CURSO DE SISTEMAS DE INFORMAÇÃO

COMPARAÇÃO DE REDES DE DETECÇÃO DE OBJETOS PARA APLICAÇÃO E RECONHECIMENTO DE ESPAÇOS PARA EVENTOS

PATRÍCIA AIRES CORADO

Palmas

2022

CURSO DE SISTEMAS DE INFORMAÇÃO

COMPARAÇÃO DE REDES DE DETECÇÃO DE OBJETOS PARA APLICAÇÃO E RECONHECIMENTO DE ESPAÇOS PARA EVENTOS

PATRÍCIA AIRES CORADO

Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Marco Antônio Firmino de Sousa.

Palmas

2022

Dados Internacionais de Catalogação na Publicação (CIP) Sistema de Bibliotecas da Universidade Estadual do Tocantins

C787c

CORADO, Patrícia Aires
COMPARAÇÃO DE REDES DE DETECÇÃO DE OBJETOS PARA APLICAÇÃO E RECONHECIMENTO DE ESPAÇOS PARA EVENTOS. Patrícia Aires Corado. - Palmas, TO, 2022

Monografia Graduação - Universidade Estadual do Tocantins – Câmpus Universitário de Palmas - Curso de Sistemas de Informação, 2022.

Orientador: Marco Antonio Firmino de Sousa

1. Processamento de Imagens. 2. Convolutional Neural Network. 3. Detecção de objetos. 4. Classificação de imagens.

CDD 610.7

Elaborado pelo sistema de geração automatica de ficha catalográfica da UNITINS com os dados fornecidos pelo(a) autor(a).

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA FUNDAÇÃO UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS

Aos **27** dias do mês de **Junho** de **2022**, reuniu-se na Fundação Universidade Estadual do Tocantins, Câmpus Palmas, Bloco B, às **19:30 horas**, sob a Coordenação do Professor **Marco Antônio Firmino de Sousa**, a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Marco Antônio Firmino de Sousa** (Orientador), Professor **Douglas Chagas da Silva** e Professora **Tamirys Virgulino Ribeiro Prado**, para avaliação da defesa do trabalho intitulado **"Comparação de Redes de Detecção de Objetos para Aplicação e Reconhecimento de Espaços para Eventos"** da acadêmica **Patrícia Aires Corado** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso (TCC). Após exposição do trabalho realizado pelo acadêmica e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: 9,0.

Sendo, portanto, o Acadêmico: (X) Aprovado ( ) Reprovado

Assinam esta Ata: 27/06/2022

Professor Orientador: Marco Antonio Firmino de Sousa

Examinador: Douglas Chagas da Silva

Examinador: Tamirys Virgulino Ribeiro Prado

Prof. Marco Antonio Firmino de Sousa
Presidente da Banca Examinadora

CURSO DE SISTEMAS DE INFORMAÇÃO

COMPARAÇÃO DE REDES DE DETECÇÃO DE OBJETOS PARA APLICAÇÃO E RECONHECIMENTO DE ESPAÇOS PARA EVENTOS

PATRÍCIA AIRES CORADO

Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Marco Antônio Firmino de Sousa.

Prof. Me. Marco Antônio Firmino de Sousa
Orientador

Prof. Me. Douglas Chagas da Silva
Convidado 1

Profa. Ma. Tamirys Virgulino Ribeiro Prado
Convidado 2

Palmas
2022

Este trabalho é dedicado à minha família, pelo apoio incondicional.

Agradecimentos

Dedico este espaço à minha mãe, irmã e namorado, que permaneceram ao meu lado nos momentos em que eu acreditava não merecer estar ao lado de ninguém, agradeço o apoio e afeto que a mim dedicaram, pois são eles que mantêm seguindo em frente.

O sucesso é a soma de pequenos esforços repetidos dia após dia.
(Robert Collier)

Resumo

O presente trabalho tem como objetivo contextualizar o leitor sobre a disciplina de Processamento de Imagens, assim como apresentar uma trajetória dos últimos cinco anos das *Convolutional Neural Network* - CNN, que realizam as etapas de processamento de imagens referentes à detecção de objetos e classificação de imagens. O trabalho aplica algumas das redes em contexto próprio, com a finalidade de identificação da rede que apresenta melhor desempenho para utilização em aplicação que faz reconhecimento de espaços para eventos, esta operando com *dataset* próprio contendo imagens recolhidas de redes sociais.

Palavras-chaves: Processamento de Imagens, *Convolutional Neural Network*, detecção de objetos e classificação de imagens.

Abstract

The present work aims to contextualize the reader about the discipline of Image Processing, as well as presenting a trajectory of the last five years of the Convolutional Neural Network - CNN, which perform the steps of processing images related to object detection and image classification. The work applies some of the networks in their own context, with the purpose of identifying of the network that presents better performance for use in an application that recognizes cement spaces for events, is operating with its own dataset containing images collected from social networks

Key-words: Image Processing, Convolutional Neural Network, Object Detection and Image Classification.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

De acordo Ghosal et al. (2018) das técnicas de aprendizado de máquina na área da inteligência artificial que encontra-se em constante aperfeiçoamento tem o processamento de imagens que utilizam as *Convolutional Neural Network*, sua finalidade é a manipulação de imagens para extração de características e reconhecimento de objetos presentes.

Por meio de técnicas e ferramentas, o processamento de imagens apresenta-se como uma alternativa para o reconhecimento de imagens complexas. Atualmente é possível reconhecer padrões de objetos por meio de instâncias de imagens, onde é aplicado algorítimos de classificação para reconhecimento de objetos.

A detecção de objetos é uma das etapas do processamento de imagem onde faz-se a delimitação das áreas e identificação da instância do objeto na imagem, trata-se de uma parte fundamental para aplicações que objetivam processar imagem com a finalidade de classificá-la.

A detecção de objetos trata da detecção de instâncias de objetos semânticos de uma determinada classe (como humanos, móveis, animais ou carros) em imagens e vídeos digitais (SANTANA, 2022).

Uma das formas existentes para implementação da detecção de objetos é por meio do uso de *Convolutional Neural Network* - CNN, tal método proporcionou diversas competições em diversas ramificações de processamento de imagens, entre elas a detecção de objetos, o que gerou diversidade nas CNNs.

Mas não existiria processamento de imagens sem as imagens, desta forma, com o advento das redes sociais uma forma de interação entre usuários é por meio do compartilhamento de imagens, o que podem ser de carater privado ou público. Com foco nas imagens públicas, as quais podem ser recolhidas para utilização em aplicações de reconhecimento, auxiliando assim no desenvolvimento de diversos projetos sobre processamento de imagens e afins.

O presente trabalho apresenta uma contextualização sobre a temática de processamento de imagens, sob a ótica das CNNs para detecção de objetos e classificação, assim como uma experimentação realizada com a aplicação do *dataset* MSRA-B em três modelos de CNNs, *Yolo*v4, *Mask*-RCNN e *Cascade Mask*-RCNN, análise de desempenho dos treinamento com a utilização de Matriz de Confusão para captura da precisão, *recall*, f1-*score* e acurácia.

1.1 Justificativa

Utilizando a Inteligência Artificial e suas ramificações como base, podemos implementar as mais variadas aplicações. Muitos programas utilizam o Processamento de Imagens, como por exemplo o desenvolvido por Réos e Farias (2019), o qual é uma ferramenta que auxilia no fluxo de trânsito ao indicar vagas de estacionamento disponíveis em ambiente simulado.

O Processamento de Imagens faz uso da Inteligência Artifical para aprimoramento do processo de reconhecimento. Alguns exemplos de aplicação são os no meio médico, como demonstram Kaufman e Santaella (2020) que mencionam o IBM Watson, um sistema que é capaz de processar grandes volumes de dados, estabelecendo correlações entre sintomas e/ou imagens em uma dimensão impossível de ser alcançada por um ser humano.

O reconhecimento facial também é um contexto bastante popular, Monteiro (2020) propôs uma ferramenta para meio Educacional, podendo então substituir o método padrão de realização de chamadas em classes de aula, e adequando para que o reconhecimento facial possa realizar este processo.

As Redes Sociais além um de meio de entretenimento, é um ambiente rico em dados públicos que podem ser utilizados em processos de *Machine Learning*, entre eles o Reconhecimento de Imagens.

1.2 Problema

Ao realizarmos uma pesquisa, no google por exemplo, sobre um local, podemos encontrar dados como horário de funcionamento, endereço e contato, na maioria das vezes deve-se entrar em contato com os administradores para levantar dados como, lotação, cardápio, se existe ou não espaço para crianças etc, fazer o detalhamento da propriedade e o que nela existe trata-se de uma ação humana sem necessidade de computação.

Uma das formas de verificação sem a necessidade de contato direto com o estabelecimento é por meio de conversação com pessoas que já estiveram no local ou por visualização em imagens ou vídeos, conteúdo abundante em plataformas de Rede Social como o *Instagram* ou *Snapchat*. Entretanto, não se torna muito fácil e cômodo realizar uma pesquisa acerca de um estabelecimento utilizando a verbalização ou levantamento de dados sem uso de máquinas.

Como antes mencionado, o uso de plataformas como Redes Sociais como maneira de se obter dados para análise, seja de imagens ou textos, tornou-se popular justamente pela quantidade de conteúdo disponível e interação que o usuário dispõe.

O intuito deste trabalho é extrair características de locais geralmente utilizados para eventos, utilizando as Redes Sociais como o *Instagram* para coleta de dados de imagens, e utilizando *Deep Learning*, ou aprendizado profundo, para treinamento de uma Inteligência Artificial que reconheça padrões de objetos na imagem e as especifiquem.

1.3 Objetivos

Avaliar modelos de aprendizado de máquina voltados para extração de características a fim de identificar ambientes utilizados em eventos sociais a partir de processamento de imagens.

1.3.1 Objetivo Geral

Estruturar um sistema que caracterize um espaço com base em fotos.

1.3.2 Objetivos Específicos

- Comparar as redes Yolov4, Mask R-CNN e Cascade Mask R-CNN;
- Analisar imagens em busca de objetos utilizando as redes;
- Gerar um relatório com matriz de confusão com resultado dos treinamentos.

2 Referencial Teórico

2.1 Trabalhos Relacionados

A detecção de objetos é uma parte essencial no contexto deste projeto, por meio deste que ocorre os processos de identificação dos objetos e a separação do fundo da imagem dos objetos, este processo pode ser encontrado em diversos trabalhos como no caso de Perez e Junior (2020), que com o objetivo de construir uma base de dados sintética para a equipe RoboFEI@Home, faz uso de uma *Convolutional Neural Network* - CNN para a detecção de objetos.

Os autores dividiram o projeto em duas etapas, uma para extrair de forma automática o fundo das imagens de amostras dos objetos utilizando *Deep Salient Object Detection* e a segunda referente ao desenvolvimento de um método para usar estas imagens e suas respectivas máscaras na criação de novas imagens. Para desenvolvimento da primeira etapa, utiliza-se Detecção de Objetos Proeminentes devido aos bons resultados apresentados e realiza testes com algoritmos de visão computacional entretanto, os resultados não apresentaram-se promissores, então os autores utilizam o *Deep Salient Object Detection*. É visualizado a necessidade de utilizar a detecção de saliências e por ser mais complicada que a detecção de bordas foi adicionado mais duas camadas convolutivas à rede. Para o proposto aplica-se a biblioteca *Keras*[1] do *Python*[2].

Emprega-se para treinamento MSRA-B, que consiste no primeiro conjunto de dados de "grande escala" na literatura para avaliação quantitativa de modelos de detecção de objetos salientes. Ele contém duas partes: MSRA-A consistindo de 20.840 imagens e MSRA-B contendo 5 mil imagens altamente inequívocas selecionadas do MSRA-A. Essas imagens cobrem uma grande variedade de cenários, como flores, frutas, animais, cenas internas e externas. Na anotação, cada imagem é redimensionada para ter um comprimento lateral máximo de 400 *pixels* e os objetos salientes são anotados manualmente por retângulos (3 assuntos para MSRA-A e 9 assuntos para MSRA-B). Com as máscaras geradas no processo de detecção, os autores desenvolvem um programa em *Python* que utiliza a imagem do objeto e sua máscara binária junto com as imagens de fundo para compor as imagens finais.

Tal projeto apresentou resultado significativo no cenário de competições robóticas, conseguindo gerar aproximadamente 3 mil exemplos de imagens em 4h com base em apenas 10 imagens, o que depende de 12 horas sem a utilização da geração sintética de imagens.

[1] https://keras.io/api/
[2] https://www.python.org/doc/

Li et al. (2017) propõem uma solução para o problema de detecção de objetos genéricos em pequena escala aplicando *Genarative Adversarial Network* - GAN, uma estrutura para aprendizagem de módulos generativos. O autor aponta a necessidade de introduzir um novo modelo e de um novo discriminador na GAN. Este modelo é condicionado nas características de baixo nível do pequeno objeto, no qual um gerador aprende a gerar uma representação residual por meio das representações de objetos grandes.

O discriminador possui dois ramos, um para diferenciar a representação gerada do original e outro para justificar a precisão da detecção. São formados dois subconjuntos com os tamanhos médios das instâncias de imagens, um com imagens pequenas e outro com imagens grandes, para o treinamento foi detalhado os parâmetros de fundo da camada convolucional e o ramo da rede discriminadora.

Sequencialmente, a rede é treinada por um ramo perceptivo para instruir a rede do gerador com base no subconjunto que contém imagens de objetos pequenos e o ramo adversário da rede discriminadora usando ambos os subconjuntos. O treinamento é continuado até que encontra-se o que os autores chamam de ponto de equilíbrio, ou objetos determinados como objetos grandes podem ser gerados para os objetos pequenos com alta precisão de detecção.

Os autores explica o gerador como uma rede de aprendizagem residual profunda que aumenta a representação de objetos para pequenos para super-resolvidos introduzindo detalhes mais refinados ausentes dos objetos pequenos através da aprendizagem residual.

Já a rede discriminadora além de ser um diferenciador entre objetos super-resolvidos, o objeto pequeno e o original, também justifica a precisão da detecção beneficiando a característica do objeto super-resolvido gerado.

Ao realizar dois experimentos utilizando estas redes, um com um *Dataset* de sinais de trânsitos conhecido como *Tsinghua-Tencent 100K* ele fornece 100 mil imagens contendo 30 mil instâncias de sinais de trânsito. Essas imagens cobrem grandes variações de iluminância e condições climáticas. Cada sinal de trânsito no *benchmark* é anotado com um rótulo de classe, sua caixa delimitadora e máscara de *pixel*. E outro *Dataset* de detecção de Pedestres conhecido como *Pedestrian Caltech benchmark* possuindo 250 mil quadros com um total de 350 mil caixas delimitadoras, os pedestres variam em aparência, pose e escala.

Por meio dos experimentos foi constatado que as GANs resolvem o problema de detecção de pequenos objetos, o gerador aprende uma representação residual dos detalhes refinados das camadas de nível inferior e aprimora as representações de objetos pequenos para se aproximar daqueles de objetos grandes, tentando enganar o discriminador que é treinado para diferenciar bem entre ambas as representações.

2.2 Aprendizado de Máquina

A referenciação de Inteligência Artificial para caracterização de alguns sistemas autônomos tem ganhado bastante ênfase na mídia, visualiza-se na explicação de sistemas como carros autônomos, reconhecimento facial e controle de jogos. Entretanto, percebe-se a superficialidade do tema quando não se leva em consideração as técnicas utilizadas para implementação de tais aplicações e funcionalidades.

O aprendizado de máquina trata-se de um campo da inteligência artificial restrita que trata do reconhecimento de padrões através de uma base de dados e posterior aplicação do aprendizado no reconhecimento das variáveis em outras unidades ou conjuntos de dados Oliveira (apud AVELINO; SILVEIRA; AMADEU, 2018). Parte de um processo, o aprendizado proporciona a classificação de um exemplo por meio do algoritmo de indução, objetivando identificar características ou atributos para então classificá-lo.

Sua indispensabilidade ampliou-se para uso convencional e atualmente, é possível identificar diversas tecnologias que participam do cotidiano do ser humano que fazem uso de algum algoritmo de aprendizagem, como nos *smartphones* e seu comando de voz, e em redes sociais, estas utilizam o fato de que os usuários postam uma quantidade elevada de fotos e usam tais informações como fonte de dados para o desenvolvimento das ferramentas aptas para processar fotos e analisá-las, o que permite que o campo da IA se desenvolva de maneira mais rápida e eficiente ao ter aliados como as redes sociais (FERNANDES et al., 2018).

O aprendizado de máquina tem como foco a criação de sistemas independentes, voltando-se para o desenvolvimento de Redes Neurais Artificiais (RNA) o qual treinam sob ótica de três tipos de aprendizado.

2.2.1 Aprendizado Supervisionado

Norvig e Russell (2014) apresenta esta forma conceituando-a como um agente (rede) que observa alguns exemplos de entrada e saída, e aprende com um método que faz a esquematização da entrada para a saída.

2.2.2 Aprendizado não Supervisionado

Nesta situação a rede aprende padrões na entrada, embora não seja fornecida nenhuma informação de entrada. A tarefa mais comum de aprendizagem não supervisionada é o agrupamento: a detecção de grupos de exemplos de entrada potencialmente úteis (NORVIG; RUSSELL, 2014).

2.2.3 Aprendizado por Reforço

A rede aprende a partir de uma série de reforços recompensas ou punições (NORVIG; RUSSELL, 2014).

2.2.4 Aprendizado Profundo

Além dos três tipos principais existe a abordagem do *Deep Learning* ou aprendizado profundo, que tem suas raízes nas redes neurais convencionais, neste é utilizado tecnologias gráficas com transformações entre neurônios para desenvolver modelos de aprendizagem de muitas camadas (POUYANFAR et al., 2018).

2.2.5 Aprendizado Semi-supervisionado

É o ramo do aprendizado de máquina preocupado com o uso de dados rotulados e não rotulados para executar determinadas tarefas de aprendizado. Este permite aproveitar as grandes quantidades de dados não rotulados disponíveis em muitos casos de uso em combinação com conjuntos normalmente menores de dados rotulados (ENGELEN; HOOS, 2020).

2.3 Redes Neurais Convolucionais

Convolutional Neural Network - CNN ou Redes Neurais Convolucionais - RNC, é uma Rede Neural que emprega aprendizado profundo e que utiliza a convolução como operação em pelo menos uma camada. CNN são um tipo de RNA especializado para o processamento de dados de entrada que possuem um formato de matriz, usualmente utilizadas para o processamento de informações visuais. Sobre a matriz Paz et al. (2020) informa:

> Sendo uma imagem uma matriz, em que cada *pixel* armazena a informação de uma cor relativa ao código RGB (*Red, Blue, Green*), usualmente, na primeira camada densa (input) cada elemento da matriz será uma entrada para cada neurônio. Enquanto a informação progride na arquitetura neural, as camadas convolucionais irão filtrar características de extração específicas a cada filtro (geralmente filtros são matrizes quadradas, 3 x 3), enquanto percorrem toda a imagem.

Embora ainda sejam empregadas em problemas com quantidade pequena de amostras rotuladas, as CNNs são bastante utilizadas, principalmente em problemas com bastante volume de dados de entrada, Oliveira e Camara (2019) citam algumas vantagens de aplicá-las:

- São redes capaz de extrair características consideradas relevantes através de aprendizado de transformações;
- Menor dependência de parâmetros de ajustes do que redes totalmente conectadas com a mesma quantidade de camadas ocultas.

Uma CNN pode ser dividida em duas partes, uma para identificação e extração das características e outra para classificação.

> A extração de características consiste em uma série de camadas convolucionais com filtros (*Kernels*) e camadas de *Pooling*, na qual é realizado a extração de características da imagem de entrada, reduzindo-a para uma forma mais fácil de processar sem perder características importantes para ter uma boa previsão. Ela recebe uma entrada constituída geralmente por imagens de tamanho normalizado e centralizadas, na qual são processadas nessas camadas iniciais. Após esse processo, essa imagem é achatada (*flattening*) e realimentada à um classificador. Este classificador é constituído por uma RNA interconectada que será responsável pela classificação da imagem através das características extraídas (SANTOS, 2020).

Souza et al. (2020) explica que o treinamento da CNN é realizado na maioria dos casos com o *backpropagation*, pois este ajusta os pesos W dos neurônios por meio do erro mensurado entre o verdadeiro e a predição da rede, utilizando componentes do vetor gradiente.

Podemos visualizar o fluxo de uma CNN na figura a seguir.

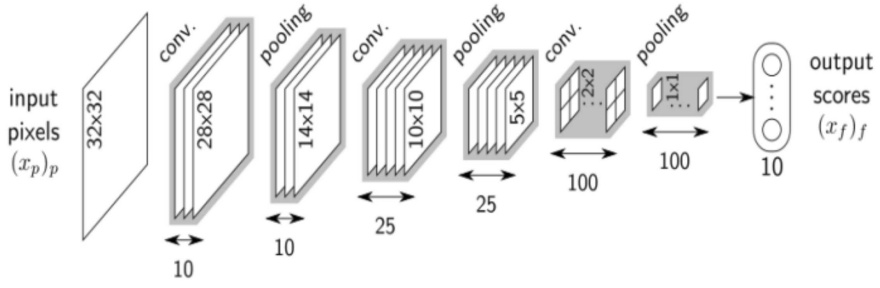

Figura 1 – Fluxo de Funcionamento de uma Rede Neural Convolucional
Fonte: Chandra (2017 apud MENDES, 2018, p. 12)

É possível diferenciar a CNN de uma Rede Neural Artificial - RNA, por meio da existência de uma camada que realiza a extração das características na imagem, para utilização na entrada da CNN. O que caracteriza esse tipo de rede é ser composta

basicamente de camadas convolutivas, que processam as entradas considerando campos receptivos locais (FREITAS et al., 2019). Esta rede possui duas camadas principais, a camada de convolução e a camada de agrupamento ou *Pooling*.

2.3.1 Camada Convolucional

Quando mencionamos a necessidade de extração de características nos referimos a Camada Convolucional, esta é a camada inicial onde existe a filtragem por pesos, cada neurônio atua como um filtro (*kernel*) que possui pesos específicos. A entrada é uma matriz de *pixels* que representa uma imagem de formato altura(a) × largura(l) × dimensão(d) , onde a dimensão consiste no número de canais (escala de cores), onde pode ser processada por operações de normalização ou de preenchimento (padding) (SANTOS, 2020).

Esta camada realiza a divisão da imagem em partes menores, que são mencionados na literatura como campos receptivos, a convolução acontece quando estes campos são multiplicados com os pesos nos filtros, o resultado desta multiplicação são as características extraídas da imagem em um mapa de atributos.

2.3.2 Camada de *Pooling*

Na camada de *Pooling* ocorre a diminuição espacial da entrada o que consequentemente diminui o custo computacional. Ela recebe os valores de uma parte da região do mapa de atributos, que foram gerados pela camada convolucional, e os substitui por alguma métrica dessa região (OLIVEIRA; CAMARA, 2019).

A redução de dimensionalidade da imagem é realizada por funções matemáticas, os tipos de funções mais utilizados são o *Max Pooling* e *Average Pooling*.O *Max Pooling* utiliza o maior valor dentro de uma vizinhança retangular coberta por um *kernel* para realizar a substituição. Já o *Average Pooling* retorna a média de todos os valores cobertos por um *kernel* (ZHOU; CHELLAPPA, 1988 apud SANTOS, 2020, p. 23)

2.4 Redes Neurais Convolucionais usadas para Detecção de Objetos e Classificação de Imagens

Com a crescente utilização do Processamento de Imagens em diversas aplicações, a comunidade de desenvolvimento percebeu a necessidade da existência de conjuntos de dados com imagens para treinamento e testes de suas redes e atualmente contamos com vários conjuntos de dados, alguns pagos outros gratuitos, um destes conjunto é o *ImageNet*.

Tal conjunto possui mais de 14 milhões de imagens divididas em mais de 20 mil categorias, todas as imagens passaram por um processo de caracterização manual. Para

estimular o desenvolvimento de CNNs que utilizem o banco de imagens, anualmente ocorre o *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), um desafio no qual a CNN com melhor desempenho na classificação ganha.

Podemos verificar algumas das melhores redes desenvolvidas durante o evento, assim como para outros conjuntos de dados, dos últimos três anos na imagem a seguir.

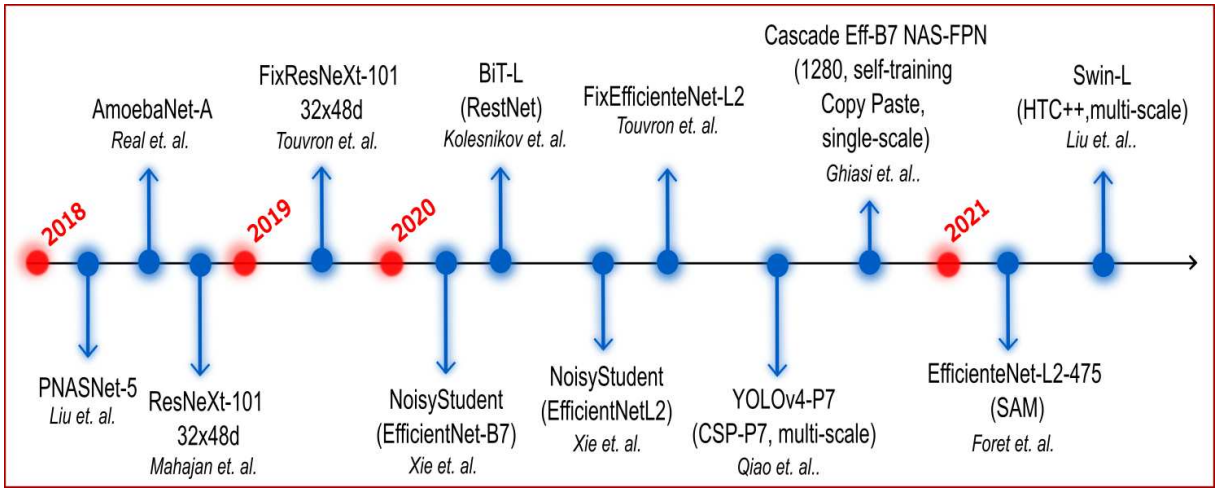

Figura 2 – Levantamento das melhores redes de classificação disponibilizadas pela *ImageNet* entre 2018 e 2021.

Fonte: da autora.

2.4.1 PNASNet-5

Esta rede é resultado de um trabalho desenvolvido por Liu et al. (2018a), o qual objetiva mostrar a aceleração de busca para as CNNs utilizando pesquisa progressiva, para isto, é criado um ambiente com o mínimo de células, tanto de treinamento quanto para testes.

Sendo empregado então um preditor substituto desenvolvido com a utilização de LSTM, em que passa por uma camada compartilhada e sigmoidal para regredir a precisão da avaliação.

2.4.2 AmoebaNet-A

Proposto por Real et al. (2018), é uma rede de classificação de imagens que objetiva melhorar a classificação realizada por algoritmos evolutivos. Para isto, modificou-se um algoritmo de seleção de torneio evolutivo, introduzindo uma propriedade de idade para favorecer os genótipos mais jovens.

O *AmoebaNet*-A tem uma precisão comparável aos modelos atuais da *ImageNet* de última geração descobertos com métodos de pesquisa de arquitetura mais complexos. Esta define uma nova precisão de *ImageNet* de 83,9% / 96,6% de última geração.

2.4.3 ResNext-101 32x48d

Mahajan et al. (2018) propõe um estudo sobre transferência de aprendizado com grandes redes convolucionais treinadas para prever *hashtag* em bilhões de imagens de mídia social.

Para isto ele faz uso da *ResNext*, redes residuais com camadas agrupadas, especificamente a *ResNext*-101 32x48d, esta possui 101 camadas, 32 grupos e larguras por grupo de 4, 8, 16, 32, e 48.

Mostrou-se melhorias em várias tarefas de classificação de imagens e detecção de objetos, e pode relatar a maior acurácia no *ImageNet*-1k, top-1 com 85,4%.

2.4.4 FixResNeXt-101

Utilizando o modelo *ResNeXt*-101 32x48d Touvron et al. (2019) propôs uma alternativa para otimização do classificador quando as resoluções de treino e testes divergem.

ResNeXt-101 é um classificador pré-treinado de forma fracamente supervisionada com 940 milhões de imagens públicas na resolução 224x224 e posteriormente otimizado para a resolução de 320x320 onde obtém-se precisão de 86,4% na classificação.

2.4.5 NoisyStudent (EfficientNet-B7)

Xie et al. (2019) explica que o*Noisy Student Training*, é uma abordagem de aprendizagem semi-supervisionada que funciona bem mesmo quando os dados rotulados são abundantes. Ele alcança 88,4% de precisão top-1 no *ImageNet*, que é 2,0% melhor do que o modelo de última geração que requer 3,5B de imagens do Instagram mal rotuladas.

Em conjuntos de teste de robustez, esta abordagem melhora a precisão do *ImageNet*-A top-1 de 61,0% para 83,7%, reduz o erro médio de corrupção do *ImageNet*-C de 45,7 para 28,3 e reduz a taxa média de alteração do *ImageNet*-P de 27,8 para 12,2.

2.4.6 BiT-L (RestNet)

O *Big Transfer* (BiT) como Kolesnikov et al. (2019) nomeou sua rede de classificação, combina alguns componentes cuidadosamente selecionados e transfere usando uma heurística simples, o que resultou em um ótimo desempenho em mais de 20 conjuntos de dados.

BiT tem um bom desempenho em uma gama surpreendentemente ampla de regimes de dados - de 1 exemplo por classe a 1 milhão de exemplos no total. A BiT atinge 87,5% de precisão top-1 no ILSVRC-2012, 99,4% no *CIFAR*-10 e 76,3% no *Benchmark* de Adaptação de Tarefa Visual (VTAB) de 19 tarefas.

BiT atinge 76,8% no ILSVRC-2012 com 10 exemplos por classe e 97. 0% no *CIFAR*-10 com 10 exemplos por aula.

2.4.7 FixEfficientNet-L2

Touvron et al. (2020) apresenta em seu artigo uma rede que supera a estrutura inicial do EfficientNet levando em conta a mesma quantidade de parâmetros.

Também afirma que o *FixEfficientNet*-B0 treinado sem dados de treinamento adicionais atinge 79,3% de precisão top-1 no *ImageNet* com parâmetros de 5,3M. Esta é uma melhoria absoluta de + 0,5% em relação ao antigo *Noisy FixEfficientNet*-B0 treinado com 300M de imagens não rotuladas.

Um *EfficientNet*-L2 pré-treinado com supervisão fraca em 300M de imagens não rotuladas e ainda mais otimizado com *FixRes* atinge 88,5% de precisão top-1, o que estabelece o novo estado da arte para *ImageNet* com um único corte.

2.5 YOLOv4-P7(CSP-P7, multi-scale)

Wang, Bochkovskiy e Liao (2020) apresenta uma abordagem de escalonamento de rede que modifica não apenas a profundidade, largura e resolução, mas também a estrutura da rede.

O modelo YOLOv4-large atinge resultados de última geração: 55,5% AP (73,4% AP50) para o conjunto de dados MS COCO, enquanto com o aumento do tempo de teste, *YOLO*v4-*large* atinge 56,0% *Average Precision* (AP) (73,3 AP50). Este modelo é projetado para GPU em núvem, afim de realizar o dimensionamento do tamanho de entrada, modificando a escala da profundidade para 2 e é utilizado o tempo de inferência como restrição para adicionar a escala da largura tradicional.

O modelo YOLOv4-*tiny* atinge 22,0% AP a uma velocidade de 443 FPS na RTX 2080Ti (WANG; BOCHKOVSKIY; LIAO, 2020).

2.6 Cascade Eff-B7 NAS-FPN(1280, self-training Copy Paste, single-scale)

Ghiasi et al. (2020) realizou um estudo sistemático do aumento *Copy-Paste* para segmentação de instância, neste é colado de forma aleatória objetos em uma imagem e selecionado aleatoriamente duas imagens que sofrem a alteração por meio de tremulação em escala aleatória e inversão horizontal, depois é selecionado um conjunto aleatório de objetos de uma das imagens e é colado na outra imagem. Seguindo para uma etapa

de remoção dos objetos totalmente obstruídos da imagem, os objetos que apresentam obstrução parcial sofrem com a alteração de suas máscaras e caixas delimitadoras.

O processo de treinamento desta rede consiste em treinar um modelo, neste caso o *Mask*-RCNN, supervisionado com o copiar e colar no rótulo de dados, gerar rótulos em dados não rotulados, colar instâncias verdadeiras em imagens pseudo-rotuladas e treinar um modelo com base nestes novos dados.

Os autores descobriram que o mecanismo simples de colar objetos aleatoriamente é bom o suficiente e pode fornecer ganhos sólidos no topo de linhas de base fortes. Além disso, mostraram que o *Copy-Paste* é aditivo com métodos semi-supervisionados que aproveitam dados extras por meio de pseudo-marcação por exemplo, o autotreinamento.

Na segmentação da instância COCO, alcançou-se 49,1 AP de máscara e 57,3 AP de caixa, uma melhoria de +0,6 AP de máscara e +1,5 AP de caixa em relação ao estado da arte anterior. Demontrou-se ainda que *Copy-Paste* pode levar a melhorias significativas no benchmark LVIS.

2.6.1 EfficientNet-L2-475 (SAM)

Foret et al. (2020) apresenta um procedimento de classificação chamado *Sharpness-Aware Minimization* (SAM), este busca parâmetros que se encontram em bairros com perdas uniformemente baixas, esta formulação resulta em um problema de otimização mínimo-máximo em que a descida do gradiente pode ser realizada de forma eficiente.

O procedimento fornece nativamente robustez para rotular o ruído no mesmo nível que os procedimentos de última geração, que visam especificamente o aprendizado com rótulos ruidosos.

2.6.2 Swin-L(HTC++,multi-scale)

Liu et al. (2021) propõem um transformador hierárquico cuja representação é calculada com janelas deslocadas. O esquema de janelamento alterado traz maior eficiência ao limitar o cálculo de autoatenção a janelas locais não sobrepostas, ao mesmo tempo que permite a conexão entre janelas.

Essa arquitetura hierárquica tem flexibilidade para modelar em várias escalas e tem complexidade computacional linear em relação ao tamanho da imagem. Essas qualidades do *Swin Transformer* o tornam compatível com uma ampla gama de tarefas de visão, incluindo classificação de imagem (86,4 precisão top-1 no *ImageNet-1K*) e tarefas de previsão densa, como detecção de objetos, 58.7 caixa AP e 51.1 máscara AP no teste COCO dev.

Seu desempenho supera o estado da arte anterior por uma margem de +2,7 AP de

caixa e +2,6 de AP de máscara em COCO e +3,2 mIoU em ADE20K, demonstrando o potencial dos modelos baseados em *Transformer* como *backbones*, redes neurais que servem como espinha dorsal para modelos de visão.

3 Metodologia

A fim de cumprir com os objetivos do presente trabalho, é realizado uma comparação entre três Redes Neurais Convolucionais com o propósito de determinar qual rede, com alteração apenas em seus parâmetros, apresenta o melhor resultado em contexto de pequenos volumes de dados, afim de utilizar no sistema de caracterização de espaços para eventos.

Para realizar as análises, são utilizadas ferramentas, em sua maioria, disponibilizadas de forma gratuita devido a variedade de material didático e acessibilidade.

O atual capítulo está dividido em uma apresentação e explicação breve do tipo de pesquisa empregada, dos materiais utilizados e posterior apresentação dos experimentos realizados e do capítulo de resultados.

3.0.1 Tipo da Pesquisa

O presente trabalho é construído com base em uma pesquisa aplicada de caráter experimental que objetiva apresentar resultados quantitativos.

3.0.2 Materiais Utilizados

Tabela 1 – Lista dos materiais utilizados para treinamento dos modelos a seguir.

Material	Versão
Google Colab	Colab Pró
CUDA	11.2
TensorFlow	1.15.2
Keras	2.2.5
Scikit-Image	0.16.2
OpenCV	4.0
Python	3.7.13
Dataset MSRA	B

3.0.2.1 Descrição dos Materiais

Devido a variedade de ferramentas utilizadas apresenta-se a seguir uma breve explicação de cada uma.

Google Colab: É uma ferramenta que permite que você misture código fonte (geralmente em *python*) e texto rico (geralmente em *markdown*) com imagens e o resultado desse código. É uma técnica conhecida como notebook (“caderno”) (SANTOS, 2020).

CUDA: É uma plataforma de computação paralela e um modelo de programação desenvolvido pela NVIDIA para computação geral em unidades de processamento gráfico (GPUs) (NVIDIA, 2022).

TensorFlow: é uma biblioteca de código aberto criada para aprendizado de máquina, computação numérica entre outras tarefas. Desenvolvido pelo *Google* em 2015 e se tornou uma das principais ferramentas para *machine learning* e *deep learning* (TECH, 2022).

Keras: É uma *Application Programming Interface* de redes neurais de alto nível, escrita em Python e capaz de rodar em cima de *TensorFlow*, CNTK, ou *Theano*, desenvolvida com o foco em possibilitar rápida experimentação (KERAS, 2022).

Scikit-Image: É uma coleção de algoritmos para processamento de imagens. Está disponível gratuitamente e sem restrições (WALT et al., 2014).

OpenCV: É uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto. Sua construção fornecer uma infraestrutura comum para aplicativos de visão computacional e acelerar o uso da percepção da máquina nos produtos comerciais (OPENCV, 2022).

Python: É uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica (PYTHON, 2022).

MSRA-B: Conjunto de dados que inclui 5.000 imagens, originalmente contendo retângulos rotulados de nove usuários desenhando uma caixa delimitadora em torno do que eles consideram o objeto mais saliente. Há uma grande variação entre as imagens, incluindo cenas naturais, animais, interiores, exteriores, etc (WANG et al., 2017).

3.0.3 Medidas de desempenho

Na literatura existe diversas formas de medir o desempenho do quanto aos diversos processos da detecção do objetos, neste trabalho será empregado o conceito de verdadeiro/positivo e falso/negativo promovido pela matriz de confusão para determinar a precisão, a sensibilidade(*recall)*, o f1-*score* e a acurácia.

Embora seja mais utilizada para algoritmos de classificação de imagens, a matriz de confusão também se adéqua na detecção de objetos pela ocorrência da segmentação da imagem durante o processo. De acordo Skeika (2019) existem quatro situações ponderadas na matriz:

Verdadeiro Positivo (VP): que representa o total de *pixels* segmentados, indicando um melhor resultado se a quantidade for grande.

Falso Positivo (FP): apresenta a quantidade de *pixels* que são identificados em uma imagem mas não pertencem a ela.

Verdadeiro Negativo (VN): indica um pixel que é identificado corretamente como não pertencente a uma imagem.

Falso Negativo (FN): mede a taxa de *pixels* pertencentes ao alvo que tenham sido classificados como não pertencentes pelo algoritmo de segmentação.

É comum a utilização das quatro ocorrências em um modelo de classificação de imagens, entretanto em modelos de detecção o Falso Negativo é desconsiderada. Por meio destas medidas é possível realizar cálculos para simplificar a visão de desempenho:

Precisão: conforme Skeika (2019) mede o total de *pixels* positivos no *ground truth* que também são considerados como positivos pela segmentação, dividido pelo número total de *pixels* positivos identificados como, de fato, pertencentes ao conjunto.

O termo *ground truth* é uma medida que indica o quão verdadeiro o algoritmo conseguiu chegar em relação a uma entrada, simplificando, é a taxa de percepção sobre a verdade que o algoritmo apresenta sobre o conjunto de validação. A precisão pode ser encontrada por meio da seguinte equação:

$$precisão = \frac{VP}{VP + FP}$$

Sensibilidade ou *Recall*: são os *pixels* que são considerados positivos tanto pela segmentação quanto no *ground truth*, pode ser obtida por:

$$recall = \frac{VP}{VP + FN}$$

F1-*Score*: média entre precisão e *recall*, quanto menor o valor menor a precisão ou o *recall*.

$$F1 - score = 2 \times \frac{precisão \times recall}{precisão + recall}$$

Acurácia: é a soma de Verdadeiros Positivos com Verdadeiros Negativos divididos pelo todo, indica a confiabilidade da rede quanto a seu aprendizado.

$$acurácia = \frac{VP + VN}{VP + VN + FP + FN}$$

3.0.4 Definição dos Experimentos

Para realização dos experimentos foram utilizadas redes que apresentaram acurácia superior a 85% dentre as apresentadas no referencial teórico, pois estas possuem maior capacidade de especialização e são redes modernas, cuja a probabilidade de se alcançar os objetivos são maiores.

Cada rede necessita de uma configuração específica em relação aos dados, as quais são explicadas a seguir no decorrer da exemplificação do experimento de cada rede, entretanto, houve a necessidade em comum da separação das imagens, em categorias estipuladas pela própria autora, para prosseguir com o treinamento e processo de comparação.

O apêndice .1 apresenta explicação em detalhes sobre alterações realizadas e construção das anotações nas redes.

3.0.5 Base de dados

Os experimentos são realizados utilizando o MSRA-B, um *dataset* bastante empregado em projetos de detecção de objetos notáveis, que contém 5 mil imagens rotuladas de três usuários, como Liu et al. (2011), Jiang et al. (2013), Achanta et al. (2009), a nove usuários, em que mapeiam uma caixa delimitadora ao redor do objeto considerado mais saliente.

O *dataset* apresenta bastante variedade entre as imagens, que podem conter pessoas, animais, cenas naturais, interiores e exteriores etc.

Com este *dataset* são distribuídas sete imagens para treinamento de cada classe e três imagens para validação, sendo o experimento dividido entre duas classes, cinco classes e vinte classes para serem treinadas pelas três redes escolhidas.

Embora o *dataset* contenha um total de cinco mil imagens, este abrange uma variedade enorme, sendo difícil identificar mais de dez imagens de uma só categoria já que não encontram-se em sequência. Como as redes necessitam de manipulação manual para adequação do tipo de anotação aceita, é necessário a seleção das imagens para posterior anotação dos objetos.

3.0.6 YOLOv4

O modelo de CNN *Yolo*v4 trata-se de uma rede de detecção de objetos de um estágio baseado em âncora, que faz uso do *backbone CSPDarknet53* como rede neural para operação na camada de classificação da imagem. Este *backbone* possui 29 camadas convolucionais de proporção 3x3 e um campo receptivo de 725x725 e 27,6 parâmetros M.

Para criação do modelo de detecção adicionam ao antigo modelo *Yolo*v3 o bloco *Spatial Pyramid Pooling* (SPP), de acordo Bochkovskiy, Wang e Liao (2020), este bloco aumenta significativamente o campo receptivo, separa o recursos de contexto mais significativos e quase não causa redução da velocidade de operação da rede.

O bloco SPP é originado do *Spatial Pyramid Matching* (SPM) que o integra à CNN e usa a operação *max-pooling* invés da operação *bag-of-word* (BOCHKOVSKIY; WANG; LIAO, 2020).

Ao modelo é empregado o *Path Aggregation Network* (PANet) como o método de agregação de parâmetros de diferentes níveis do *backbone* para diferentes níveis de detector. É aprimorado toda a hierarquia de recursos com sinais de localização precisos em camadas inferiores de baixo para cima do caminho de aumento, que encurta o caminho da informação entre as camadas inferiores e o recurso superior (LIU et al., 2018b).

O modelo utiliza *DropBlock* como método de regularização, uma forma de *dropout* estruturado, onde as unidades em uma região contígua de um mapa de características são descartadas juntas (GHIASI; LIN; LE, 2018).

É então, aplicado um novo método para aumento de dados reconhecido como *Mosaic* e *Self-Adversarial Training* (SAT). O *Mosaic* mistura quatro imagens de treinamento, isso permite a detecção e objetos fora de seu contexto normal, ou seja, com dimensionamento reduzido.

Já o SAT representa um método para aumento de dados, este opera em dois estágios, no primeiro a imagem é modificada, invés dos pesos, no segundo estágio a rede é treinada para identificar um objeto na imagem modificada.

De forma geral, nas modificações do *YOLO*v4, em relação à sua versão anterior, estão o método *Path Aggregation Network* (PAN), o *Spatial Attention Module* (SAM) e *Cross mini-Batch Normalization* (CmBN), para adequação, do *design* proposto, para treinamento e detecção.

Em relação ao *backbone*, além do método *Mosaic* e SAT, faz uso do *CutMix*, um método que serve para cobrir a imagem recortada na região do retângulo de outras imagens, e ajusta o rótulo de acordo com o tamanho da área de mixagem, é também aplicado suavização de rótulos de classe.

Do mesmo modo, é aplicado ao *backbone* a função de ativação *Mish*, função que resolve o problema do gradiente nulo, *Cross-stage partial connections* (CSP), *Multiinput weighted residual connections* (MiWRC).

Quanto ao detector é aplicado CIoU-*loss*, CmBN, regularização com *DropBlock*, Mosaic, SAT, múltiplas âncoras para uma verdade, *cosine annealing scheduler*, hiperparâmetros ótimos e treinamento aleatório.

Além da atribuição da função *Mish*, bloco SPP, bloco SAM, PAN e DIoU-NMS.

Para treinamento são utilizados o *framework darknet* ajustado para implementação do modelo, juntamente com a biblioteca de visão computacional *OpenCV* aplicados em nuvem com a utilização da plataforma de computação paralela CUDA, a figura 3 a seguir apresenta com mais detalhes a execução do treinamento do modelo.

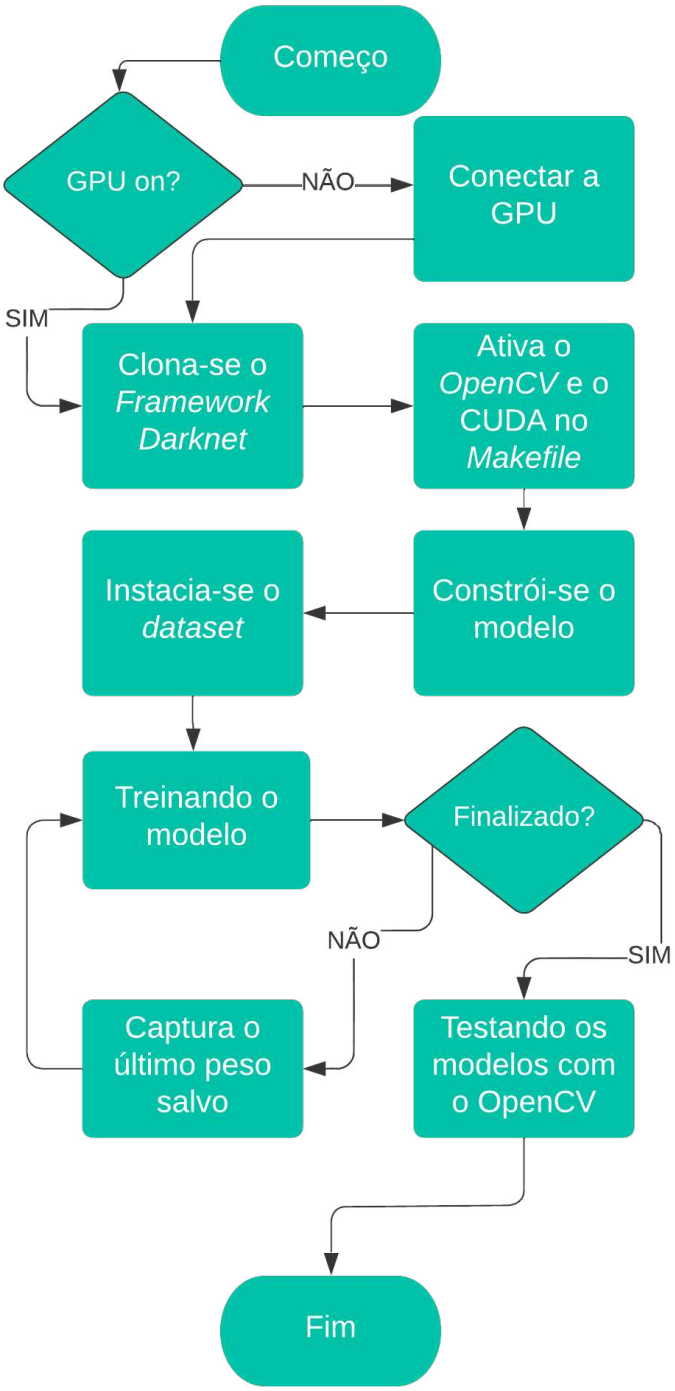

Figura 3 – Fluxograma de execução do modelo *Yolo*v4 em plataforma de processamento em nuvem.

Fonte: da autora.

Dentre as necessidades para utilziação do modelo apresentam-se a delimitação manual nas imagens, que são realizados utilizando a ferramenta *YoloLabel*[1] e alterações nos parâmetros presentes no arquivo de pesos padrão, que são originados do treinamento com o *dataset* COCO.

[1] https://github.com/developer0hye/Yolo$_L$*abel*

O modelo gera automaticamente o gráfico de perda atualizado, entretanto, impossibilita-se o resgate dos gráficos devido a perda de conexão com o *google colab*.

Para este trabalho, o arquivo de pesos, anteriormente citado é alterado da seguinte forma:

Tabela 2 – Ajuste de parâmetros no arquivo de pesos para treinamento da rede *Yolov4*

	Classes	*Max Batches*	*Steps*	*Filters*
2 Classes	2	6000	4800,5400	21
5 Classes	5	10000	8000,9000	30
20 Classes	20	40000	32000,36000	75

O treinamento do modelo para as duas classes escolhidas, aproximou-se de 15 horas de execução, persistindo os dados do treinamento para análise posterior, sendo salvos atualizados a cada mil épocas, totalizando 6 mil épocas na execução. Quanto ao treinamento do modelo com cinco classes aproximou-se de 21 horas de execução, sendo seguindo o experimento com duas classes e tendo seus pesos atualizados a cada mil épocas, em um total de 10 mil épocas executadas. O treinamento para vinte classes seguiu com 52 horas de processamento.

3.0.7 Mask-RCNN

Tal modelo caracteriza-se por ser uma implementação elaborada do modelo *Faster*-RCNN, o qual possui duas saídas para cada objeto candidato, um rótulo de classe e um deslocamento da caixa delimitadora. Para desenvolvimento do modelo, é adicionado um terceiro ramo, com a finalidade de criar a máscara do objeto.

Este é um modelo de dois estágios em que o primeiro conta com *Region Proposal Network* (RPN), que propõe as caixas delimitadoras e o segundo que utiliza a *Region of Interest* (RoI) de cada caixa delimitadora para extrair recursos, e executar a classificação e regressão da caixa, é gerado uma máscara binária para cada região de interesse.

Para aprimoramento das máscaras, o modelo utiliza *Fully Convolutional Networks* FCN, conforme os criadores do método Long, Shelhamer e Darrell (2015), FCN são redes que recebam entradas de tamanho arbitrário e produzam saídas de tamanho correspondente com inferência e aprendizado eficientes. Para (HE et al., 2017) isso permite que cada camada na ramificação da máscara mantenha o *layout* espacial explícito do objeto m $\times$ m sem reduzi-lo em uma representação vetorial sem dimensões espaciais.

Devido ao comportamento *pixel* a *pixel* promovido pelas camadas totalmente conectadas, a rede exige que o mapa de recursos mantenha alinhamento, afim de preservar a correspondência espacial explícita do *pixel*. E para isto, a rede apresenta uma camada chamada *RoIAlign*.

RoIAlign é uma camada que alinha de forma adequada os recursos extraídos da entrada que difere do método padrão, *RoIPool*, faz uso de interpolação bilinear para calcular os valores exatos do recursos de entrada, sua utilização faz-se necessária para evitar uma quantificação dos limites do *RoI*.

Para ganho em precisão e velocidade, a rede utiliza como *backbone* o *Feature Pyramid Network* (FPN), de acordo acordo os autores, Lin et al. (2016), da FPN é uma rede que objetiva alavancar um recurso piramidal da hierarquia *ConvNet*, que tem semântica de baixo a alto nível, e construir uma pirâmide de recursos com semântica de alto nível por toda parte, esta rede usa uma arquitetura *top-down* com conexões laterais para construir um pirâmide de recursos de rede a partir de uma entrada de escala única (HE et al., 2017).

Para desenvolvimento da *head*, é reaproveitado dos modelos anteriores, o *ResNet* uma estrutura de aprendizado residual para facilitar o treinamento de redes que são substancialmente mais profundas.

Todas as convoluções da rede são 3×3, exceto a saída da convolução que é 1×1, as de-convoluções são 2×2, e utilizando *ReLU* em camadas ocultas como função de ativação.

Para treinamento do modelo é utilizado o *Python*, a biblioteca *TensorFlow*, juntamente ao *Keras* e o *Scikit-Image*, durante o treinamento é essencial adequar os arquivos do *Python* e do *Keras* às versões utilizadas, pois o modelo não possui mecânica atualizada, apresentando a necessidade de ajuste manual.

Durante o treinamento é inicialmente utilizado os pesos pré-treinados fornecidos por Abdulla (2017), os quais são feitos com a utilização do COCO *dataset*. Os pesos atualizados são salvos no *drive* para questão de perda durante o treinamento, seus pesos são salvos um a um, diferentemente do *Yolov4* que salvava a cada 1000 épocas.

Como parâmetros de alteração, o modelo exige a alteração dos diretórios para treinamento, teste, e inserção manual das classes dentro do arquivo de customização, as outras alterações serão mencionadas na apresentação dos resultados dos experimentos.

Além das alterações já mencionadas, também no arquivo de customização, é fundamental a modificação dos seguintes parâmetros:

Tabela 3 – Ajuste de parâmetros no arquivo de customização para treinamento da rede *Mask* R-CNN

	Classes	*Name*	*Images per GPU*	*Step per Epoch*	*EPOCH*
2 Classes	1+2	*object*	1	100	200
5 Classes	1+5	*object*	1	100	500
20 Classes	1+20	*object*	1	100	2000

3.0.8 Cascade R-CNN

É uma extensão da arquitetura *Faster* R-CNN em que foca-se na sub-rede de detecção, igualmente ao Mask-RCNN esta rede faz uso do *Region Proposal Network* (RPN) não se limitando apenas a este, Cai e Vasconcelos (2019) afirma que objetivando aumentar a qualidade das hipóteses e do detector, é empregado uma combinação de regressão de caixa delimitadora em cascata e detecção em cascata.

Devido a não uniformidade de um único mecanismo de regressão, é aplicada, às caixas delimitadoras, uma abordagem em que a tarefa de regressão é decomposta em uma sequencia de etapas mais simples, consistindo em uma cascata de regressões especializadas.

Afim de promover aumento entre os limites de *Intersection over Union* IoU U, o detector fornecido pela rede faz uso de regressão em cascata como um mecanismo de re-amostragem que altera a distribuição de hipóteses processadas pelas diferentes etapas. De acordo os autores um mecanismo de regressão de caixa delimitadora treinado para uma certa função indicadora U tende a produzir caixas delimitadoras de IoU mais alta.

Cai e Vasconcelos (2019) apresenta três vantagens na utilização de detecção em cascata: o potencial de *overfitting* em grandes limites de IoU é reduzido, uma vez que exemplos positivos se tornam abundantes em todos os estágios. Em segundo lugar, os detectores de estágios mais profundos são ideais para limiares de IoU. Terceiro, porque alguns valores discrepantes são removidos à medida que o limite de IoU aumenta.

Em divergência com o *Mask*-RCNN, o *Cascade Mask*-RCNN possui várias ramificações de detecção, em que é adicionado uma única *head* de previsão de máscara no primeiro ou último estágio da rede e adicionando uma ramificação de segmentação a cada estágio em cascata.

De forma geral o *Cascade Mask*-R-CNN é implementado com quatro estágios: um RPN e três *heads* de detecção com limiares U = 0,5, 0,6, 0,7. Nenhum aumento de dados é empregado, exceto a inversão de imagem horizontal.

Para treinamento do modelo é utilizado o *Python*, a biblioteca *TensorFlow*, juntamente ao *Keras* e o *Scikit-Image*, assim como no *Mask RCNN* a rede necessita de alteração dos arquivos do para adequação às versões utilizadas.

Durante o treinamento é inicialmente utilizado os pesos pré-treinados fornecidos por Abdulla (2017), os quais são feitos com a utilização do COCO *dataset*, estes são persistidos para casos de perda ou falha durante o processo de treinamento, entretanto para adequação do modelo é utilizado o repositório Anonymoussss (2019) que apresenta o modelo desenvolvido utilizando regressão e cascata.

Como a principal diferença entre esta rede e a *Mask*-RCNN está no detector, sendo então possível reaproveitar o arquivo de customização que fornece dados das classes e

caminhos para o *dataset*, sendo então empregado os mesmos parâmetros:

Tabela 4 – Ajuste de parâmetros no arquivo de customização para treinamento da rede *Cascade* R-CNN

	Classes	*Name*	*Images per GPU*	*Step per Epoch*	*EPOCH*
2 Classes	1+2	*object*	1	100	200
5 Classes	1+5	*object*	1	100	500
20 Classes	1+20	*object*	1	100	2000

3.1 Resultados

Para as medidas de desempenho, são calculados a precisão, a sensibilidade (*recall*), acurácia e f1-*score*:

3.1.1 Yolov4

3.1.1.1 Com 2 Classes

Tabela 5 – Resultados obtidos a partir do treinamento do modelo *Yolov4* com 2 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Garrafa	1	1	1	100%
Lata	0	0	0	0%

Para o experimento com apenas duas classes o modelo apresenta-se com divergência entre as classes, conseguindo manipular precisamente a classe Garrafa mas não apresentando resultados na classe Lata, essa divergência pode ser consequência da escolha das imagens, mesmo o *dataset* MSRA-B abrangendo uma quantidade diversificada de imagens, algumas apresentam conteúdo de baixa qualidade, no caso das latas, uma parte das imagens apresentavam imagens de latas torcidas ou amassadas.

3.1.1.2 Com 5 Classes

Como resultado foi obtido o gráfico de perda de generalidade para o experimento com 5 classes.

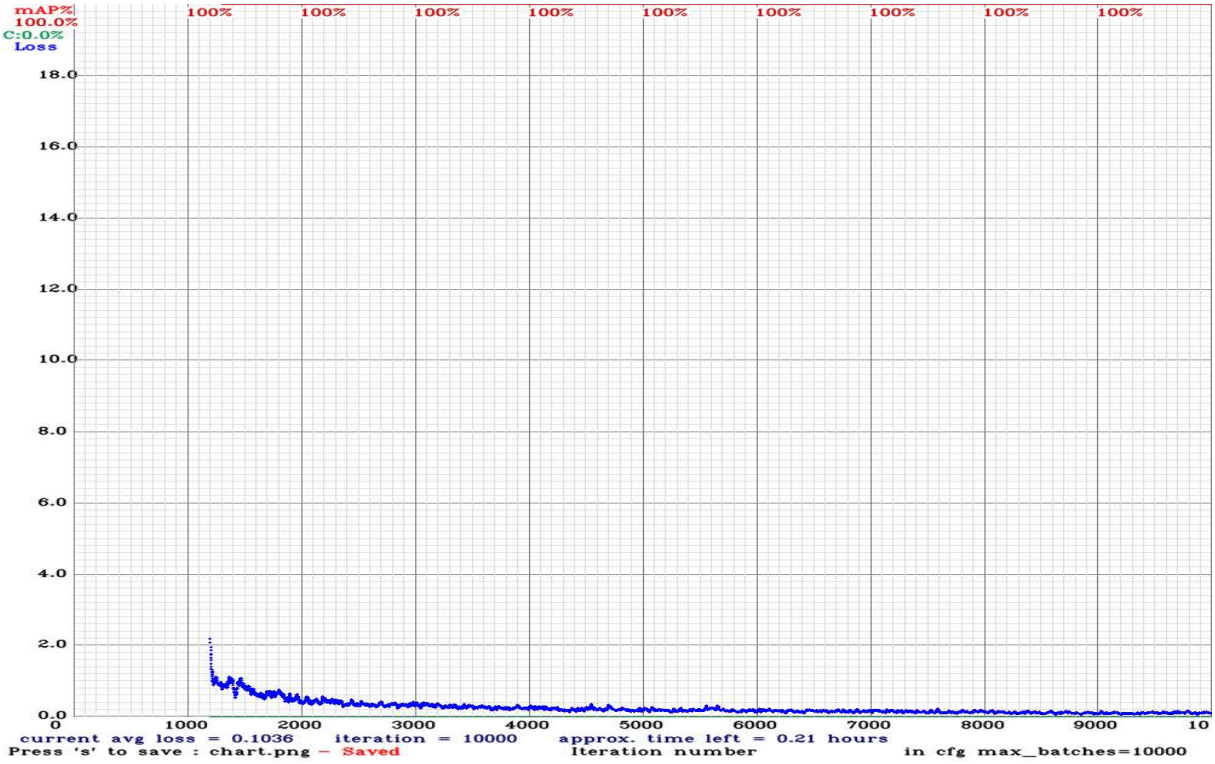

Figura 4 – Gráfico de perda dos pesos 1000 a 10000 interações
Fonte: da autora.

A Figura 4 apresenta um gráfico de perda de generalidade com início em mil e termina em 10 mil épocas. Ao analisá-la é possível observar que a perda deixa de ser extremamente significante a partir de 4 mil épocas, o que indica a estabilidade do processo de aprendizado acarretando em especialização da rede no conjunto de treinamento.

A tabela a seguir apresenta os resultados das medidas de desempenho aplicadas sobre o treinamento com 5 classes.

Tabela 6 – Resultados obtidos a partir do treinamento do modelo *Yolov4* com 5 classes.

Classes	Precisão	Recall	f1-*score*	Acurácia
Flor	0,75	0,42	1,26	37,5%
Placa	0,5	0,33	0,39	25%
Ave	1	0,33	0,49	33%
Ovo	0,66	0,33	0,44	28%
RelogioPonteiro	1	0,33	0,49	42,8%

Em relação a acurácia do treinamento de cinco classes, o resultado já tornou-se menos promissor, mesmo as classes que apresentaram uma ótima precisão não possuíram uma acurácia relevante o suficiente para caracterizar como um treinamento robusto.

3.1.1.3 Com 20 Classes

O resultado para a detecção com vinte classes seguiu semelhante ao apresentado para cinco classes, mesmo as classes com uma boa precisão não proporcionaram uma boa

Tabela 7 – Resultados obtidos a partir do treinamento do modelo *Yolov4* com 20 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Aranha	0	0	0	0%
Borboleta	0,66	0,58	0,61	5,71%
Cachorro	0,40	0,05	0,08	5,40%
Caracol	0	0	0	0%
Carro	1	0,08	0,14	8%
Cavalo	1	0,11	0,19	11%
Chapéu	0,5	0,03	2	2,9%
Concha	1	0,03	0,05	3%
Folha de bordo	0,66	0,05	0,09	5,7%
Gato	1	0,03	0,05	3,03%
Hibiscu	1	0,08	0,14	8,57%
Maçã	1	0,05	0,09	5,8%
Moeda	1	0,05	0,09	5,8%
Moto	1	0,08	0,14	8,5%
Ovelha	1	0,08	0,14	8,5%
Peixe	0,66	0,05	1,02	5,7%
Relógio de pulso	1	0,08	0,14	8,5%
Sapo	0	0	0	0%
Taça	0,57	0,11	0,49	10%
Urso	0,5	0,03	2	2,9%

acurácia, o que indica que de uma forma geral o modelo não realizou um bom treinamento.

3.1.2 Mask R-CNN

3.1.2.1 Com 2 classes

A tabela a seguir mostra os resultados obtidos a partir do treinamento proposto.

Tabela 8 – Resultados obtidos a partir do treinamento do modelo *Mask*-RCNN com 2 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Garrafa	0,71	0,83	0,76	62%
Lata	0,2	0,5	0,28	16%

Com esta abordagem a variação, assim como no caso *Yolov4*, ocorreu de forma drástica, apresentando melhor especialização no aprendizado da classe Garrafa.

3.1.2.2 Com 5 Classes

Com estas alterações, o treinamento originou a seguinte análise de desempenho:

Embora não muito eficiente apenas uma classe apresentou um treinamento positivo, enquanto as outras tiveram seus verdadeiros positivos apresentados em zero, significando

Tabela 9 – Resultados obtidos a partir do treinamento do modelo *Mask*-RCNN com 5 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Flor	0	0	0	0%
Placa	0	0	0	0%
Ave	0	0	0	0%
Ovo	0	0	0	0%
RelogioPonteiro	0,40	0,66	0,49	33%

que, de todos os *pixels* identificados, nenhum correspondia a imagem analisada como verdadeira, demonstrando falha durante o processo de aprendizado.

3.1.2.3 Com 20 Classes

Originando o seguinte resultado de treinamento:

Tabela 10 – Resultados obtidos a partir do treinamento do modelo *Mask*-RCNN com 20 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Cavalo	0,33	1	0,49	33%
Gato	0,66	1	0,79	66%

Embora seja um treinamento com vinte classes, apenas duas são apresentadas pois são as únicas que apresentaram seus Verdeiros Positivos, como as medidas de desempenho necessitam dele para apresentar resultados, torna-se improdutivo apresentar as medidas das outras classes pois estas estão em zero.

3.2 Cascade Mask R-CNN

3.2.1 Com 2 Classes

Em seguida realiza-se os cálculos das medidas de desempenho, que apresentam-se desta forma:

Tabela 11 – Resultados obtidos a partir do treinamento do modelo *Cascade Mask*-RCNN com 2 classes.

Classes	Precisão	*Recall*	f1-*score*	Acurácia
Garrafa	0	0	0	0%
Lata	0	0	0	0%

Com o treino de duas classes o modelo não apresenta-se capaz de identificar os Verdadeiros Positivos, sabe-se que o modelo realizou o treinamento pois os valores referentes aos Falsos Positivos e Falsos Negativos são expostos.

3.2.1.1 Com 5 classes

Assim como em duas classes, os resultados obtidos do treinamento com cinco classes são totalmente anulados devido a falta dos Verdadeiros Positivos. Diferentemente da rede *Yolo*v4 e *Mask*-RCNN, a rede baseada em cascata apresenta total falha no treinamento com poucos dados de amostras.

3.2.1.2 Com 20 classes

Para este modelo todas as classes apresentaram-se sem resultado em seus Verdadeiros Positivos, estendendo a hipótese apresentada no experimento com cinco classes, que compreende a rede como imprópria para manipulação com baixo volume de dados.

Tabela 12 – Tempo de execução dos experimentos em combinação com a rede

		YOLOv4	Mask RCNN	Cascade RCNN
Tempo de execução (horas)	2 classes	15	6	4
	5 classes	21	12	9
	20 classes	52	48	42

A seguir é apresentado a média da precisão obtida dos treinamentos para os contextos propostos, levando em consideração as redes aplicadas:

Tabela 13 – Média da Precisão obtida através dos treinamentos realizados nos três contextos em relação às redes aplicadas

		YOLOv4	Mask RCNN	Cascade RCNN
Mean Avarage Precision (MAP)	2 classes	50%	40%	0%
	5 classes	59,92%	13%	0%
	20 classes	59,68%	0%	0%

Um resultado percebido trata-se da diferença da média em relação a quantidade de classes treinadas, como pode ser percebido observando a Tabela 13 a rede *Mask*-RCNN apresenta resultados de forma decrescente conforme aumenta-se o número de classes treinadas por vez, já a rede *Yolo*v4 mostra-se capaz de treinar melhor com cinco classes.

Quanto ao exercício e poder computacional necessário para o treinamento das redes, como pode ser visto na Tabela 12 a rede *Yolo*v4 apresenta resultados após um longo tempo de treinamento, em contrapartida a rede *Mask RCNN*, trata-se de uma rede bastante leve, podendo ser treinada com apenas 20 épocas para cada classe quando aplicada em grandes volumes de dados.

Os experimentos seguiram com base em redes atuais, dos últimos cinco anos, que ainda não apresentam ser totalmente capazes de trabalhar com poucos dados, como é amplamente pacificado na literatura.

Embora a normalidade demonstre que a necessidade de grandes volumes de dados, a rede *Yolo*v4 no mesmo contexto de pouco volume, apresenta-se como alternativa para treinamento em pequenos *datasets*, a rede *Mask-RCNN* consegue resultados de treinamento em pequenos volumes, desde que treinado poucas classes.

4 Conclusão

O processamento de imagens é uma abordagem em ascensão no meio computacional, sua finalidade é a manipulação de imagens, como a delimitação de objeto, redimensionamento ou aumento de imagens, a conversão em imagem binaria para extração de características, delimitando a existência ou não de cor no pixel, redimensionamento da imagem para exclusão de fundo, e enfim reconhecimento de objetos presentes.

A aplicação em conjunto com as *Convolutional Neural Network* - CNN, apresentando-se como forma dinâmica e eficiente para o processamento de imagens, expandiu a quantidade de recursos a serem utilizados em cada etapa do processamento, com isto, observa-se a grande variedade de trabalhos voltados tanto para a detecção de objetos quanto para a classificação de imagens.

Um das maneiras de se obter trabalhos relativos à temática, podem ser encontrados em *sites* de *datasets* como por exemplo o *ImageNet* e o COCO. Contudo, tal performance não necessariamente é vinculado a um ambiente de imagens categoricamente construído, como no caso dos *datasets* citados, para funcionamento, abrindo possibilidades para desenvolvimento de um banco de imagens próprio.

Este trabalho apresenta uma comparação entre redes convolutivas focadas na detecção de objetos apresentando como a melhor em questão de precisão a rede Yolov4 que apresentou resultados superiores a 50% sendo então a rede provável a ser escolhida em uma aplicação de detecção e reconheicimento de objetos para caracterização de possíveis ambientes voltados para eventos em geral.

Referências

ABDULLA, W. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. [S.l.]: Github, 2017. <https://github.com/matterport/Mask_RCNN>.

ACHANTA, R. et al. Frequency-tuned salient region detection. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, p. 1597–1604, 2009.

ANONYMOUSSSS. 2019. Disponível em: <https://github.com/anonymoussss/Cascade-Mask-RCNN>.

AVELINO, J. et al. A sociedade de controle: manipulação e modulação nas redes digitais. *São Paulo:Hedra*, 2018.

BOCHKOVSKIY, A. et al. Yolov4: Optimal speed and accuracy of object detection. 04 2020.

CAI, Z.; VASCONCELOS, N. Cascade R-CNN: high quality object detection and instance segmentation. *CoRR*, abs/1906.09756, 2019. Disponível em: <http://arxiv.org/abs/1906.09756>.

CHANDRA, P. A survey on deep learning its architecture and various applications. 2017. Disponível em: <https://web.archive.org/web/20180516180348id_/http://gvschoolpub.org:80/journals/AJNNIA/vol1_no2/2.pdf>.

ENGELEN, J. E. van; HOOS, H. H. A survey on semi-supervised learning. *Machine Learning*, v. 109, n. 2, p. 373–440, 2020. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-019-05855-6>.

FERNANDES, J. G. L. et al. Inteligência artificial: Uma visão geral. In: . [S.l.: s.n.], 2018.

FORET, P. et al. Sharpness-aware minimization for efficiently improving generalization. *CoRR*, abs/2010.01412, 2020. Disponível em: <https://arxiv.org/abs/2010.01412>.

FREITAS, D. S. de et al. Reconhecimento da ceratoconjuntivite infecciosa bovinautilizando imagens termográcas e redes neurais convolucionais. Revista Brasileira de Computação Aplicada, v. 11, p. 133–145, 2019. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/9210/114114872>.

GHIASI, G. et al. Simple copy-paste is a strong data augmentation method for instance segmentation. *CoRR*, abs/2012.07177, 2020. Disponível em: <https://arxiv.org/abs/2012.07177>.

GHIASI, G. et al. *DropBlock: A regularization method for convolutional networks*. arXiv, 2018. Disponível em: <https://arxiv.org/abs/1810.12890>.

GHOSAL, S. et al. An explainable deep machine vision framework for plant stress phenotyping. *Proceedings of the National Academy of Sciences*, v. 115, n. 18, p. 4613–4618, 2018. Disponível em: <https://www.pnas.org/doi/abs/10.1073/pnas.1716999115>.

HE, K. et al. Mask r-cnn. In: . [S.l.: s.n.], 2017. p. 2980–2988.

JIANG, H. et al. Salient object detection: A discriminative regional feature integration approach. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2013. p. 2083–2090.

KAUFMAN, D.; SANTAELLA, L. O papel dos algoritmos de inteligência artificial nas redes sociais. 2020.

KERAS. Keras: Deep learning library for theano and tensorflow. 2022.

KOLESNIKOV, A. et al. Large scale learning of general visual representations for transfer. *CoRR*, abs/1912.11370, 2019. Disponível em: <http://arxiv.org/abs/1912.11370>.

LI, J. et al. Perceptual generative adversarial networks for small object detection. 2017. Disponível em: <https://arxiv.org/pdf/1706.05274v2.pdf>.

LIN, T.-Y. et al. *Feature Pyramid Networks for Object Detection*. arXiv, 2016. Disponível em: <https://arxiv.org/abs/1612.03144>.

LIU, C. et al. Progressive neural architecture search. Johns Hopkins University, 2018. Disponível em: <https://arxiv.org/pdf/1712.00559v3.pdf>.

LIU, S. et al. Path aggregation network for instance segmentation. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018.

LIU, T. et al. Learning to detect a salient object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 33, n. 2, p. 353–367, 2011.

LIU, Z. et al. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. Disponível em: <https://arxiv.org/abs/2103.14030>.

LONG, J. et al. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015.

MAHAJAN, D. et al. Exploring the limits of weakly supervised pretraining. *CoRR*, abs/1805.00932, 2018. Disponível em: <http://arxiv.org/abs/1805.00932>.

MENDES, M. de S. Aprendizado em profundidade na descriÇÃo semÂntica de imagens. In: . [S.l.]: Universidade Federal de Ouro Preto, 2018.

MONTEIRO, L. InteligÊncia artificial: A importÂncia do reconhecimento facial na educaÇÃo. *Revista Presença Geográfica*, v. 7, n. 1, p. 111, 2020. ISSN 24466646. Disponível em: <https://periodicos.unir.br/index.php/RPGeo/article/view/5317>.

NORVIG, P.; RUSSELL, S. *Inteligência artificial: Tradução da 3a Edição*. Elsevier Brasil, 2014. ISBN 9788535251418. Disponível em: <https://books.google.com.br/books?id=BsNeAwAAQBAJ>.

NVIDIA, C. Cuda zone. 2022. Disponível em: <https://developer.nvidia.com/cuda-zone>.

OLIVEIRA, C. Aprendizado de máquina e modelação do comportamento humano.

OLIVEIRA, P. F.; CAMARA, C. E. AnÁlise de desempenho de um algoritmo desenvolvido para soluÇÃo de deep learning utilizando redes neurais convolucionais para anÁlise de contraste de imagens. Revista Ubiquidade, v. 2, n. 1, p. 86–87, 2019. ISSN SSN 2236-9031. Disponível em: <https://revistas.anchieta.br/index.php/RevistaUbiquidade/article/view/1010/893>.

OPENCV team. About. 2022. Disponível em: <https://opencv.org/about/>.

PAZ, K. B. et al. IdentificaÇÃo de defeitos do tipo "panela" em pavimento asfÁltico por meio de redes neurais convolucionais. 34º Congresso de Pesquisa e Ensino em Transporte da ANPET, p. 866, 2020. Disponível em: <http://repositorio.ufc.br/bitstream/riufc/56719/1/2020_eve_kbpaz.pdf>.

PEREZ, B. de F. V.; JUNIOR, P. T. A. Geração de conjunto de dados sintéticos para detecção de instâncias de objetos. Departamento de Engenharia Elétrica, Centro Universitário FEI and Departamento de Ciência da Computação, Centro Universitário FEI, 2020. Disponível em: <https://fei.edu.br/sites/artigos_sicfei_2020/117_SICFEI2020_ARTIGO.pdf>.

POUYANFAR, S. et al. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 5, sep 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3234150>.

PYTHON, S. F. What is python? executive summary. 2022. Disponível em: <https://www.python.org/doc/essays/blurb/>.

REAL, E. et al. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018. Disponível em: <http://arxiv.org/abs/1802.01548>.

RéOS, J. P. P.; FARIAS, A. R. O uso de inteligência artificial para reconhecimento de vagas disponíveis em estacionamentos. 2019. Disponível em: <http://189.8.209.204/handle/satc/373>.

SANTANA, J. P. Uma ferramenta visual para o desenvolvimento de modelos de detecção de objetos com deep learning no ensino superior. 2022. Disponível em: <https://repositorio.ufsc.br/handle/123456789/233125>.

SANTOS, L. F. de A. Classificador baseado em redes neurais convolucionais para algoritmos rmlsa em eons. Universidade de Brasília, p. 18, 2020. Disponível em: <https://bdm.unb.br/bitstream/10483/27263/1/2020_LuizFilipeDeAndradeSantos_tcc.pdf>.

SKEIKA, E. L. Utilização de redes neurais completamente convolucionais para identificação e medição de crânios fetais. *Universidade Tecnológica Federal do Paraná*, p. 43–44, 11 2019. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/4714/1/utilizacaoredesneuraisconvolucionais.pdf>.

SOUZA, V. et al. Análise comparativa de redes neurais convolucionais no reconhecimento de cenas. XI Computer on the Beach, p. 420, 2020. Disponível em: <https://www.researchgate.net/profile/Luan-Silva-7/publication/346593579_Analise_Comparativa_de_Redes_Neurais_Convolucionais_no_Reconhecimento_de_Cenas/links/5fc8d57c92851c00f849d724/

Analise-Comparativa-de-Redes-Neurais-Convolucionais-no-Reconhecimento-de-Cenas.pdf>.

TECH, D. O que é tensorflow? para que serve? 2022.

TOUVRON, H. et al. Fixing the train-test resolution discrepancy. *CoRR*, abs/1906.06423, 2019. Disponível em: <http://arxiv.org/abs/1906.06423>.

TOUVRON, H. et al. Fixing the train-test resolution discrepancy: Fixefficientnet. *CoRR*, abs/2003.08237, 2020. Disponível em: <https://arxiv.org/abs/2003.08237>.

WALT, S. van der et al. scikit-image: image processing in Python. *PeerJ*, v. 2, p. e453, 6 2014. ISSN 2167-8359. Disponível em: <https://doi.org/10.7717/peerj.453>.

WANG, C. et al. Scaled-yolov4: Scaling cross stage partial network. *CoRR*, abs/2011.08036, 2020. Disponível em: <https://arxiv.org/abs/2011.08036>.

WANG, J. et al. Salient object detection: A discriminative regional feature integration approach. *International Journal of Computer Vision*, v. 123, n. 2, p. 251–268, 2017. ISSN 1573-1405.

XIE, Q. et al. Self-training with noisy student improves imagenet classification. *CoRR*, abs/1911.04252, 2019. Disponível em: <http://arxiv.org/abs/1911.04252>.

ZHOU, Y. T.; CHELLAPPA, R. Computation of optical flow using a neural network. ICNN, p. 71–78, 1988.

.1 Apêndices

.1.1 *Yolo*v4

A rede apresenta necessidade de delimitação manual na imagem, ou seja, é necessário realizar a identificação do objeto apresentando sua classe de classificação e sua localização na imagem.

Para isto foi utilizado uma ferramenta chamada de *YoloLabel*, a qual gera um arquivo de texto com o mesmo nome da imagem que estiver em seu sistema no momento, esta utiliza um arquivo de texto para reconhecimento das classes de objetos.

Na ferramenta é possível selecionar a classe pertencente do objeto, e ao utilizar o mouse em uma ação de clicar e arrastar na imagem, é possível desenhar a caixa delimitadora ao redor dele.

O arquivo gerado pela ferramenta apresenta a classe do objeto, a posição X_centralizada, a posição Y_centralizada, a altura e a largura do objeto. Feito o processo manual, a rede necessita da criação de alguns arquivos de configuração para manuseio do *dataset* que está sendo utilizado:

1. Um arquivo do tipo .names, que apresenta todos os nomes das classes, escritas uma em cada linha, presentes no *dataset* para treinamento;
2. Um arquivo do tipo .data, em que apresenta a quantidades de classes a serem treinadas, a localização em diretórios dos arquivos de treinamento, de teste, e o *.names*, é recomendado o direcionamento para um ambiente de *backup*, pois a rede realiza este processo a cada 100 e 1000 interações.
3. Um arquivo de texto para as imagens de treinamento, que referenciará cada arquivo de texto criado, para as imagem de treinamento, pelo *YoloLabel*, cada referência(caminho de diretório) em uma linha do documento.
4. Um arquivo de texto para as imagens de teste, seguindo o mesmo padrão do arquivo de treinamento, entretanto, com as referencias para as imagens de teste.

Para treinamento do modelo *Yolov4* é essencial o *download* e modificação do arquivo de pesos disponível no repositório AlexeyAB/*darknet*:

1. Alteração de lote para o padrão, *batch*=64;
2. Alterar as subdivisões de linha para *subdivisions*=16;
3. Alterar o valor máximo de lotes para *max_batches*=(número de classes*2000), é recomendado que o número mínimo deste parâmetro seja de 6000, pois em casos em

que a quantidade de classes seja menor que 3, pode comprometer o resultado do treinamento de forma negativa, devido a insuficiência de interações(épocas);

4. Alterar o parâmetro *steps*=80% do *max_batches* e 90% do *max_batches*;
5. Alterar o tamanho das imagens na rede, para qualquer múltiplo de 32, neste caso foram mantidos os parâmetros *width*=416 e *height*=416;
6. Alterar o número de classes nas três camadas da rede, *classes*=2;
7. alterar os filtros para *filters*=(número de classes+5)*3, eles são 3 e estão localizados antes de cada [*yolo*] na seção [*convolutional*].

É realizada o *download* dos pesos pré-treinados, estes operam com a utilização do *framework darknet*, existem outros que também implementam o modelo, mas para utilização neste trabalho foi escolhido o *darknet* pela amplitude de suporte e informações sobre sua utilização.

Realizado o *download*, encaixa-se o arquivo de configuração modificado, o *dataset* contento os arquivos de treinamento, teste, nomes de classes(*.names*) e *data*, e inicia-se o treinamento do modelo.

.1.2 *Mask*-RCNN

Assim como a rede *Yolov4*, esta rede também necessitou de manipulação manual, sendo necessário a delimitação dos objetos na imagem. Para isto foi utilizado a ferramenta VGG *Image Annotator*, a qual gera um arquivo com a classe e a posição da objeto na imagem em um arquivo *.jason*.

O algorítimo possui hierarquia específica então todas as imagens de treinamento encontram-se em uma pasta chamada de *train* junto com suas anotações em arquivo.*json*, as imagens de validação encontram no mesmo diretório que a pasta *train* mas em uma pasta chamada *val* junto também com suas anotações.*json*.

Para treinamento, foi utilizado o repositório disponibilizado por Abdulla (2017), o qual consta com os dados para modificação e treinamento em *dataset* customizado, e os pesos da rede pré-treinada no *dataset* COCO.

Para treinamento em *dataset* customizado, é necessário a alteração no arquivo *custom*.py, sendo este o arquivo de configuração de treinamento e *dataset*, dentro deste, na classe chamada *CustomConfig*, é necessário a modificação dos parâmetros:

1. *Name*: para o nome da classe em treinamento, em casos de multiplas classes utilizar um nome fantasia;

2. *Images per GPU*: para casos em que utilizam mais de uma;

3. Número de Classes: *background*+número de classes;

4. *Step per Epoch*: a quantidade de passos para cada época;

Na classe *CustomDataset* dentro da função *load_custom* mantem-se apenas os parâmetros *dataset_dir* e *subset*, é necessário atribuir manualmente as classes que serão processadas, pois o algoritmo não apresenta uma forma dinâmica para identificação pelo arquivo de treinamento então, onde se vê:

"self.add_class("nomefantasia", 1, "nomedaclasse")";

"name_dict = "nome_classe1": 1,"nome_classe2": 2,"name_classe3": 3";

Faz-se necessário também adicionar o local do arquivo de treinamento:

"annotations1 = json.load(open('diretório do arquivo *.json*'))";

Em: "self.add_image()"o primeiro parâmetro é o nome fantasia, então é obrigatório a mudança deste; Dependendo do formato em que as imagens anotadas é necessário reencrever a função *load_mask*, pois esta é escrita para reconhecimento de anotações em polígonos.

Na função *train* altera-se apenas o diretório na chamada da função *load_custom* para o diretório em que se encontram os arquivos de treinamento e validação.

Feito isto, basta apenas modificar o parâmetro *EPOCH* em *model.train()* para a quantidade de épocas necessárias para o treinamento, este modelo possui uma boa resolução mesmo com o processamento de poucas épocas.

Para as medidas de desempenho foi utilizado o repositório de AarohiSingla/*Mask-RCNN-on-Custom-Dataset-2classes-*, onde é desenvolvido funções para resgate dos valores da matriz de confusão.