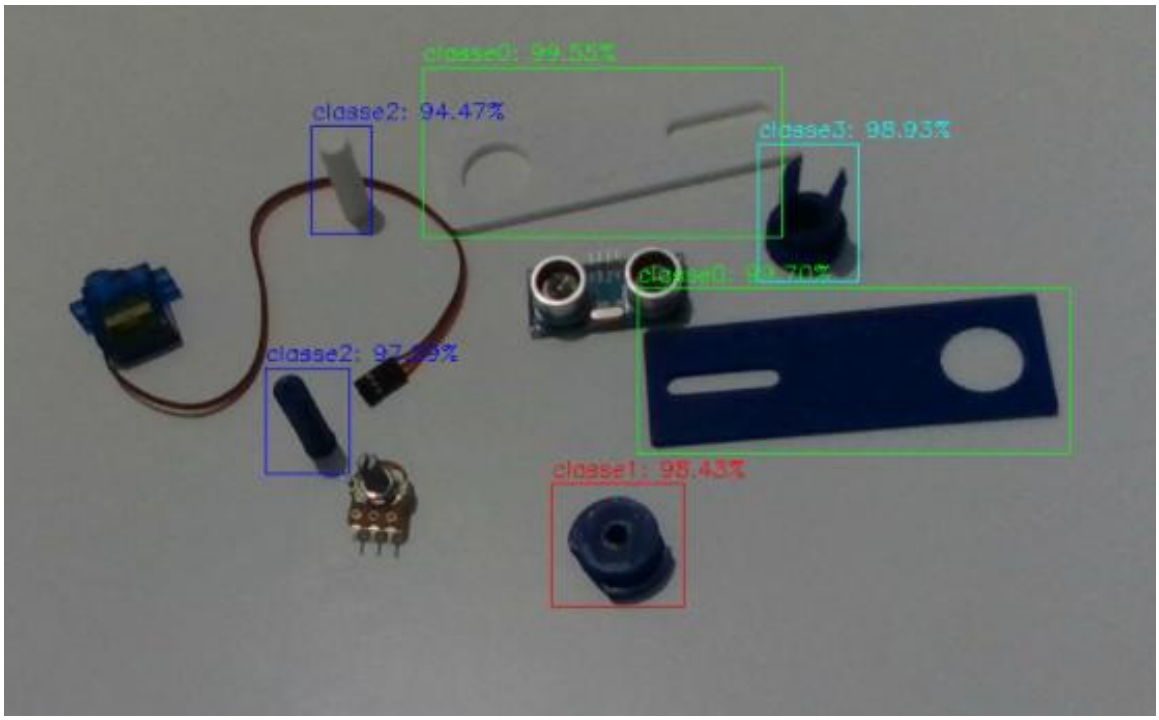
Abaixo estão na Figura 17 e 18 temos o resultado de imagens passadas pelo detector com os pesos treinados.

Figura 17 Imagem um processada pelo modelo de detecção com os pesos treinados



Fonte: autoria própria.

Figura 18 Imagem dois processada pelo modelo de detecção com os pesos treinados.



Fonte: autoria própria.

A placa Nvidia T4 fornecida pelo colab se mostrou eficaz no treinamento do modelo. Uma vantagem da plataforma é que essa placa foi fornecida de forma gratuita. Seu custo de mercado é bastante elevado, logo o colab torna bem mais acessível trabalhar com essa placa. Na Figura 19 temos o valor da placa de vídeo oferecida em

(DELL, 2022).

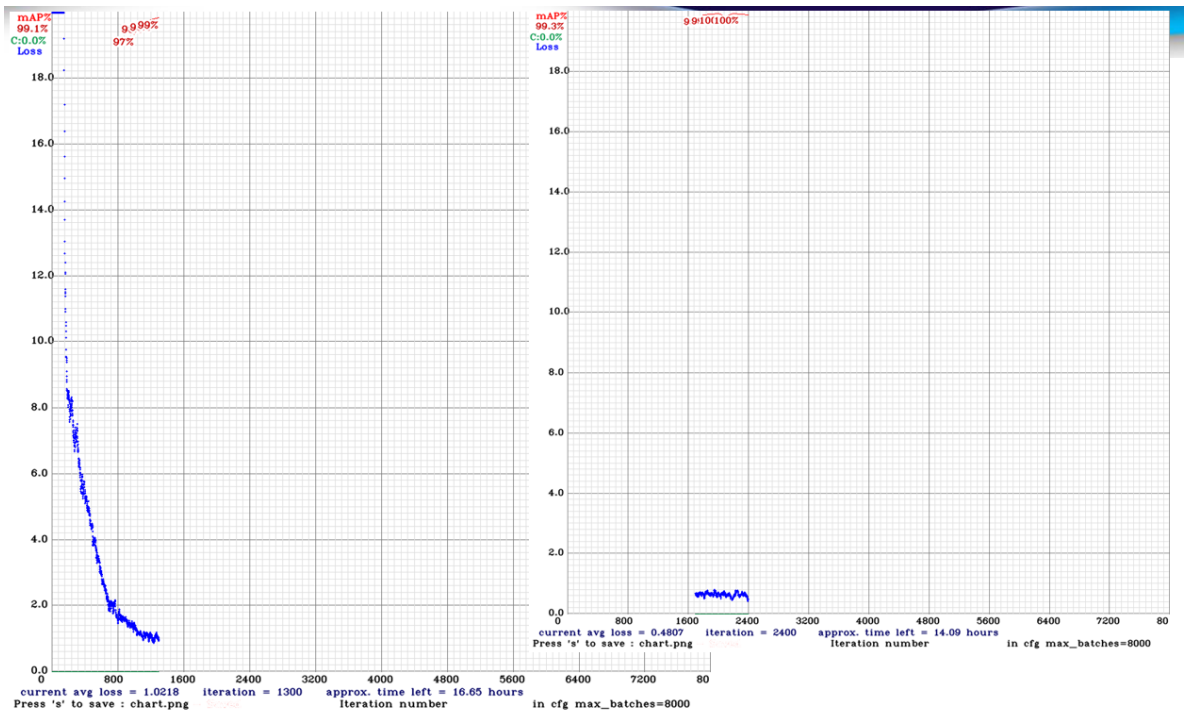
Figura 19 Preço da GPU Nvidia T4.

Fonte: DELL, 2022

Outra vantagem da plataforma colab é a possibilidade de treinar modelos que exigem mais tempo. Mesmo a plataforma parando a execução após 8h, foi possível retornar o treinamento após 2 dias, do ponto em que ela tinha parado a execução.

Na Figura 20 temos a execução sendo interrompida após 8h e depois a execução a partir do ponto em que os pesos tinham sido treinados. Na imagem é possível notar que não houve perda no progresso do treinamento.

Figura 20 Grafico reiniciação do treinamento apos parada.



Fonte: DELL, 2022

Outra vantagem da plataforma é a curva de aprendizado. Durante os

procedimentos para criação e treinamento do modelo foram usados conhecimentos em python, Linux e makefiles. São poucos conhecimentos comparado com plataformas como o vertex AI que utiliza Docker e uma CLI própria de comandos. Alem disso não foi necessária nenhuma configuração de ambiente de execução, como download de drivers.

5 CONSIDERAÇÕES FINAIS

A necessidade de hardware especializado e a curva de aprendizado necessária para criação de modelos de detecção podem tornar inacessível ou dificultar a criação do modelo. Neste trabalho foi visto como a plataforma do colab contribui positivamente para a superação de desafios no processo de construção de um sistema de detecção, seja por sua simplicidade no uso ou pela infraestrutura adequada e gratuita.

Também foi mostrado o desempenho de algoritmos como o YOLO, que conseguiu uma acurácia média de 99,1% após o treinamento. Os artefatos produzidos para este trabalho estão disponíveis em drive, 2022. Esses artefatos podem ser usados no futuro como material de referência para trabalhos que tenham detecção usando o modelo YOLO em seu escopo.

Em Hatab, 2020 é apresentado um modelo de detecção de defeitos utilizando o algoritmo YOLO. Como trabalho futuro podem ser usados os parâmetros de treinamento e o colab desse projeto em um novo *dataset* de defeitos.

REFERÊNCIAS

RAMOS, André. **Fisiologia da Visão: Um estudo sobre o 'ver' e o 'enxergar'**. Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, 2006. Disponível em: <https://sites.unifoa.edu.br/portal/plano_aula/arquivos/04054/Fisiologia%20da%20visao%20-%20MODULO%20I.pdf>. Acesso em: 06 de junho de 2022.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. Edição. Ithaca: Springer, 2011. 812p.

MASSON, Juliano Emir Nunes. **Geração de mapas de profundidade utilizando redes neurais convolucionais**. 2021. Dissertação (Mestrado em Engenharia de Automação e Sistemas)- Universidade Federal de Santa Catarina, Florianópolis, 2021.

Zhao, Baojun, Hongshuo Wang, Linbo Tang, and Yuqi Han. **Towards Long-term UAV Object Tracking via Effective Feature Matching**. Electronics Letters 56.20 (2020): 1056-059. Web.

RATH, Sovit. **Custom Object Detection Training using YOLOv5**. 2022. Disponível em: <https://learnopencv.com/custom-object-detection-training-using-yolov5/>. Acesso em: 19 de maio de 2022.

cs231n stanford.edu. **CS231n Convolutional Neural Networks for Visual Recognition**. cs231n stanford.edu, 2022. Disponível em: <https://cs231n.github.io/convolutional-networks/>. Acesso em: 19 de maio de 2022. Drive 2022. Disponivel em: <https://drive.google.com/drive/folders/1Y7Fez4UkO0BCbRcX5ReAZE4QpMSb8Bw8?usp=sharing >. Acesso em: 16 de junho de 2022.

Hatab, M., Malekmohamadi, H., Amira, **A. Surface Defect Detection Using YOLO**. 2020. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-55180-3_37> . Acesso em: 19 de maio de 2022.

GIRSHICK, Ross et al. **Rich feature hierarchies for accurate object detection and semantic segmentation** Tech report (v5). 2014. Disponível em: <https://arxiv.org/abs/1311.2524>. Acesso em: 19 de maio de 2022.

REDMON, Joseph, et al. **You Only Look Once: Unified, Real-Time Object Detection**. University of Washington, 2016. Disponível em: <https://arxiv.org/abs/1506.02640>. Acesso em: 19 de maio de 2022.

BARELLI, Felipe. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV**. Editora Casa do Código, v. 3, f. 128, 2018. 256 p.

BACCIU, Davide; LISBOA, Paulo J G; VELLIDO, Alfredo. **Deep Learning In Biology And Medicine**. World Scientific, v. 3, f. 166, 2022. 332 p.

BEHERE, Sagar; TÖRNGREN, Martin. **A functional architecture for autonomous driving**. 2015. Disponível em: https://www.academia.edu/34898174/A_functional_architecture_for_autonomous_driving. Acesso em: 18 abr. 2020.

MILANO, Danilo; BARROZO, Honorato Luciano. **VISÃO COMPUTACIONAL**. Academia. Limeira, 2022. Disponível em: https://www.academia.edu/9621896/VIS%C3%83O_COMPUTACIONAL_Palavras-Chaves. Acesso em: 12 mai. 2022.

KING, Davis. **Easily Create High Quality Object Detectors with Deep Learning**. 2016. Disponível em: http://blog.dlib.net/2016/10/easily-create-high-quality-object.html. Acesso em: 25 mai. 2022.

MCCULLOCH, Warren S.; PITTS, Walter. **A Logical Calculus of the Ideas Immanent in Nervous Activity**, f. 12. 1942. 23 p.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. Bookman Editora, v. 1, f. 449, 2006. 898 p.

POPESCU, Marius-Constantin et al. **Multilayer Perceptron and Neural Networks**. WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, v. 8, jul 2009.

IZADYYAZDANABADIA, Mohammadhassan et al. **Convolutional Neural Networks: Ensemble Modeling, Fine-Tuning and Unsupervised Semantic Localization for Intraoperative CLE Images**. Journal of Visual Communication and Image Representation, Arizona, 2018.

PEREIRA, Basilio de Braganca; RAO, Calyampudi Radhakrishna; OLIVEIRA, Fabio Borges de. **Statistical Learning Using Neural Networks: A Guide for Statisticians and Data Scientists with Python**. CRC Press, v. 3, f. 117, 2020. 234 p.

ANTON, Howard; RORRES, Chris. **Álgebra Linear com Aplicações** - 10. ed.. Bookman Editora, f. 393, 2012. 786 p.

MONTGOMERY, Douglas C.. **Estatística aplicada e probabilidade para engenheiros** (6a. ed.)., f. 328. 2015. 655 p.

XU, Guangyuan; HUANG, Shuangxi. **A Novel Non-maximum Suppression Algorithm Based on Affinity Propagation Clustering**. In: 4TH IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL CYBER-PHYSICAL SYSTEMS (ICPS), n. 4. 2021. Anais eletrônicos [...]. 270,276 p. Disponível em: https://ieeexplore.ieee.org/document/9468162. Acesso em: 26 mai. 2022.

CHET REDMON, Joseph. Darknet: **Open Source Neural Networks in C.** pjreddie. Disponível em: https://pjreddie.com/darknet/. Acesso em: 7 mar. 2022.

GOOGLE. Olá, este é o Colaboratory. Disponível em: https://colab.research.google.com/notebooks/intro.ipynb?hl=pt_BR. Acesso em: 1 mar. 2022.

DELL. **NVIDIA® Tesla® T4 16GB Passiva, solteiro Slot, perfil baixo GPU Customer Install**. Disponível em: https://www.dell.com/pt-br/shop/nvidia-tesla-t4-16gb-passiva-solteiro-slot-perfil-baixo-gpu-customer-install/apd/490-bflb/pe%C3%A7as-baterias-e-upgrades?gacd=9690632-15009-5761040-276815260-0&dgc=ST&cid=71700000069469182&gclid=CjwKCAjwqauVBhBGEiwAXOepkeUVTRTN6a3TX8f9M5Q0L8q7dA0r9QKQ6nSsdrpp-2M-W8i730KpexoCw6sQAvD_BwE&gclsrc=aw.ds&nclid=DKvtW4Vj_4vLOYZW3wQpheWDQNBHUbxvbXpPiu1ox1x63yw_w1oxpy0If4NNTkXR. Acesso em: 11 Maio 2022.