UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CAMPUS DE ILHA SOLTEIRA

PEDRO AFONSO BARBOSA SISMOTO

ESTUDO E APLICAÇÃO DA VISÃO COMPUTACIONAL EM IMAGENS AÉREAS PARA DETECÇÃO DE EDIFICAÇÕES SITUADAS A JUSANTE DE BARRAGENS

Ilha Solteira
2021

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

PEDRO AFONSO BARBOSA SISMOTO

ESTUDO E APLICAÇÃO DA VISÃO COMPUTACIONAL EM IMAGENS AÉREAS PARA DETECÇÃO DE EDIFICAÇÕES SITUADAS A JUSANTE DE BARRAGENS

Trabalho de Graduação (TG) apresentado como parte dos requisitos para obtenção do grau de Engenheiro Mecânico, junto ao Curso de Graduação em Engenharia Mecânica, da Faculdade de Engenharia da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de Ilha Solteira.

Prof. Dr. Amarildo Tabone Paschoalini
Orientador

Ilha Solteira
2021

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Sismoto, Pedro Afonso Barbosa.
S623e Estudo e aplicação da visão computacional em imagens aéreas para detecção de edificações situadas a jusante de barragens / Pedro Afonso Barbosa Sismoto. -- Ilha Solteira: [s.n.], 2021
90 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Mecânica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2021

Orientador: Amarildo Tabone Paschoalini
Inclui bibliografia

1. Segurança de barragens. 2. YOLOv5. 3. Inteligência artificial. 4. Aprendizado de máquina.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

Pedro Afonso Barbosa Sismoto

ESTUDO E APLICAÇÃO DA VISÃO COMPUTACIONAL EM IMAGENS AÉREAS PARA DETECÇÃO DE EDIFICAÇÕES SITUADAS A JUSANTE DE BARRAGENS

Trabalho de Graduação (TG) apresentado como parte dos requisitos para obtenção do grau de Engenheiro Mecânico, junto ao Curso de Graduação em Engenharia Mecânica, da Faculdade de Engenharia da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de Ilha Solteira.

Trabalho de Graduação **APROVADO** pela comissão examinadora.

COMISSÃO EXAMINADORA:

Prof. Dr. Amarildo Tabone Paschoalini

UNESP – Câmpus de Ilha Solteira (Orientador)

Henrique Edno Leoncini Carvalho

UNESP – Câmpus de Ilha Solteira

Me. Matheus Silva Proença

UNESP – Câmpus de Ilha Solteira

Ilha Solteira – SP
20 de agosto de 2021

Dedico este trabalho à minha mãe, Márcia (in memorian) e, ao meu pai, José Luiz, pela eterna inspiração e orgulho.

AGRADECIMENTOS

Ao professor Dr. Amarildo Tabone Paschoalini, pela confiança e, principalmente, pela oportunidade e privilégio de sua orientação, cujas contribuições perdurarão na minha trajetória pessoal e profissional.

Aos meus pais, Márcia (*in memorian*) e José Luiz, minha fonte de admiração e inspiração, portadores do meu maior amor e orgulho, por sempre dedicarem com esmero seus planos e seus esforços em prol da minha edificação.

À minha companheira de vida, amiga e namorada, Amanda, pelo apoio e pela motivação cuja presença traz conforto e alegria nesta caminhada.

Ao meu primo, João Pedro, e meu amigo, Eduardo, irmãos de coração que estiveram presentes, distantes ou não, em todas as etapas da minha vida.

À todas as pessoas que participaram das etapas deste trabalho e me apoiaram em sua elaboração, contribuindo para o fomento ao conhecimento e à contribuição para a comunidade científica do país.

“Existe apenas uma luz na ciência, e acendê-la em qualquer lugar é iluminar todos os lugares”.

Isaac Asimov (1981)

RESUMO

As diretrizes envolvendo a segurança das regiões situadas a jusante de barragens sofreu alterações substanciais com a atualização da Lei Federal nº 14.334/2010 pela Lei Federal nº 14.066/2020, que estabelecem a Política Nacional de Segurança de Barragens. Dentre as atualizações, destaca-se a identificação de edificações em área vulnerável para elaboração de mapas de inundação, a qual é competência do empreendedor. Trata-se de uma atividade vagarosa e passível de erros, cujo processo poderia ser otimizado por intermédio da visão computacional. Neste cenário, a visão computacional, já consolidada em análises de imagens, possui um papel de destaque em abordagens envolvendo imagens aéreas. Isto posto, o presente trabalho buscou estudar e avaliar o desempenho do algoritmo *YOLOv5* na detecção de edificações utilizando imagens aéreas. Para o treinamento do modelo computacional, foi utilizado um conjunto de dados de disponibilidade gratuita, conhecido como *AIRS*, que reúne uma grande coletânea de imagens aéreas, cuja eficácia do treinamento foi posteriormente avaliada em imagens colhidas em diferentes regiões por meio do *Google Earth*. A realização de testes com diferentes imagens obtidas do *AIRS* resultou em precisões e *recalls* girando em torno de 90%, utilizando hiperparâmetros genéricos, para um conjunto de dados de 1000 imagens rotuladas. Já na utilização das imagens e vídeos de testes obtidos do *Google Earth*, a detecção de edificações enfrentou dificuldades, principalmente em razão das imagens possuírem diferenças significativas em relação às imagens empregadas no treinamento do modelo, fator que compromete sua aplicação direta e, consequentemente, a confiabilidade das detecções das edificações. Logo, para aumentar a eficácia das detecções e incrementar o emprego desta ferramenta, pode-se adequar os dados utilizados no treinamento com imagens semelhantes às amostradas nos dados finais de teste. Dentre outras possibilidades, a integração desta ferramenta com veículos aéreos possibilitaria análises assistidas e consequente otimização do processo de detecção em tempo real. Por fim, destaca-se que a crescente evolução no segmento possibilita avanços promissores, não só na detecção de edificações, mas na caracterização completa dos demais objetos presentes em imagens aéreas.

Palavras-chave: *Segurança de Barragens, YOLOv5, Inteligência Artificial, Aprendizado de Máquina.*

ABSTRACT

The guidelines involving the safety of regions located downstream of dams underwent substantial changes with the updating of Federal Law No. 14.334/2010 by Federal Law No. 14.066/2020, which establish the Dams Safety National Policy. Among the updates, there is the identification of buildings in a vulnerable area for the preparation of flood maps, which is the responsibility of the entrepreneur. It is a slow and error-prone activity, the process of which could be optimized through computer vision. In this scenario, computer vision, already consolidated in image analysis, plays a prominent role in approaches involving aerial images. That said, the present work aimed to study and evaluate the performance of the *YOLOv5* algorithm in the detection of buildings using aerial images. For the training of the computational model, a dataset of free availability, known as *AIRS*, was used, which brings together a large collection of aerial images, whose training effectiveness was later evaluated on images collected in different regions through *Google Earth*. Testing with different images obtained from the *AIRS* resulted in accuracies and recalls around 90%, using generic hyperparameters, for a dataset of 1000 labeled images. In the use of test images and videos obtained from *Google Earth*, the detection of buildings faced difficulties, mainly because the images have significant differences in relation to the images used in the training of the model, a factor that compromises its direct application and, consequently, the reliability of building detections. Therefore, to increase the effectiveness of detections and increase the use of this tool, it is possible to adapt the data used in training with images similar to those sampled in the final test data. Among other possibilities, the integration of this tool with air vehicles would enable assisted analysis and consequent optimization of the detection process in real time. Finally, it is highlighted that the growing evolution in the segment allows promising advances, not only in the detection of buildings, but in the complete characterization of the other objects present in aerial images.

Keywords: *Dam Safety, YOLOv5, Artificial Intelligence, Machine Learning.*

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A detecção de estruturas e edificações, utilizando imagens aéreas, possui grande importância para diversas aplicações (HUERTAS, NEVATIA, 1988). Dentre estas, torna-se pertinente citar a identificação e localização de edificações situadas a jusante de barragens para permitir a ação dos empreendedores, em conjunto com órgãos de proteção e defesa civil, no resgate e apoio envolvendo cenários emergenciais. Além desta aplicação mencionada, a detecção de objetos pode ser utilizada em imagens aéreas para diversos estudos envolvendo o uso e ocupação do solo, acompanhamento e manejo florestal e extração de malhas rodoviárias (NÓBREGA, 2007; SOUZA, 2019; PADARIAN, MINASNY, MCBRATNEY, 2020).

A Lei Federal nº 12.334, de 20 de setembro de 2010, alterada pela Lei Federal nº 14.066, de 30 de setembro de 2020, estabelece a Política Nacional de Segurança de Barragens (PNSB) e, dentre as diretrizes e atribuições, ressalta a importância do reconhecimento e delimitação da região denominada Zona de Autossalvamento (ZAS), bem como de suas edificações e do cadastramento da população situada em áreas vulneráveis (BRASIL, 2010).

Para a delimitação da ZAS, são realizados os chamados “Estudos de Rompimento” para avaliar as áreas atingidas por uma eventual ruptura de barragem. Algumas simulações geram manchas de inundação que abrangem um grande número de edificações situadas em regiões com acesso precário e distribuídas de modo disperso e heterogêneo ao longo da área de estudo, usualmente são utilizadas imagens aéreas para contabilização das edificações.

Porém, durante a análise manual de tais imagens pelas equipes de profissionais envolvidos nesta detecção, duas grandes tratativas cabem ser ressaltadas: a possibilidade de erro ou desatenção humana na identificação dos objetos e o tempo despendido na análise de grandes áreas de interesse.

Como forma de otimizar e automatizar tais detecções, o presente trabalho almeja utilizar recursos computacionais para reduzir esforços humanos e auxiliar analistas na detecção de edificações situadas a jusante de barragens. Para isso, a visão computacional, como ferramenta do aprendizado de máquina, mostra-se uma possível alternativa para cumprir a premissa em questão.

O termo aprendizado de máquina, comumente encontrado na literatura na terminologia inglesa *machine learning*, refere-se à detecção automática de padrões expressivos dispostos em um conjunto de dados, cujas ferramentas permitem, ao sistema, aprender e se adaptar em busca de um resultado ou objetivo desejado (SHALEV-SHWARTZ, BEN-DAVID, 2014). Na visão

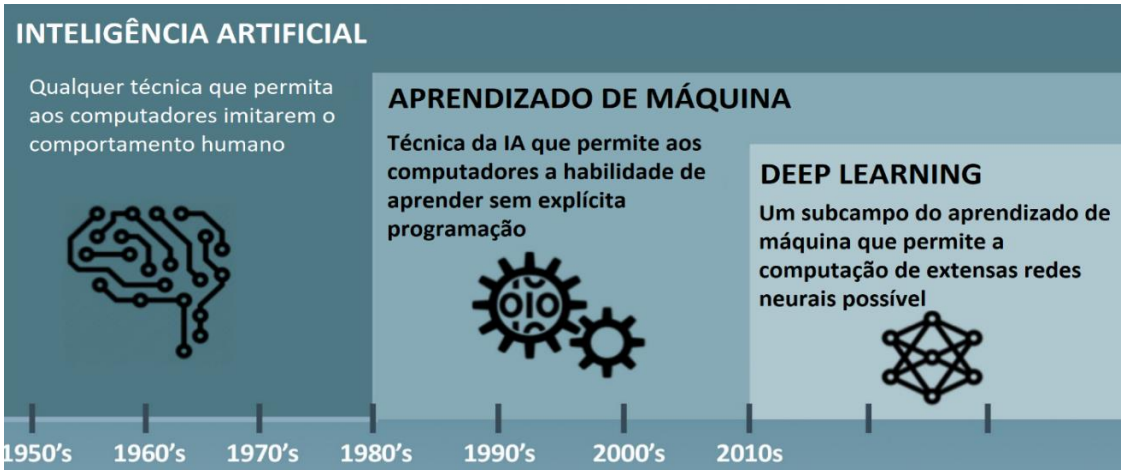
de Samuel (1959), considerado um pioneiro na técnica, o aprendizado de máquina é o estudo que permite aos computadores aprendam tarefas sem explícita programação.

Atualmente, esta técnica se apresenta cada vez mais difundida e acessível, principalmente pelo avanço do poder de processamento do volume de dados digitais, da redução do custo de armazenamento, da disponibilidade de algoritmos e bibliotecas de código-aberto e, também, da visualização de dados para tomada de decisões nos mais diversos setores da atualidade (MUELLER, MASSARON, 2021).

O contexto do aprendizado de máquina pode ser encontrado dentro da perspectiva da Inteligência Artificial (AI, do inglês *Artificial Intelligence*), cujos diversos métodos de aprendizagem devem ser selecionados e especificados de acordo com a natureza do modelo, podendo ser supervisionado ou não, reforçado, ou ainda de profunda aprendizagem, usualmente encontrada sob o termo inglês *Deep Learning* (MULLER, MASSARON, 2021).

O aprendizado profundo é atualmente uma área de pesquisa extremamente ativa que tem obtido grande sucesso em uma vasta gama de aplicações, tais como reconhecimento de padrões fala, visão computacional, mineração de dados e classificação de doenças, sendo muito semelhante ao aprendizado natural do ser humano, principalmente através do desenvolvimento de um senso de percepção e assimilação de informações (ANTWEILER, COPELAND, 2001; GRACE, 2018). A Figura 1 mostra a relação entre estas áreas do campo da Inteligência Artificial.

Figura 1 – Relação entre as áreas do campo de Inteligência Artificial



Fonte: Adaptado (JEFFCOCK, 2018)

Lobo (2020), por sua vez, pontua que a Inteligência Artificial é um ramo da ciência da computação que se propõe a desenvolver sistemas que simulem a capacidade humana na percepção de um problema, identificando seus componentes e, com isso, é capaz de resolver problemas e propor/tomar decisões. Outra definição, citada pelo mesmo autor, indica a criação

de sistemas inteligentes de computação capazes de realizar tarefas sem receber instruções diretas de humanos.

A Inteligência Artificial está transformando nossa relação com a tecnologia e é a base da revolução digital em curso a partir da confluência de tecnologias do mundo digital (Internet das Coisas, *Blockchain* e plataformas digitais), do mundo físico (veículos autônomos, impressão 3D, robótica avançada, novos materiais) e do mundo biológico (manipulação genética) (KAUFMAN, 2018).

O grande matemático inglês Alan Turing, pioneiro na ciência da computação que conhecemos hoje de maneira consolidada, já pontuava em meados de 1947 a capacidade de máquinas desempenharem comportamentos semelhantes à inteligência humana. Em uma das suas mais famosas publicações, *Computing Machinery and Intelligence*, propôs indagações referentes ao que seria entendido por este comportamento pensante, bem como quais as condições necessárias para considerar uma máquina inteligente (MCCARTHY, 2004).

Arthur Lee Samuel, outro grande propulsor de desenvolvimentos no campo da Inteligência Artificial e responsável por popularizar o termo *machine learning*, publicou em 1959 um artigo no jornal de pesquisa e desenvolvimento da IBM enaltecendo a capacidade de computadores desempenharem atividades de maneira superior aos seus próprios desenvolvedores, como no caso do jogo de tabuleiro de damas, objeto de estudo de seu artigo (SAMUEL, 1959).

Como ressalta Silva (2017), os avanços nestas áreas, principalmente devido à criação de novos algoritmos de treinamento de redes profundas e *hardware* mais potente, possibilitaram o desenvolvimento de redes inteligentes extremamente complexas, capazes de realizar tarefas cognitivas de alto nível. Tal evolução abriu um leque de possibilidades de aplicações, que hoje já estão ao nosso alcance, como carros autônomos, assistentes virtuais e classificadores automáticos de imagens.

Grandes outros exemplos de suas aplicações também podem ser mencionados, como o *Google Translate*, *DeepFace*, *IBM Watson*, bem como os aplicativos que utilizam linguagem natural para fazer recomendações e realizar ações como a *Siri*, *Google Now*, *Cortana* e *Alexa*, respectivamente da *Apple*, *Google*, *Microsoft* e *Amazon* (HOY, 2018).

Como menciona Haykin (2007), a utilização de redes neurais permite decompor uma tarefa complexa em diversas tarefas simplificadas, tornando possível a solução de problemas que, até então, seriam intratáveis. Neste contexto, utilizar-se-á tais ferramentas na detecção de edificações para estudar e avaliar, entre outras questões, o desempenho, a praticidade e a viabilidade de implantação.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Estudar a arquitetura, ferramentas e recursos computacionais disponíveis no campo da visão computacional para implementar um algoritmo, em um conjunto de imagens aéreas personalizadas, visando avaliar o desempenho da detecção de edificações situadas a jusante de barragens.

2.2 OBJETIVOS ESPECÍFICOS

- Estudar os principais parâmetros constituintes do aprendizado de máquina voltados para o campo da visão computacional;
- Construir e manipular um conjunto de dados com imagens aéreas personalizadas;
- Implementar o algoritmo *YOLOv5* voltado para a visão computacional na detecção de edificações;
- Avaliar o desempenho do modelo e gerar imagens e vídeos provenientes das detecções;
- Fomentar discussões acerca da otimização do modelo e demais parâmetros embarcados;
- Avaliar a aplicabilidade em cenários reais envolvendo a segurança da população residente a jusante de barragens.

3 REVISÃO BIBLIOGRÁFICA

3.1 PROCESSAMENTO DIGITAL DE IMAGENS

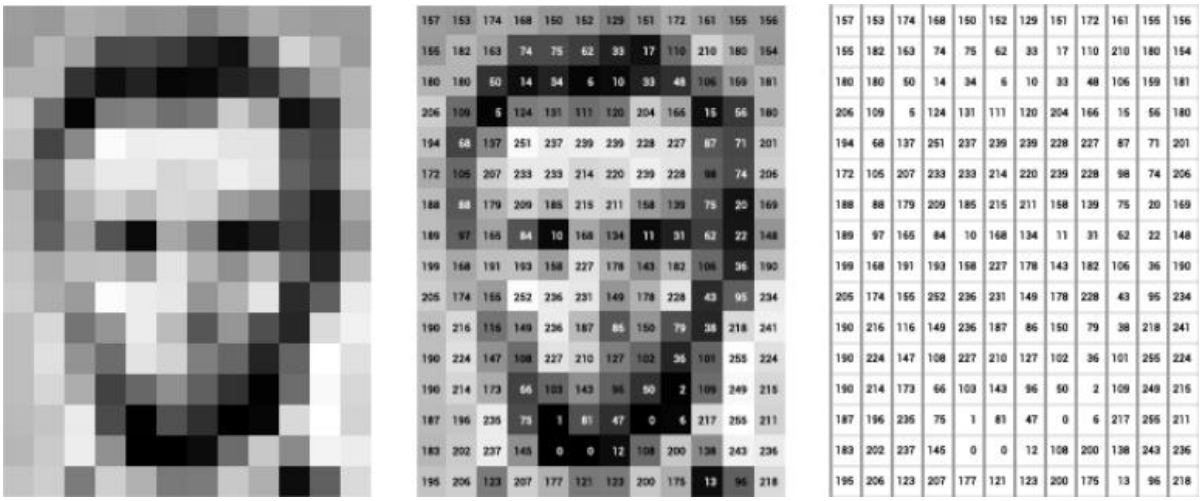
A forma, tamanho, textura e as cores constituem características importantes das imagens que estão sendo utilizadas e manipuladas em diversas aplicações envolvendo o cenário de visão computacional na detecção de objetos e classificações de imagens (ACHARYA; RAY, 2005). Deste modo, o processamento de imagem consiste na aplicação de diversas operações para extrair informações relevantes à área de interesse (CHIUCHISAN, 2013).

3.1.1 Sistemas e composições das imagens

As imagens coloridas, no meio digital, são formadas por três canais (ou bandas) independentes, representando, respectivamente, a magnitude referente às diferentes colorações presentes em sua composição. Esta composição, por sua vez, é conhecida como RGB, termo inglês que refere-se às iniciais correspondentes às cores vermelho (*Red*), verde (*Green*) e azul (*Blue)*. Já as imagens conhecidas popularmente como “preto e branco” podem ser representadas tanto em uma escala binária quando em uma escala cinzenta composta de uma única matriz, de modo que cada elemento representa um valor de intensidade variando do preto ao branco (KUMAR, VERMA,2010; VIJAYAN, JOHN, SEN, 2014).

Como exemplo, a Figura 2 ilustra uma imagem em escala cinzenta. Nesta imagem, a magnitude da coloração é feita por meio do valor associado à cada *pixel* sendo que, para uma imagem de 8-bits, esta magnitude varia do valor zero (preto) ao valor 255 (branco), ou seja, existe a possibilidade da magnitude de cada *pixel* assumir 256 valores distintos, sendo que qualquer valor situado entre este intervalo representará uma intensidade da cor cinza (SCURI, 1999).

Figura 2 - Representação da matriz referente aos valores de cada pixel da imagem

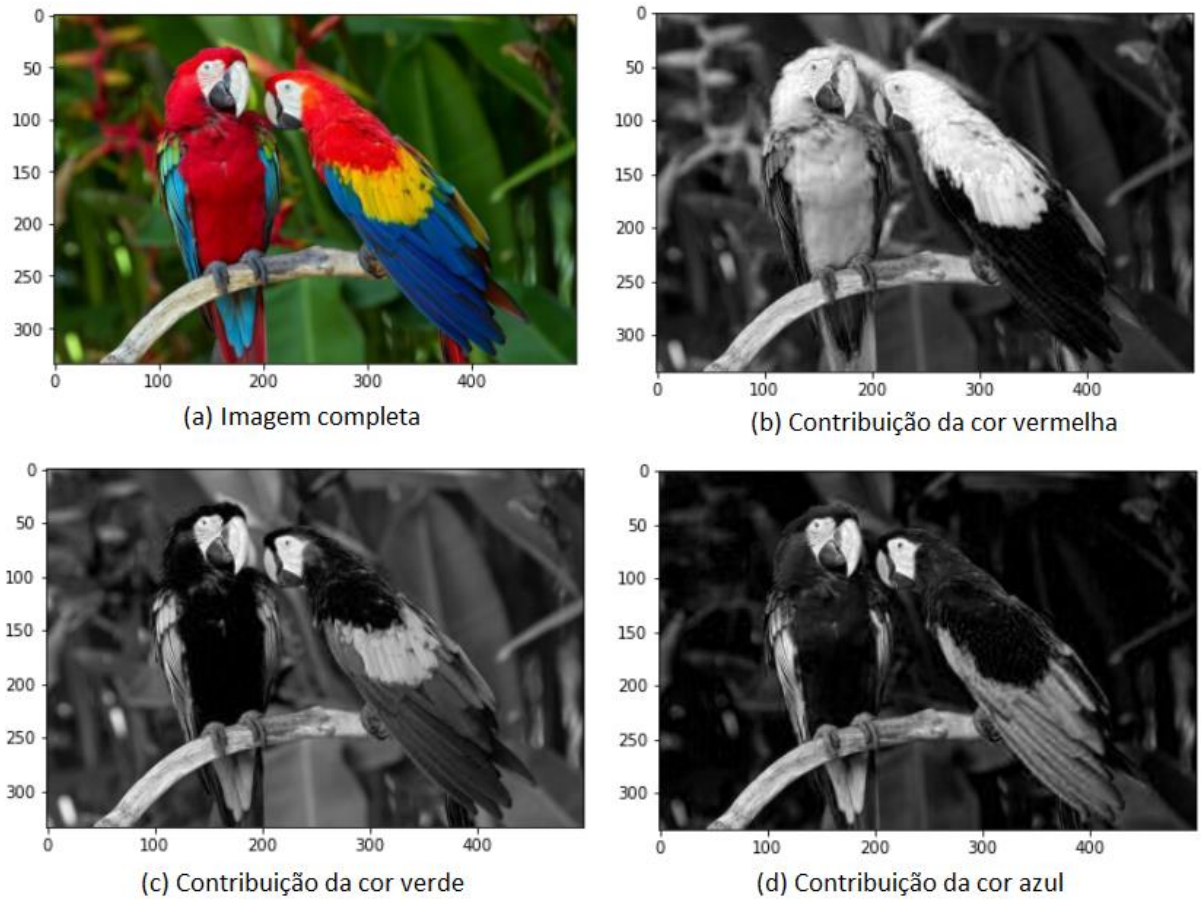


Fonte: STANFORD, [s.d]

As imagens coloridas possuem, em cada um dos canais, um comportamento semelhante ao demonstrado pela Figura 2, porém, ao invés da variação ser entre a intensidade das cores preto e branco, o efeito é referente às três cores que compõem a imagem. Ou seja, para cada um dos canais, haverá uma matriz com valores referentes à intensidade de cada cor em determinada posição, cuja concatenação resultará na imagem final (AQEL, ALQADI, 2018).

Para ilustrar a composição de uma imagem RGB, elaborou-se a Figura 3. Nesta figura, cada uma das três matrizes foram propositalmente isoladas e representadas em uma escala cinzenta sendo que, quanto maior a intensidade da coloração branca, maior será a intensidade e a contribuição da respectiva cor na imagem completa.

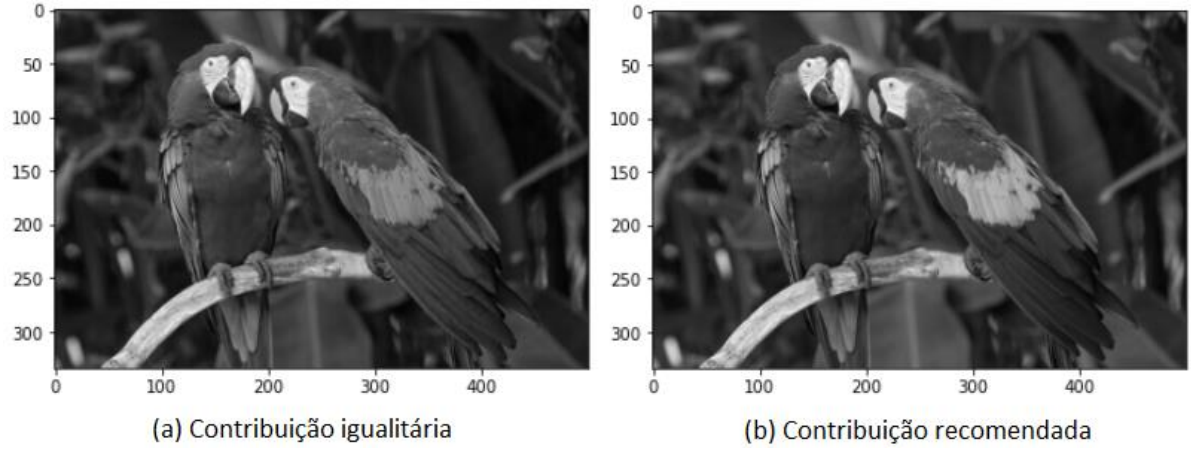
Figura 3 - Exemplo da contribuição de cada cor na imagem completa



Fonte: Elaborado pelo autor.

No entanto, nota-se que para elaborar uma imagem na escala cinzenta a partir de uma imagem RGB, alguns cuidados devem ser tomados. De certo modo, para transformar a imagem original para a escala cinzenta, não é recomendado utilizar uma contribuição igualitária de cada componente, ou seja, a elaboração desta imagem não deve levar em consideração um terço da magnitude de cada contribuinte. Para esta conversão, existem algumas contribuições singulares mais adequadas, como por exemplo a proporção de cerca de 29,89% da contribuição da cor vermelha, 58,70% da cor verde e 11,40% da cor azul (LU, XU, JIA, 2010; DHAL *et al.*, 2019). A fim de ilustrar qualitativamente a diferença entre a contribuição igualitária de cada cor na composição da imagem na escala cinzenta em comparação com a proporção recomendada anteriormente, mesmo que sutil ao olhar humano, elaborou-se a Figura 4.

Figura 4 - Exemplificação da diferença entre as contribuições na conversão da imagem RGB para a escala cinzenta



Fonte: Elaborado pelo autor.

Vale ressaltar que a conversão das imagens RGB para a escala cinzenta é realizada de modo relativamente simples, fator que não é verdadeiro para o cenário contrário visto que envolve mais do que uma simples transformação, processo comumente conhecido como colorização (LEVIN, LISCHINSKI, WEISS, 2004; ZHANG, ISOLA, EFROS, 2016).

Também cabe pontuar que algumas imagens, em específico, possuem mais do que três canais em sua composição. Estas imagens multibandas, ou hiperespectrais, podem ser comumente encontradas em aplicações envolvendo sensoriamentos aeroespaciais, por exemplo (WEI, DOBIGEON, TOURNERET, 2015; FERRARIS *et al.*, 2017).

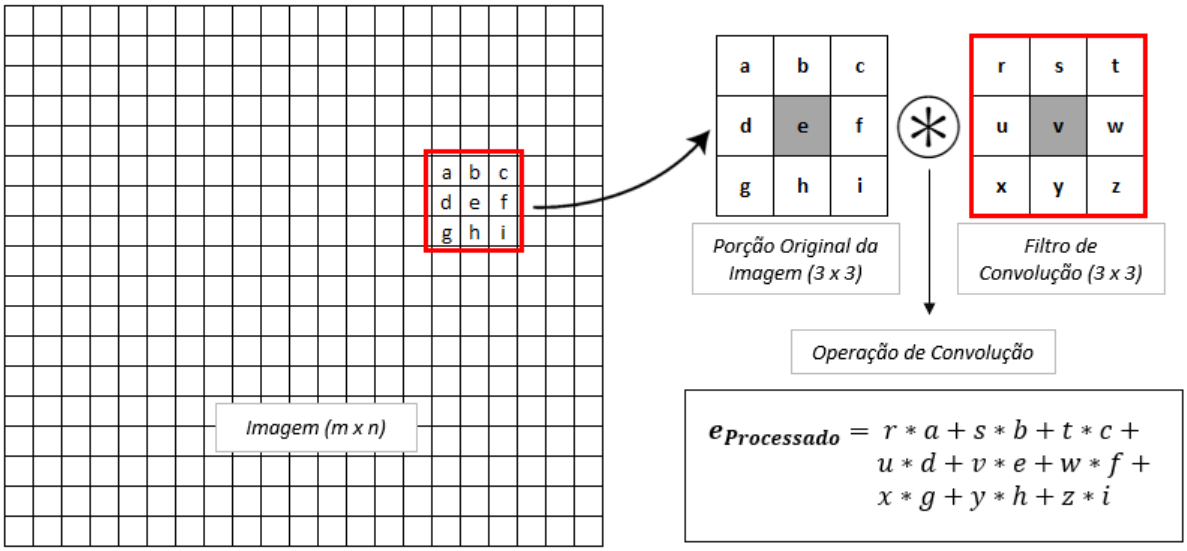
3.1.2 Convolução, filtros e máscaras

A filtragem de imagens digitais consiste em uma ou mais operações que são empregadas com o intuito de realçar alguma característica específica da imagem, concebidos muitas das vezes para e extração de padrões e singularidades relevantes para estudos, bem como para adequação e condicionamento gráfico para uso nas mais variadas aplicações (FERGUS, PERONA, ZISSERMAN, 2004; QUEIOZ, GOMES, 2006; WU, 2010).

De modo genérico, a operação conhecida como convolução é atribuída ao processo de filtragem de porções sequenciais efetuada pelo deslocamento progressivo de uma máscara (também conhecida como janela móvel ou *kernel*) em intervalos pré-estipulados por toda a extensão da imagem, resultando em efeitos relativos às características do filtro de convolução em questão (JESUS, COSTA JR, 2015). Considerando aspectos matemáticos, a operação de convolução possui algumas diferenças com este processo envolvendo a rotação da máscara, porém, em razão da popularização e principalmente convenção, frequentemente o termo é

utilizado na mesma denotação da operação de filtragem, possuindo também uma íntima relação com a operação de correlação cruzada (SHYNK, 1992; VORBURGER, 2011; GALOOGAHI, SIM, LUCEY, 2013). O modo como é feita a operação do filtro de convolução está melhor exemplificado na Figura 5.

Figura 5 - Exemplo da aplicação de um filtro de convolução

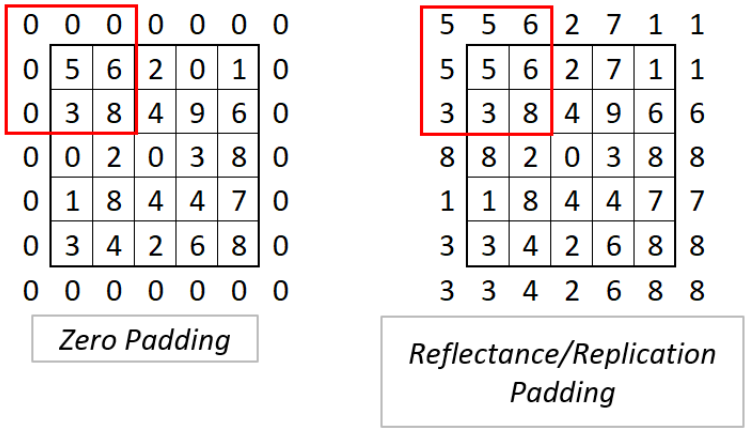


Fonte: Elaborado pelo autor.

Tomando o exemplo ilustrado na Figura 5, o filtro de convolução pode ser passado em intervalos estipulados ao longo da extensão da imagem, cuja variável responsável pelo "salto" do filtro é conhecida como *stride*. De modo genérico, o valor de *stride* é unitário e o desenvolvimento do filtro é feito à cada *pixel*, porém, para reduzir esforços e consequente diminuição do volume de saída, pode-se atribuir valores maiores que a unidade ao *stride* (LI *et al.*, 2015; ZANIOLO, MARQUES, 2020).

Uma condição relevante, que despende atenção por parte do analista, é observada durante a passagem da máscara nos elementos que situam-se nas periferias da imagem. Visto que não há elementos pré-existentes para a realização da operação de convolução, dois critérios são comumente adotados: entradas nulas, conhecido como *zero padding* ou entradas espelhadas, mais conhecido como refletância/replicação (NESTMEYER, GEHLER, 2017). A Figura 6 ilustra estes dois exemplos do efeito de borda.

Figura 6 - Exemplo das possibilidades mais comuns para lidar com os efeitos de borda



Fonte: Elaborado pelo autor

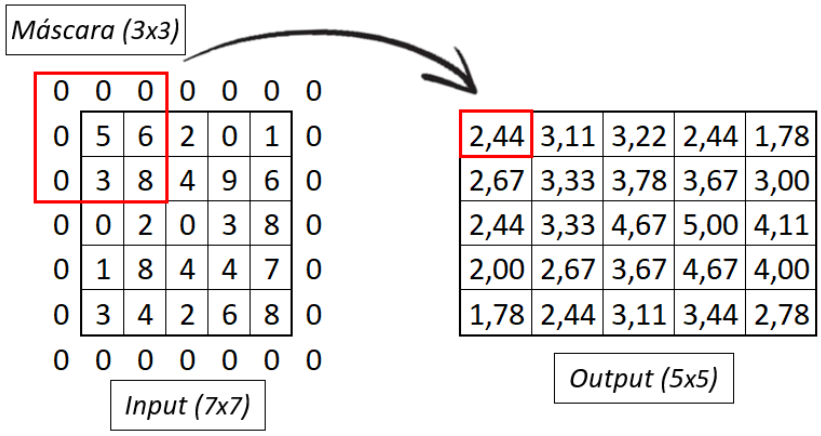
Diversas máscaras podem ser adotadas para realçar as aspectos relevantes das imagens para extração de padrões e características intrínsecas, sendo que, neste estudo, algumas máscaras específicas bastante difundidas foram analisadas (QUEIROZ, GOMES, 2006), começando inicialmente pela máscara responsável pelo efeito de suavização.

3.1.2.1 Suavização

A máscara responsável por promover o efeito de suavização, responsável pela característica embaçada, turva ou até mesmo desfocada das imagens, é amplamente utilizada para reduzir variações bruscas de intensidade e atenuar ruídos (BAI, ZENG, GULBERG, 2000; CHEN *et al.*, 2006).

Para alcançar este efeito de suavização, pode-se utilizar, por exemplo, a média aritmética dos valores dos *pixels* em uma vizinhança. A moda ou a mediana também são opções para desempenhar este efeito, bem como uma máscara usada de modo abrangente guiada pela distribuição Gaussiana dos valores dos *pixels* (QUEIROZ, GOMES, 2006; GEDRAIT, HADAD, 2011). A Figura 7 ilustra um exemplo da aplicação da máscara envolvendo a média aritmética dos valores de cada *pixel.*

Figura 7 - Exemplo da aplicação da máscara com a operação de média aritmética

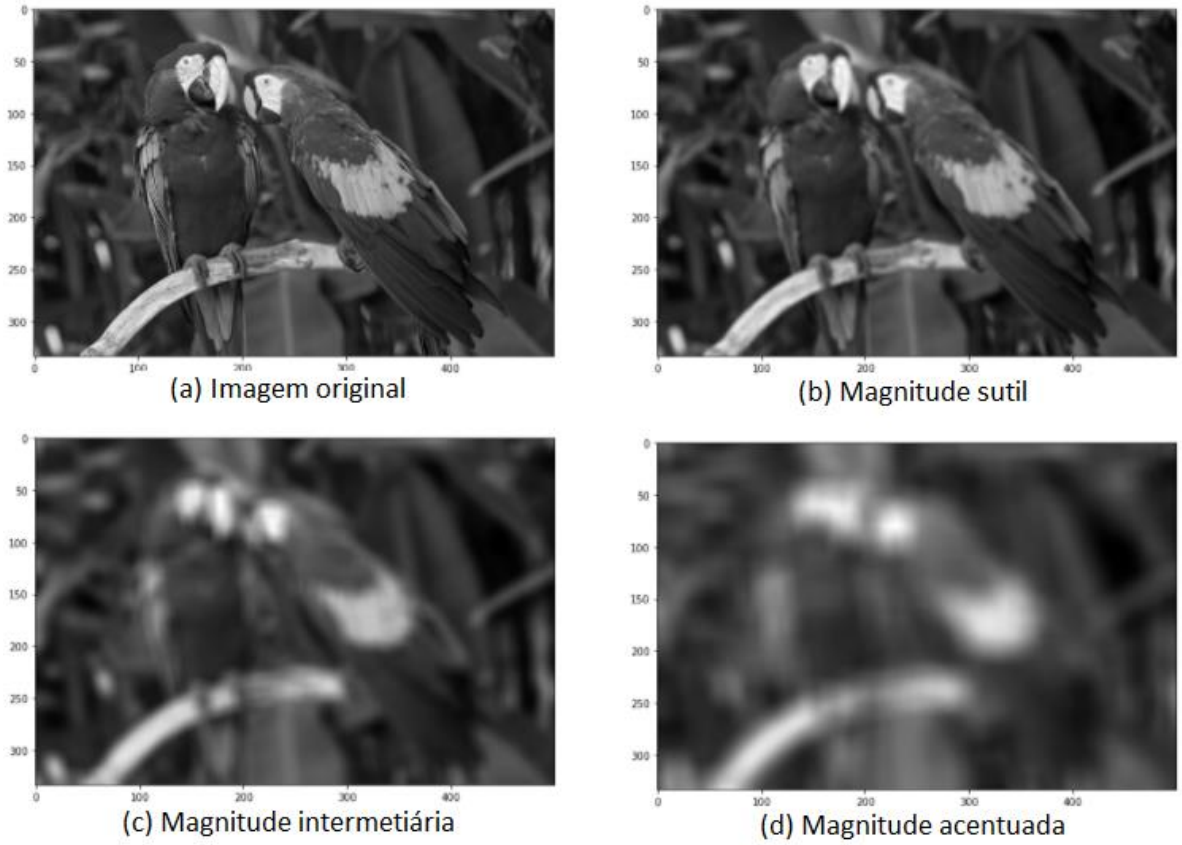


Fonte: Elaborado pelo autor.

Cabe pontuar que, para manipular a magnitude do efeito provocado pelo filtro pode-se alterar a dimensão da máscara. Para maiores efeitos de ofuscação pode-se aumentar a dimensão da máscara e, para efeitos mais atenuados, pode-se reduzir sua dimensão. Também é valido mencionar que os efeitos também podem ser amplificados, de modo independente da dimensão da máscara, passando-a mais de uma vez de modo progressivo e cumulativo ao longo da imagem (SHEN, CASTAN, 1992; HE, SUN, TANG, 2010).

A fim de ilustrar os diferentes efeitos da aplicação, elaborou-se, utilizando a mesma imagem dos exemplos anteriores, o resultado para diferentes dimensões de máscaras na Figura 8. Ressalta-se que a imagem original possui 334x500 *pixels* e as máscaras utilizadas nos itens (b), (c) e (d) possuem, respectivamente 5x5, 15x15 e 30x30 *pixels*.

Figura 8 - Exemplos do efeito do incremento das dimensões das máscaras



Fonte: Elaborado pelo autor.

Já o processo envolvendo a nitidez das imagens, consiste em aguçar as características, contornos e contrastes das imagens de modo à evidenciar, como um todo, os aspectos da imagem, tornando-a mais sensível. Este processo também é conhecido pela aplicação de filtros passa-alta, proporcionando um efeito inverso ao ocasionado pela aplicação do filtro de suavização, ou seja, de filtros passa-baixa. Neste caso, utiliza-se também máscaras voltadas para a detecção de bordas para proporcionar estes efeitos (CRUZ, 2011).

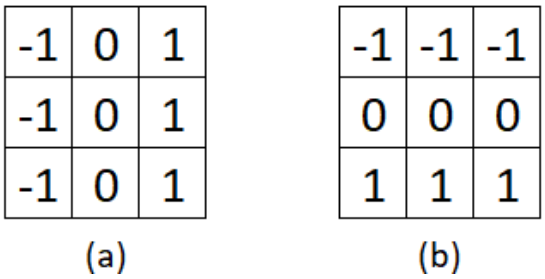
3.1.2.2 Detecção de bordas

O processo de detecção de bordas é uma ferramenta fundamental utilizada nas aplicações de processamento de imagens para extração de características por meio da diferenciação das fronteiras de um objeto de desejo em relação à imagem que se encontra (AL-AMRI, KALYANKAR, KHAMITKAR, 2010). Deste modo, este pré-processamento das imagens conta com a aplicação de filtros especificamente concebidos para realçar a fronteira dos objetos de acordo com a variação abrupta de intensidade em relação à vizinhança (ROBINSON, 1977).

Para cada *pixel* da imagem é computado um vetor gradiente que determina uma magnitude referente à orientação da direção da troca de intensidade, cujo valor será significativamente maior para regiões onde esta variação é mais significativa, resultando no que é conhecido como um vetor responsivo. Este vetor, por sua vez, pode ser submetido à um limiar usualmente conhecido como *threshold* de histerese para atenuar ou amplificar a magnitude de atuação, geralmente utilizado para eliminar respostas que não são significativas à classificação (ROBINSON, 1977; ZIOU, TABBONE, 1998).

Dois grandes exemplos de máscaras utilizadas para convolucionar uma imagem para detecção das bordas estão ilustrados na Figura 9 sendo que, no primeiro exemplo, as linhas verticais da imagem são evidenciadas, enquanto no segundo exemplo são evidenciadas as linhas horizontais (SHRIVAKSHAN, CHANDRASEKAR, 2012). Nota-se que a matriz ilustrada na Figura 9 (b) é justamente a matriz transposta ilustrada na Figura 9 (a).

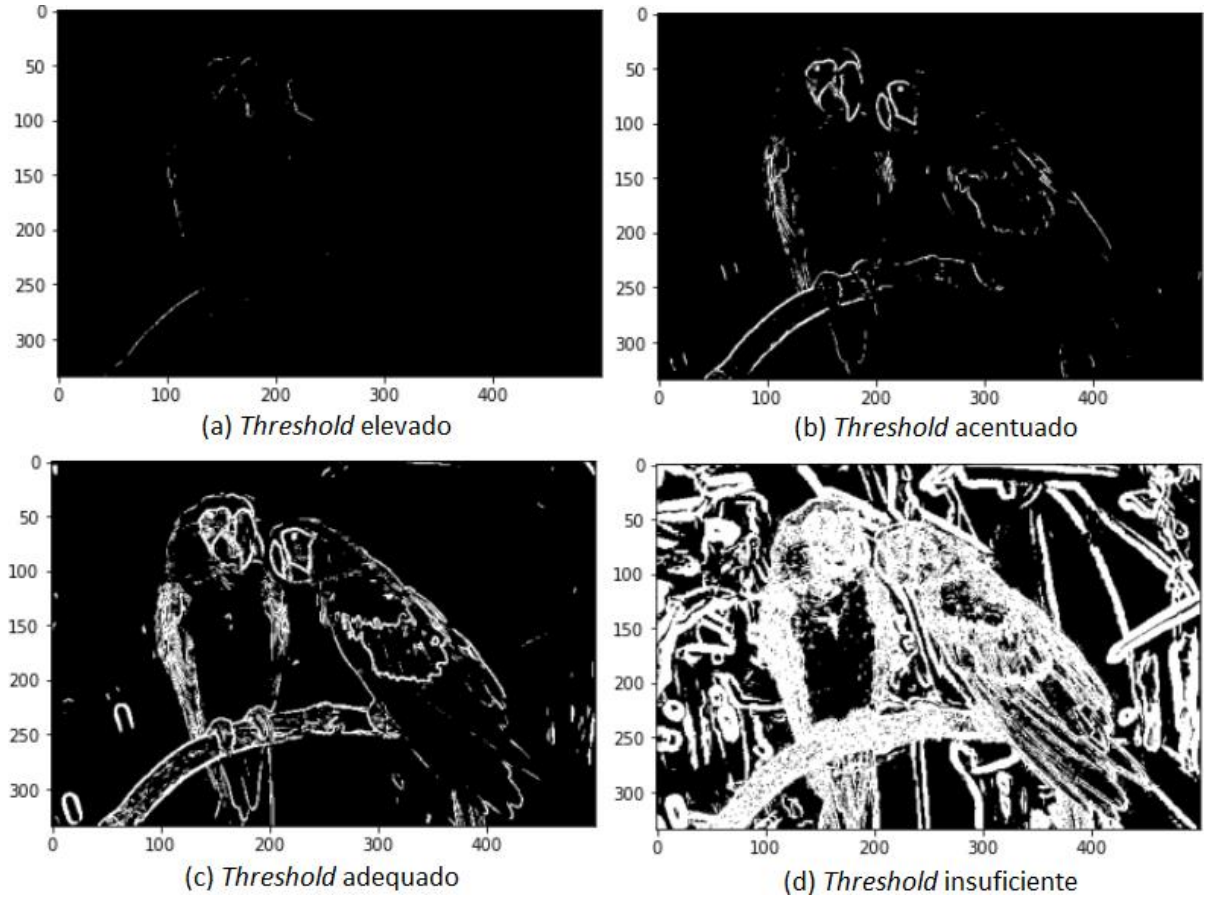
Figura 9 - Exemplo de máscaras utilizadas para detecção de linhas verticais e horizontais



Fonte: Adaptado (SHRIVAKSHAN, CHANDRASEKAR, 2012).

Portanto, o processo consiste em convolucionar a imagem desejada com um filtro desejado, utilizando as técnicas para resolver os problemas de borda descritos anteriormente e estabelecer um valor de *threshold* para atenuar o grau de resposta do vetor de resposta. A Figura 10 ilustra o efeito da convolução das imagens para diferentes valores de *threshold* a fim de ilustrar o impacto no resultado final.

Figura 10 - Exemplo da variação da magnitude do *threshold*

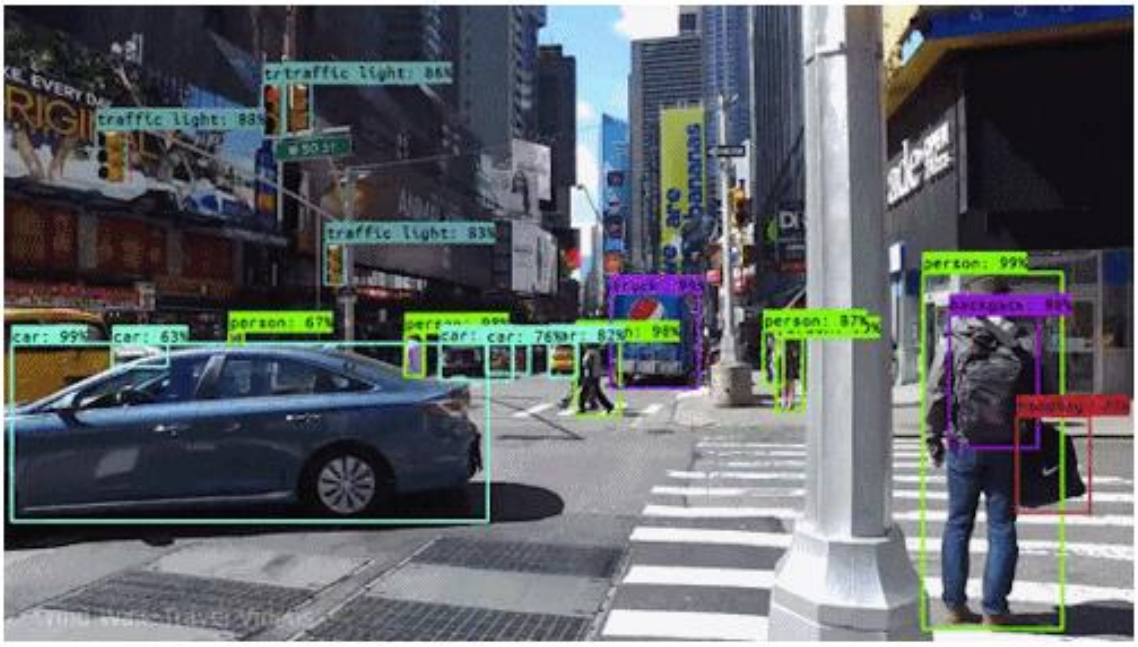


Fonte: Elaborado pelo autor.

3.2 DETECÇÃO DE OBJETOS

Dada uma imagem (ou um vídeo) e uma ou mais classes pré-especificadas, relacionada à objetos de interesse, a detecção de objetos visa classificar e localizar objetos de interesse presentes, cujo processo consiste em duas grandes etapas: o reconhecimento do objeto por meio da atribuição de um rótulo relacionado à uma classe particular e a localização do objeto na imagem ou vídeo em questão usualmente por meio da projeção de um polígono conhecido como *bounding box* (PAPAGEORGIOU, OREN, POGGIO, 1998; SZEGEDY, TOSHEV, ERHAN, 2013; BORJI *et al.*, 2015). A Figura 11 ilustra um exemplo da aplicação desta técnica com a classe dos objetos identificados e sua localização por *bounding boxes*.

Figura 11 - Exemplo da detecção de objetos utilizando *bounding boxes*



Fonte: PYTHON AWESOME, 2021.

Há diferentes métodos para definir o modo como a *bounding box* é projetada na imagem, bem como suas características e formas, porém, usualmente é utilizado um polígono de quatro arestas que pode ser definido de dois modos diferentes (SZEGEDY, TOSHEV, ERHAN, 2013;YU *et al.*, 2016):

a) definindo um ponto no canto superior direito e outro ponto no canto inferior esquerdo do objeto detectado;
b) definindo um ponto médio no objeto e, posteriormente, a largura e a altura do polígono.

Cabe ressaltar que estas *bounding boxes* podem ser representadas também por geometrias espaciais, preenchidas ou não, regulares ou não, apresentando diferentes cores, espessuras e informações de acordo com o critério do desenvolvedor (MOUSAVIAN *et al.*, 2017; XU, ANGUELOV, JAIN, 2018).

A classificação de objetos consiste em uma sequência de eventos que tem a premissa de extrair as características salientes e significativa das imagens, armazenando-as individualmente em um vetor de características, usualmente conhecido como vetor de características, ou então, *descriptor*, para posterior classificação (BLUM *et al.*, 2012).

Para a detecção de um único objeto existem somente duas categorias possíveis de enquadramento, sendo a primeira categoria representada pela presença do objeto em questão na imagem em análise e a segunda categoria representada pela ausência do objeto. Então, para treinar o classificador, faz-se necessário possuir diversas imagens contendo o objeto de desejo

(imagens positivas) e diversas imagens que não possuem o objeto que se deseja detectar (imagens negativas) a fim de propiciar uma nítida diferença entre a presença ou ausência deste objeto na imagem (WANG *et al.*, 2013).

Em relação às imagens disponíveis para o treinamento, como a maioria das classificações está ligada de forma íntima às dimensões dos objetos, recomenda-se que estas imagens possuam a mesma dimensão a fim de trazer fidelidade análise. O mesmo não se aplica para as imagens cuja análise será aplicada, que pode ser significativamente maior que as imagens de treinamento e conter vários objetos distintos (XIAO, RASUL, VOLLGRAF, 2017).

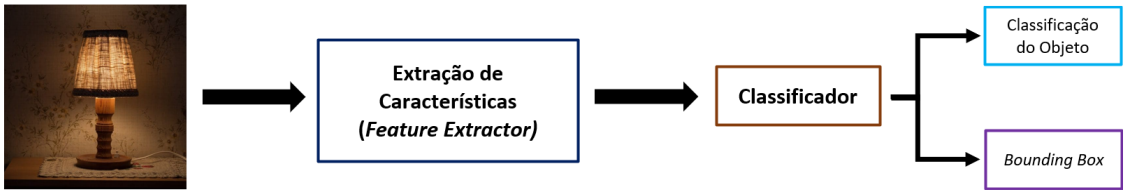
Uma técnica clássica envolvendo a detecção de objetos consiste em deslizar uma janela em intervalos pré-estipulados por toda a imagem - cuja terminologia inglesa *Slinding Windows* refere-se justamente à este processo - para verificar a existência do objeto de interesse na seção analisada, sendo que a dimensão da janela e o intervalo de deslizamento são parâmetros que impactam diretamente na eficiência, precisão e no esforço computacional do resultado (SHURAN, JIANXIONG, 2014; LAMPERT, BLASCHKO, HOFMANN, 2017).

Um dos principais parâmetros da janela diz respeito às suas dimensões que, em geral, sugere-se utilizar as mesmas dimensões das imagens utilizadas no treinamento, de modo que ao deslizar progressivamente a janela, o objeto utilizado no treinamento ocupe boa parte de sua extensão, favorecendo o reconhecimento por meio da correspondência de características (SUDOWE, LEIBE, 2011; HOSANG *et al.*, 2015). Também é comum observar que a dimensão das janelas é composta por números ímpares, justamente pela finalidade de possuir um *pixel* central bem definido.

Durante estas operações, torna-se comum o surgimento de problemas envolvendo o tamanho e orientação das imagens, que será melhor tratado no item 3.2.1. No final do processo, haverá um mapa referente à passagem da janela ao longo da imagem que carregará uma probabilidade das porções analisadas conterem o objeto desejado acompanhado de valores de *threshold* para ajuste da detecção e posterior classificação (ZHAO *et al.*, 2019).

Por fim, elaborou-se a Figura 12 para ilustrar, de modo esquemático, a sequência de eventos envolvendo a detecção de objetos, também conhecida como *pipeline*.

Figura 12 - Sequência de eventos (*pipeline*) da detecção de objetos



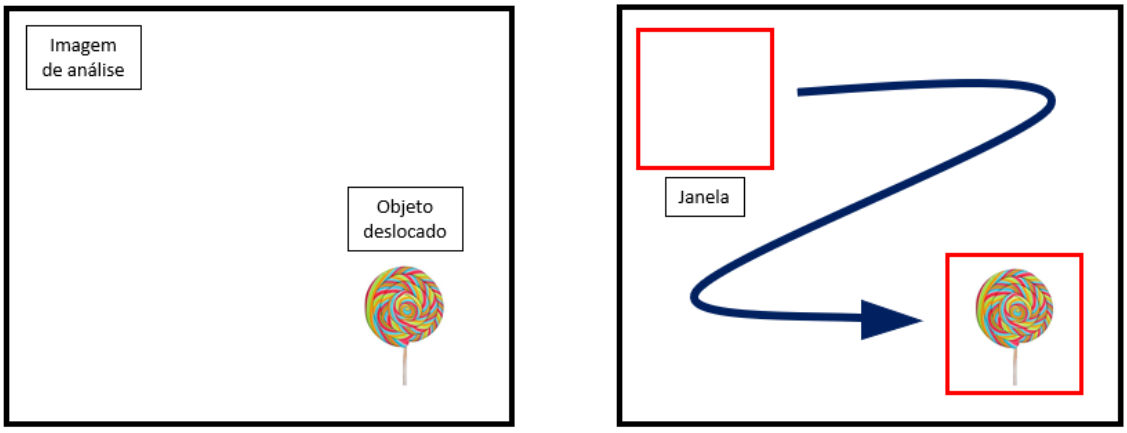
Fonte: Adaptado (PEI *et al.*, 1995).

3.2.1 Invariância translacional, rotacional e escala

Uma propriedade desejável para um modelo de detecção de objetos está na capacidade do detector tornar-se independente da disposição dos objetos nas imagens, como por exemplo, deslocamentos em relação ao centro da imagem, rotações do objeto em relação à um eixo considerado natural para sua posição e o problema da escala de imagens (FANG, HAUSLER, 1990).

Para o problema que envolve o deslocamento do objeto em relação ao centro da imagem, como ilustra a Figura 13, pode-se evitá-lo por meio da passagem da janela deslizante por toda a imagem de modo progressivo e discretizado em pequenos intervalos, de modo que, em algum momento, a janela passará pelo objeto e acusará a incidência das características familiares.

Figura 13 - Esquematização da passagem da janela ao longo da imagem para encontrar o objeto



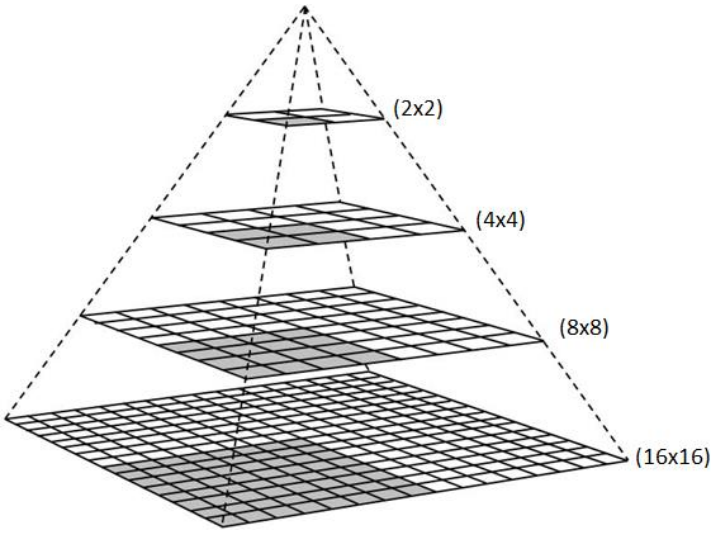
Fonte: Elaborado pelo autor.

Já para a problemática envolvendo a escala dos objetos pode ser solucionada por meio de uma sequência de passadas da janela do seguinte modo (STARCK, MURTAGH, BIJAOUI, 1998):

a) Primeiramente, a janela é deslizada normalmente sobre a imagem de análise e é gerado um mapa inicial de classificação;
b) Em sequência, pode-se alterar o tamanho da imagem de análise, reduzindo ou aumentando suas dimensões, de modo que o tamanho da janela mantém-se constante, e então é repetido o processo de deslizamento da janela. Neste passo também é gerado um mapa de classificação referente à esta escala específica;
c) Repete-se o processo de alteração da escala da imagem de análise, mantendo o tamanho da janela constante, até que as varreduras demonstrem suficiência para a análise. Assim como nos passos anteriores, também é gerado um mapa de classificação respectivo à cada escala.

Portanto, ao final do processo, pondera-se os mapas de acordo com a magnitude dos valores positivos para a detecção do objeto desejado. Este processo é conhecido como Pirâmide Gaussiana e, em geral, despende um esforço computacional elevado e é considerado um método que utiliza “força bruta” para lidar com a problemática envolvendo a escala das amostras utilizadas no treinamento (ADELSON *et al.*, 1984; STRENGERT, KRAUS, ERTL, 2016). A Figura 14 ilustra esquematicamente a alteração nas escalas das imagens, que remetem à geometria piramidal em razão destas alterações e dá o nome ao método.

Figura 14 - Esquematização da Pirâmide Gaussiana



Fonte: Adaptado (SOUSA JÚNIOR, 2007)

Por fim, para contornar a problemática das imagens rotacionadas, pode-se tanto rotacionar as imagens utilizadas no treinamento quanto rotacionar a imagem de análise. Neste segundo caso, o processo é análogo à problemática de escala, ou seja, varre-se a imagem inteira com a janela gerando o mapa de classificação, rotaciona-se a imagem de análise e repete-se o processo para um número julgado adequado de ângulos em relação à imagem original. Vale ressaltar que também pode-se rotacionar somente a janela em diferentes ângulos e repetir a análise, ponderando as respostas no final do processo de acordo com as máximas contribuições de cada cenário (FANG, HAUSLER, 1990; STARCK, MURTAGH, BIJAOUI, 1998). Para ilustrar o problema de rotação, elaborou-se a Figura 15.

Figura 15 - Exemplo de objetos semelhantes rotacionados na mesma imagem

Fonte: Elaborado pelo autor.

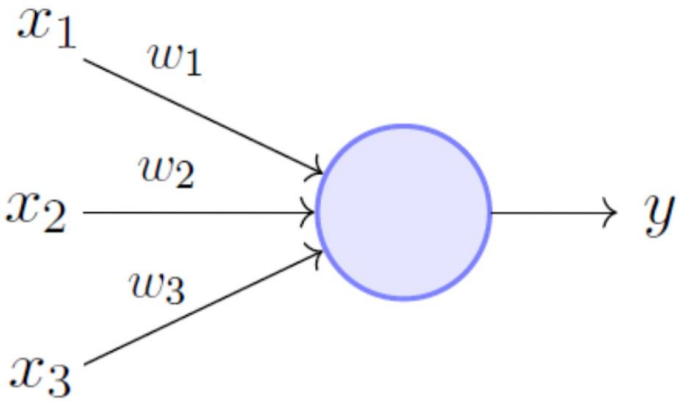
Outras técnicas que lidam com as problemáticas supracitadas, como, por exemplo o algoritmo *YOLO*, contribuinte de grandes avanços no cenário de visão computacional, serão melhores detalhadas na seção 3.4.

3.3 REDES NEURAIS ARTIFICIAIS

Redes neurais artificiais, também conhecidas na terminologia inglesa *Artificial Neural Networks*, representam modelos computacionais inspirados no sistema nervoso central dos seres vivos e foram desenvolvidos para tarefas que envolvem reconhecimento, semelhante a natureza biológica do cérebro (MÜLLER, REINHARDT, STRICKLAND, 1995).

Análogo às estruturas biológicas, as redes neurais possuem elementos chamados *perceptrons* que, em suma, referem-se ao modelo matemático de um neurônio biológico, cujas terminologias muitas vezes são utilizadas de modo intercambiável (HAGAN, DEMUTH, 1999; RIEDMULLER, LERNEN, 2014). Estes neurônios realizam operações nos dados recebidos, também chamada de combinações, devolvendo uma reposta baseada no modo como foram orientá-los à respondê-la (ROSENBLATT, 1958; ZENG, YEUNG, 2001). O modelo de *perceptron* desenvolvido por Rosenblatt (1958) pode ser observado de modo ilustrativo da Figura 16.

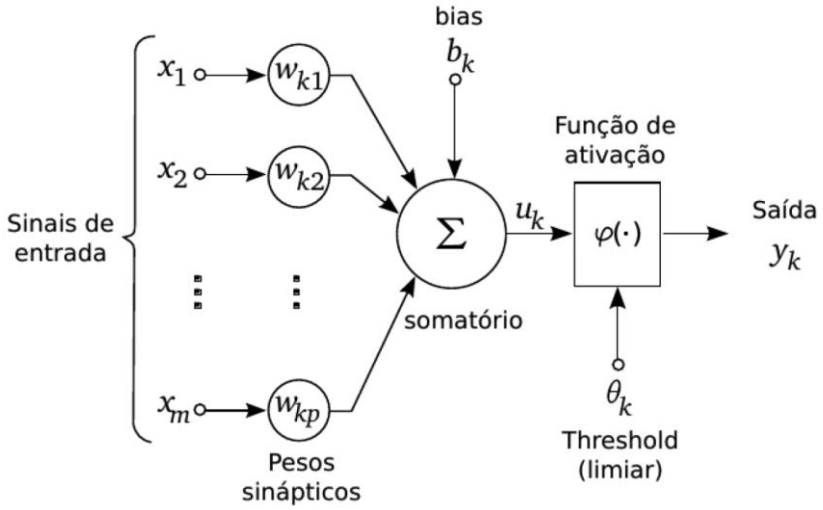
Figura 16 - Exemplo do modelo de um *perceptron* e suas entradas



Fonte: ROSENBLATT, 1958.

Observando a Figura 16, nota-se que as entradas x_1, x_2 e x_3 são acompanhadas de seus pesos sinápticos w_1, w_2 e w_3, respectivamente, números reais cuja operação resulta em uma saída y também real. Cabe ressaltar que comumente há uma entrada constante especial associada à um peso próprio denominada bias. A Figura 17 já ilustra um passo à frente do funcionamento do neurônio.

Figura 17 - Exemplo da progressão de informações de uma unidade perceptiva



Fonte: SILVA, SCHIMIDT, 2016.

Como mencionado anteriormente, o neurônio realiza operações de acordo com os valores de entrada e seus respectivos pesos e, a partir do resultado, toma-se uma decisão. Para ilustrar esse processo, considera-se uma soma ponderada destes termos de entrada como descreve a Equação 1:

$$\sum_{i=0}^{n} x_i w_i = 1w_0 + x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \qquad (1)$$

Após o cálculo da soma ponderada, este valor passará por uma função de ativação (do inglês, *activation function)*. Esta função é um importante parâmetro inerente à unidade de processamento, pois além de conectar diferentes neurônios, pode ser escolhida de modo à propiciar melhores resultados para uma determinada aplicação, fator que será melhor detalhado no item 3.3.4. Para exemplificar seu comportamento, a unidade ilustrada na Figura 17 utilizou uma função degrau como função de ativação e, com base em um valor limite (*threshold)* para uma classificação binária, retornará 0 ou 1 dependendo da magnitude da soma ponderada (SHARMA, SHARMA, ATHAIYA, 2020). A Equação 2 ilustra o comportamento da função degrau para um *threshold* igual a 0,7.

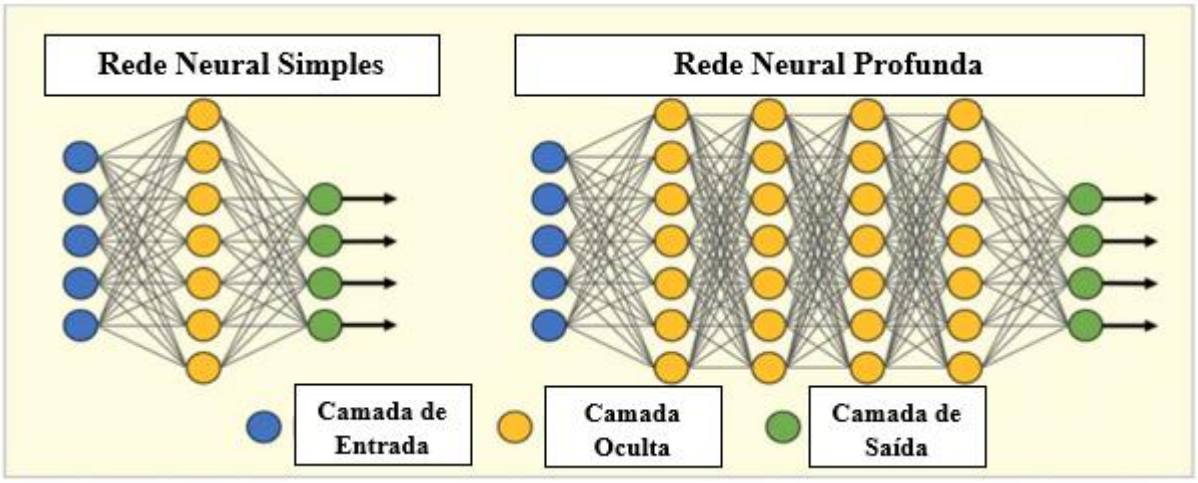
$$f(x) = \begin{cases} 1\,, & x \geq 0{,}7 \\ 0\,, & x < 0{,}7 \end{cases} \tag{2}$$

Ou seja, tomando de exemplo o caso em questão, para um valor resultante da soma ponderada igual a 0,85, a função degrau resultará no valor unitário. Em uma situação de aplicação no modelo, o valor unitário confirmará a existência do parâmetro desejado na análise e, portanto, a resposta do neurônio será positiva para tal evento. Também no item 3.3.4 serão comentadas outras funções de ativação e como suas contribuições afetam no resultado das operações.

3.3.1 Arquitetura de Redes Neurais Profundas

Denota-se por rede neural um agrupamento de neurônios, dispostos nos mais diferentes arranjos, a fim de desempenharem juntos uma tarefa que, comparada à um único neurônio, trará um aumento significativo da eficiente e otimização do modelo (HAYKIN, 2007; GORGENS, 2009). A Figura 18 ilustra dois tipos de redes neurais para posterior tratativas de suas diferenças.

Figura 18 - Diferença entre redes neurais simples e profundas



Fonte: COELHO, 2019.

Em redes neurais, os neurônios agrupam-se em diferentes camadas para desempenharem diferentes funções na estrutura do modelo. Denominam-se camadas ocultas todas as camadas que situam-se entre as camadas de entrada e as camadas de saída. Esta nomenclatura vem justamente em razão destes neurônios não serem "visíveis" aos observadores, representando processos internos, porém fundamentais, do modelo. Destas camadas ocultas também surge a nomenclatura "profunda", atribuída quando um modelo possui mais do que uma única camada oculta em sua arquitetura (BERNARDOS, VOSNIAKOS, 2007; PASZKE *et al.*, 2016).

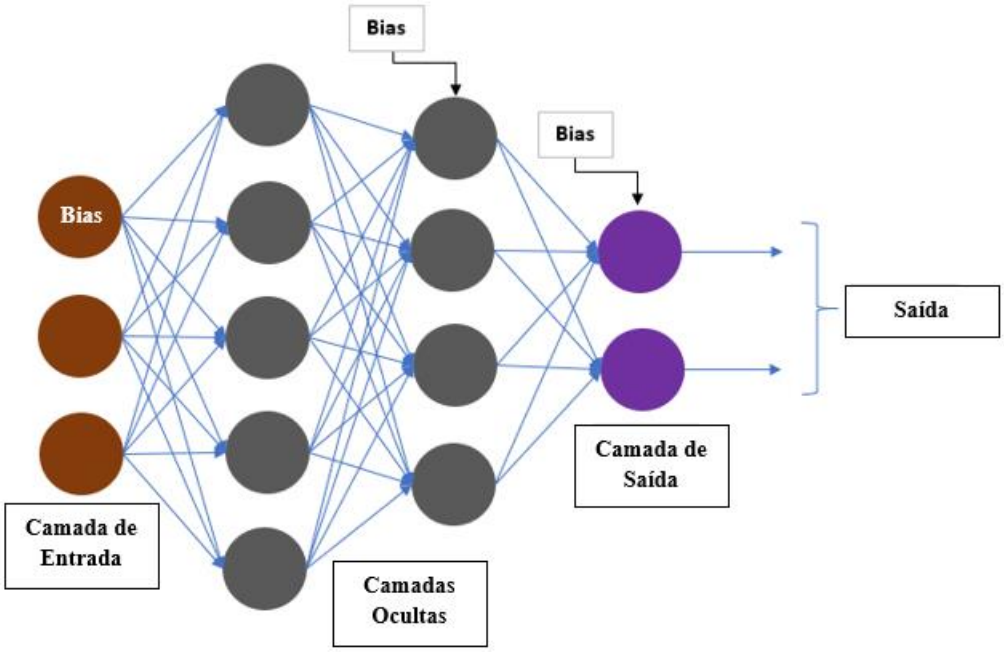
Cabe ressaltar que a camada de entrada não é constituída de neurônios, mas de dados que alimentarão todo o modelo. Também é válido mencionar que o número de neurônios de saída é exatamente o mesmo que o número de possibilidades para uma classificação. Como exemplo de uma classificação binária, haverá dois neurônios atribuídos ao cenário positivo ou negativo da detecção do parâmetro desejado (AZIZ, WONGM, 1992).

Em redes neurais também existem algumas nomenclaturas que são atribuídas de acordo com as características de sua construção. Camadas completamente conectadas são redes neurais que possuem uma ligação completa entre todas as suas camadas, ou seja, cada neurônio está diretamente conectado com os neurônios da camada anterior. *Feed-foward* é uma nomenclatura que refere-se ao sentido de propagação das informações, denotando que este sentido sempre parte da entrada para a saída, não havendo o fenômeno conhecido como *back propagation*, comumente encontrado em redes neurais recorrentes. Por fim, é comum deparar-nos com redes *non-skipped*, denotando que não há "saltos" de informações entre as camadas, ou seja, a informação é passada de camada em camada de modo progressivo e contínuo. Também é valido mencionar que comumente as redes neurais são conhecidas por *perceptrons* de múltiplas camadas (REED, MARKSII, 1999; SCHWING, URTASUN, 2015; HUANG *et al.*, 2017).

3.3.2 Número de camadas, pesos e neurônios

De modo geral, pode-se atribuir a complexidade do modelo de modo diretamente proporcional ao número de parâmetros (pesos) e, para ilustrar a relação entre o número de camadas e o número de neurônios, elaborou-se a Figura 19.

Figura 19 - Exemplo de uma rede neural com seus pesos e neurônios



Fonte: Elaborado pelo autor.

No caso do exemplo da Figura 19, nota-se que existem um total de 11 neurônios e 49 pesos sinápticos considerando o caso das camadas completamente conectadas com alimentação progressiva. Observa-se também que existem uma entrada bidimensional acompanhada de um termo de bias. Para realizar o cálculo do número de pesos de uma rede neural, deve-se multiplicar os números de neurônios entre uma camada e sua anterior, acrescido do valor da bias para a terceira camada em diante. Deste modo, a Equação 3 ilustra o cálculo:

$$Número\ de\ Pesos = (3x5) + (6x4) + (5x2) = 49 \quad (3)$$

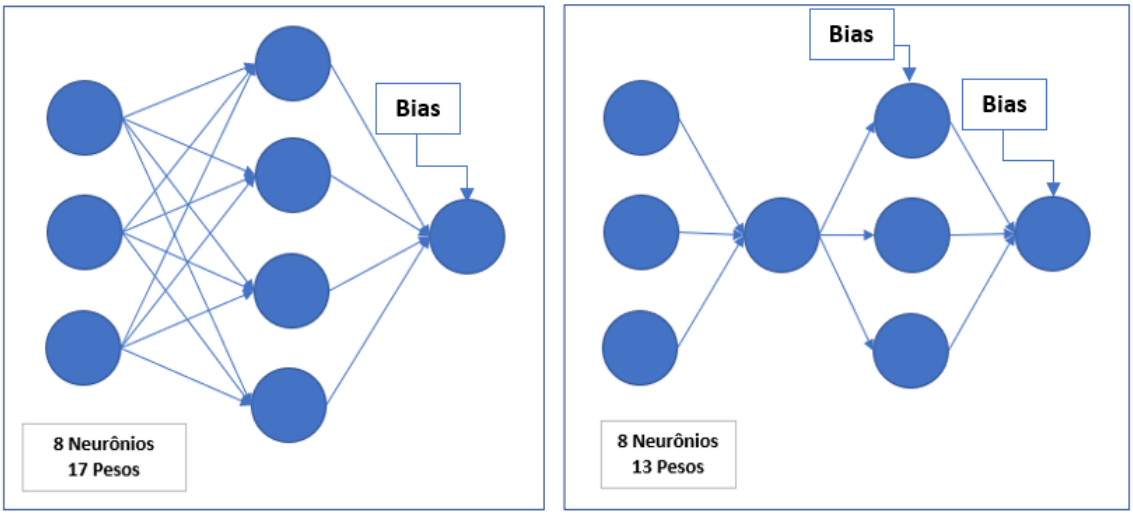
Em geral, quanto maior o número de pesos, maior será a flexibilidade da rede neural em lidar com as atribuições propostas pelo modelo, porém, maior será a magnitude da sobreposição aos dados (LAWRENCE, GILES, 2000).

Um importante aspecto dos arranjos dos neurônios está em relação às diferentes possibilidades acompanhadas de diferentes número de pesos mantendo o número de neurônios constante. Ou seja, os mesmos 11 neurônios podem resultar em um número diferente de pesos somente rearranjando-os em diferentes arranjos.

Deste modo, é possível rearranjar neurônios para tornar a rede cada vez mais profunda, reduzindo o número de parâmetros sem perder o poder de representação do modelo. Este pensamento implica que com um número muito reduzido de neurônios em grandes profundidades, podemos dar origem ao mesmo poder de representação que muitos neurônios na mesma camada. Em geral, a profundidade da rede reduz o número de neurônios e, consequentemente, o número de pesos, aumentando a flexibilidade do modelo, tornando possível modelar funções que, em uma única camada, iria requerer um volumoso número de neurônios para um número reduzido devidamente arranjado nas camadas profundas. Vale ressaltar que tanto número de neurônios quanto o número de camadas são hiperparâmetros (SCHÖLKOPF, PLATT, HOFMANN, 2007; HAN *et al.*, 2015).

A fim de ilustrar como o rearranjo dos neurônios impacta no número de parâmetros e pesos, elaborou-se a Figura 20.

Figura 20 – Impacto na diferença de pesos pelo rearranjo de camada dos neurônios



Fonte: Elaborado pelo autor.

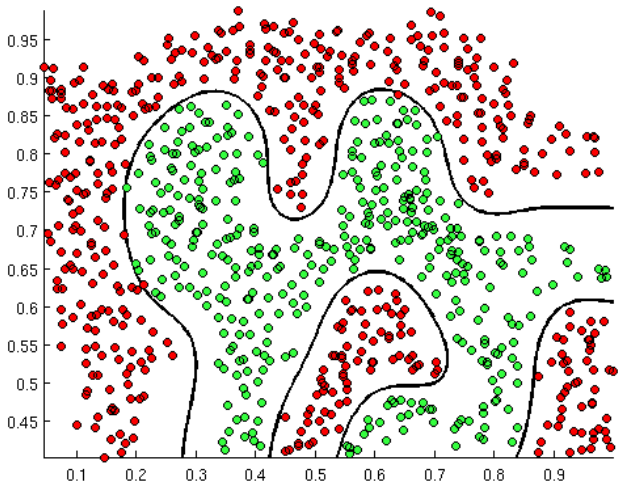
3.3.3 Limites de Decisão

Cada neurônio representa um hiperplano em um espaço tridimensional ou uma linha de dimensões finitas em um plano que, atuando em conjunto, geram o que é conhecido como o limite ou fronteira de decisões. Estes limites são representados por funções contínuas e representam o campo usado para a tomada de decisão em classificações, por exemplo, sendo gerado a partir das interseções sucessivas das linhas geradas pelos neurônios (CHULHEE, LANDGREBE, 1997; HE, LE, SONG, 2018).

A importância do termo referente à bias dá-se justamente por permitir um *offset* destas linhas em relação à origem do plano, cujo deslocamento permite que a inclinação das linhas

seja ajustada de forma que permita representá-la com maior fidelidade, ocasionando uma compensação benéfica para a classificação. Com a ausência deste termo, a inclinação das linhas serão sempre forçadas a passar pela origem do plano, prejudicando a elaboração dos limites. Caso a origem seja um ponto de interesse para início das linhas, o termo será automaticamente zero e permitirá tal ajuste (DIETTERICH, KONG, 1995; TUMER, GHOSH, 1996). Para exemplificação, a Figura 21 ilustra um destes limites de decisões para uma classificação binária.

Figura 21 - Exemplo de um limite de decisões para uma classificação binária



Fonte: NG, A., 2012.

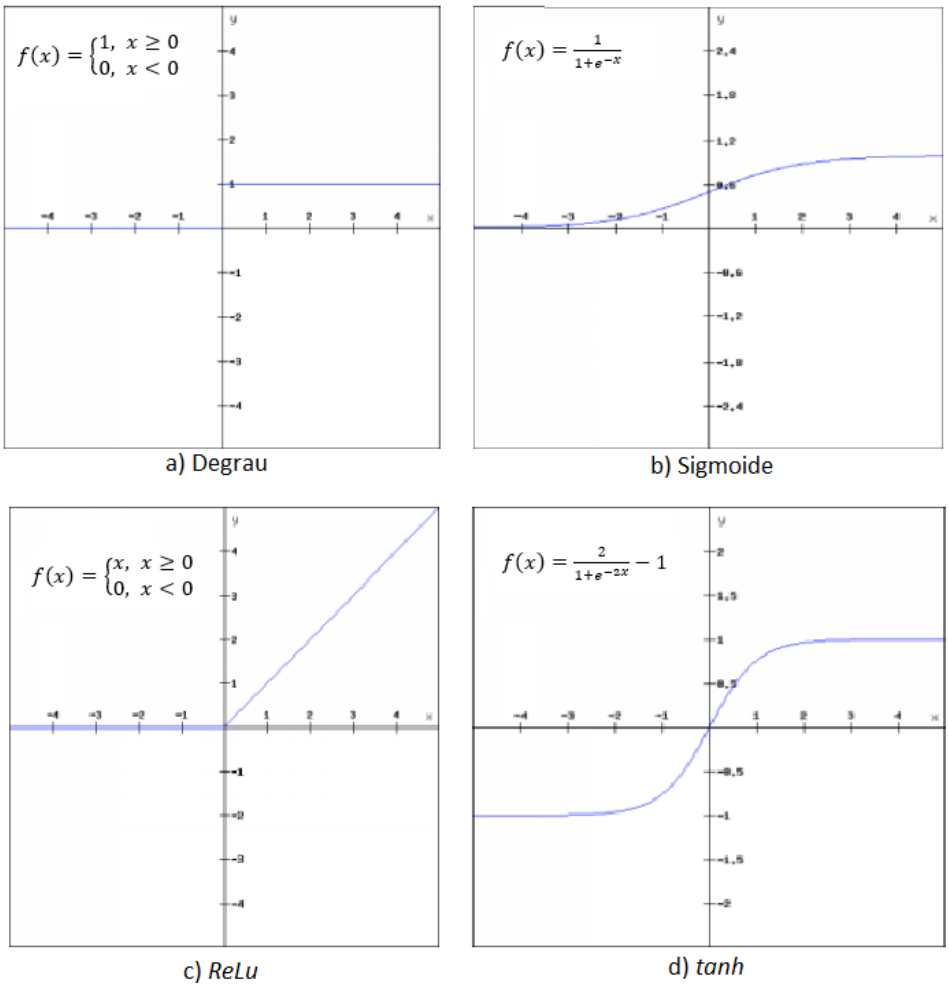
3.3.4 Funções de ativação

A aplicação das funções de ativação retornará valores que serão comparados à um determinado *threshold* que, dependendo da faixa em que se encontram, resultarão em uma saída positiva (1) ou negativa (0) para a constatação presente no elemento analisado. Caso estas funções de ativação não existissem em uma rede neural, todos os neurônios se comportariam como um único neurônio ou então como um modelo de regressão linear. Deste modo, estas funções permitem que cada neurônio contribua para a formação dos limites de decisões (LESHNO *et al.*, 1993; HUANG *et al.*, 2006).

Existem vários tipos de funções de ativação que podem ser utilizadas para determinada solicitação, podendo ser lineares ou não-lineares (geralmente utilizadas), sendo que suas recomendações de aplicação variam de acordo com as situações. Algumas funções mais utilizadas nestes modelos são as funções linares, degrau, rampa, sigmoide, *tanh e ReLU*, por exemplo (SCHMIDT-HIEBER, 2020; SHARMA, SHARMA, ATHAIYA, 2020).

A fim de ilustrar o comportamento e as particularidades de algumas destas funções de ativação, elaborou-se a Figura 22.

Figura 22 - Visualização gráfica de algumas funções de ativação comuns



Fonte: Adaptado de (SHARMA, SHARMA, ATHAIYA, 2020)

3.3.5 Parâmetros de treinamento

O treinamento busca encontrar os melhores parâmetros (pesos) para reduzir o máximo possível as perdas decorrentes destas aquisições. A perda, muitas vezes mencionada como custo, pode ser definida de diversos modos, sendo comumente adotada como o quadrado da diferença entre o valor obtido e um valor de referência. Portanto, o treinamento colabora para adequar os melhores valores dos pesos para minimizar a perda global para o conjunto de dados em análise (WEBER, 1994; ZHAO *et al.*, 2019).

A adequação desses pesos depende da natureza do modelo e dos custos envolvidos nessa escolha e, normalmente, despende-se diversas iterações que mudam ligeiramente os parâmetros e avalia seus impactos nas perdas até encontrar uma combinação que satisfaça as solicitações (WEBER, 1994; CHENG, 2016).

O modo como o treinamento é feito baseia-se em um palpite inicial, que inicialmente pode estar muito longe do esperado, mas servirá de referência para os próximos passos. Existem

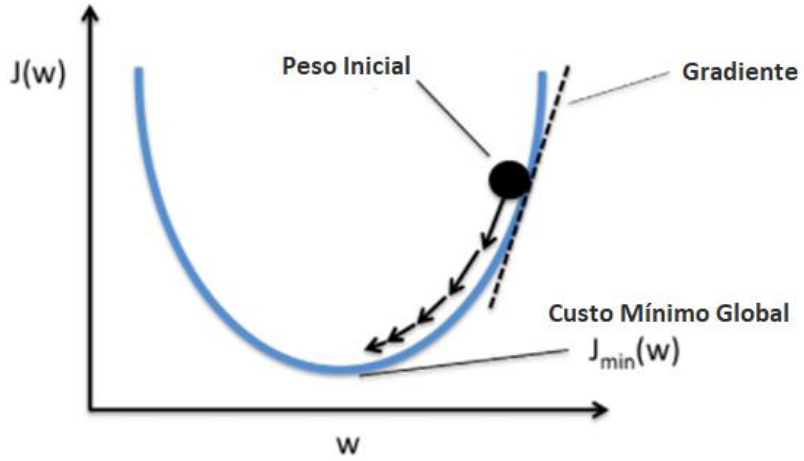
algumas formas de assegurar a redução das perdas no desenvolvimento do treinamento, assunto que será abordado em sequência (KUMAR, 2017).

O método da descida do gradiente refere-se à aplicação de um algoritmo que contribui para garantir a redução das perdas em cada iteração. Em geral, tende a ser uma parcela que despende consideráveis esforços computacionais em decorrência do cálculo do gradiente, porém é amplamente utilizado por assegurar uma progressão em direção à redução de perdas. Esta metodologia preconiza que o ponto de interesse deve caminhar sempre em direção oposta à direção do gradiente e, dessa forma, será assegurado que a perda será minimizada (ANDRYCHOWICZ *et al.*, 2016;RUDER, 2016).

O processo consiste com a inicialização do aprendizado com alguns parâmetros iniciais e, então, calcula-se a perda associada. Após o cálculo da perda, computa-se o valor do gradiente desta iteração, fornecendo a direção de caminho para os próximos passos, que será oposta à direção ascendente do gradiente. Assim, após um caminho em uma magnitude pré-estabelecida, atualiza-se os parâmetros e o processo é repetido.

Em suma, se a função de custo pode ser dita convexa, a descida do gradiente retornará parâmetros muito otimizados, que desenvolveram grande fidelidade do modelo aos dados. Porém, caso a função de custo não seja convexa, a descida do gradiente poderá retornar um mínimo local dependendo dos locais onde os parâmetros foram iniciados (SCHORFHEIDE, 2000; BENGIO *et al.*, 2006). A fim de ilustrar este método, a Figura 23 ilustra uma função de custo perfeitamente convexa $J(w)$ com relação à um único parâmetro w.

Figura 23 - Exemplo da aplicação da descida de gradiente e inicialização de pesos



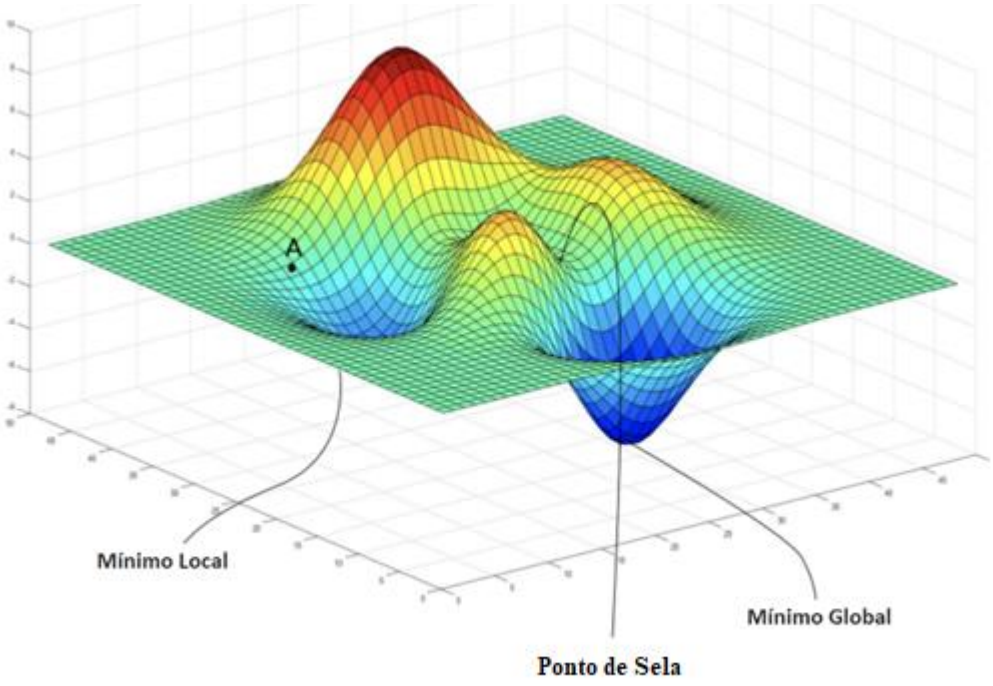
Fonte: Adaptado (SURYANSH, 2018)

Analisando a Figura 23, nota-se que quando a função custo é perfeitamente convexa, não importa onde os pesos forem iniciados, sempre acabarão atingindo o mínimo global $J_{min}(w)$ quando o sentido oposto do gradiente é percorrido. A velocidade que este mínimo será atingido

dependerá da amplitude dos passos, parâmetro conhecido na literatura como o tamanho dos passos, ou então, taxa de aprendizagem. Este parâmetro, aliado aos demais supracitados, possuem uma contribuição significativa na velocidade de convergência dos modelos (ZEILER, 2012; DANIEL, TAYLOR, NOWOZIN, 2016).

Porém, muitas vezes as funções de custo não são convexas e, quando há outros parâmetros podem haver vários pontos de mínimo, pontos esses conhecidos como mínimos locais. Neste caso, há uma preocupação acentuada com o local de início dos pesos pois, dependendo dessa inicialização, pode-se aprisionar o modelo à um mínimo local, ou então à um ponto de sela e inviabilizar a análise (AVRIEL, ZANG, 1980; JUNG, LEE, LEE, 2008). A Figura 24 ilustra uma dessas situações.

Figura 24 - Exemplo de uma superfície que pode aprisionar o algoritmo de descida do gradiente



Fonte: Adaptado (KADAM, 2020)

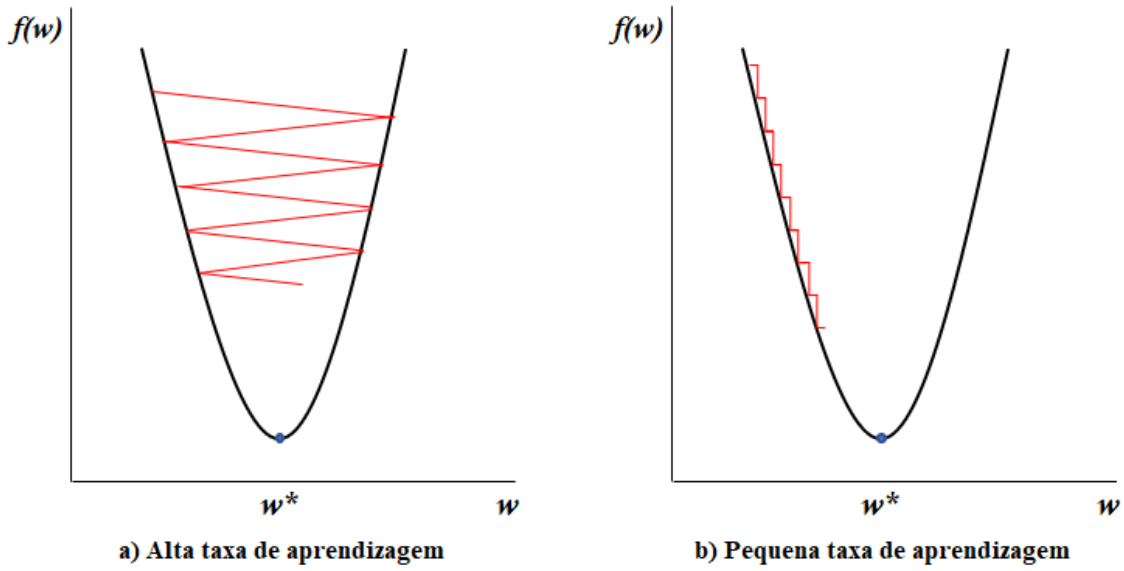
Tomando a Figura 24 como exemplo, caso o modelo seja iniciado no ponto A, a descida de gradiente o levará para o mínimo local e este estará aprisionado. Também é possível que o modelo encontre um ponto de sela, causando também uma estagnação do modelo.

Vale ressaltar que apesar de desejado, o mínimo global não deve ser tratado como critério necessário e exclusivo para o modelo, visto que, mesmo em um mínimo local, o modelo poderá ter grande representatividade e desempenhar resultados satisfatórios para viabilizar a análise desejada. Um método comum de contornar o problema dos mínimos locais e dos pontos de sela consiste na distribuição normal e aleatória do valor dos pesos ao longo da superfície,

inicializando o modelo em diferentes pontos e, consequentemente, permitindo ao modelo experimentar diferentes mínimos (IOFFE, SZEGEDY, 2015; SANTURKAR *et al.*, 2018).

Como mencionado anteriormente, um dos maiores desafios práticos do desenvolvimento de redes neurais está na escolha de uma taxa de aprendizado adequada para o modelo. Este parâmetro descreve a magnitude dos passos que serão dados em direção ao mínimo (local ou global), cuja magnitude pode contribuir ou desacelerar o processo de aprendizagem. Há muitas tratativas literárias que buscam uma taxa de aprendizagem ideal, porém, abordagens empíricas desempenham-se de modo mais satisfatório para a maioria das aplicações sendo que, comumente, adota-se o valor igual a 0,01 como ponto de partida (ZEILER, 2012; DANIEL, TAYLOR, NOWOZIN, 2016). A Figura 25 ilustra a diferença qualitativa que implica as diferentes taxas de aprendizagem na convergência do modelo.

Figura 25 – Exemplo da diferença qualitativa da variação da taxa de aprendizagem



Fonte: Adaptado (GATTAROLA, 2017)

Outro conceito fundamental trata-se do modo como os dados são apresentados para treinamento. Define-se por *epoch* o ato de apresentar completamente todo o conjunto de dados disponível para treinamento do modelo. A atualização dos parâmetros pode ser realizada a cada *epoch*, denotando que o modelo está atuando no *batch mode*, ou então, no modo à batelada, como também para cada exemplo analisado, denotando o modo estocástico (YOU, GITMAN, GINSBURG, 2017; BROWNLEE, 2018).

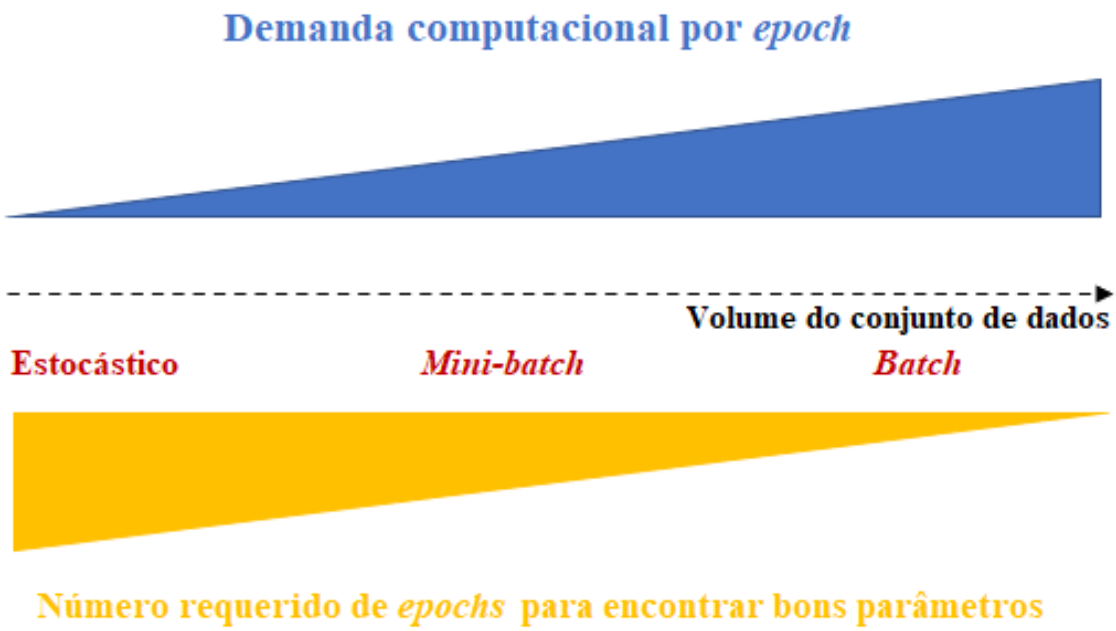
Geralmente, utiliza-se o *batch mode* quando o volume do conjunto de dados não é muito grande, possibilitando que o modelo seja altamente controlável. Já para conjunto de dados volumosos, recomenda-se que a atualização dos parâmetros seja feita a cada análise, ou seja, trabalhe no modo estocástico. Porém, o modo estocástico implica que o caminho até o mínimo

será mais turbulento e não tão bem definido como no *batch mode* (YOU, GITMAN, GINSBURG, 2017; BROWNLEE, 2018).

Existe uma abordagem intermediária conhecida como *mini-batch*. Esta abordagem consiste na divisão do conjunto de dados em diversos segmentos e, para cada segmento, é utilizada uma abordagem de batelada para atualização dos parâmetros. Já em relação à interação dos segmentos fracionados, utiliza-se uma abordagem estocástica. Desse modo, o *mini-batch* possibilita um equilíbrio entre a natureza estocástica e de lote do algoritmo, mostrando-se muito eficiente em diversos cenários por evitar deslocamentos bruscos em razão de suas propriedades de suavização (HINTONS, SRIVASTAVA, SWERSKY, 2012; LI *et al.*, 2014).

Cabe ressaltar também que a abordagem estocástica requer um grande número de *epochs* para convergência, porém, em geral, baixos recursos computacionais. Uma abordagem de lotes já requer menores números de *epochs*, porém geralmente é despendido grandes recursos computacionais (GOYAL *et al.*, 2017; LIN *et al.*, 2018). Uma relação entre estas características pode ser melhor visualizada na Figura 26.

Figura 26 – Relação entre o volume de dados e a demanda computacional para obter bons parâmetros



Fonte: Adaptado (KHOR, 2016)

Em geral deseja-se que todos os lotes possuam distribuições idênticas de informações entre si, porém muitas das vezes é possível que haja heterogeneidades, fenômeno conhecido como *covariate shift*. Para a solução deste caso, utiliza-se uma técnica para normalização dos dados, cuja abordagem colabora para que a média das variáveis dos lotes tendam a zero e o desvio padrão a um em cada lote. Ou seja, a ideia é tornar os lotes mais homogêneos, cujo

impacto prático reflete na rápida convergência do modelo pela descida do gradiente (SUGIYAMA, KRAULEDAT, MÜLLER, 2007; IOFFE, SZEGEDY, 2015).

Para isso, adiciona-se mais dois parâmetros à cada camada, de modo independente, conhecidos como parâmetros de normalização. Estes parâmetros devem ser desenvolvidos juntamente com os pesos e não impostos inicialmente antes do treinamento, ou seja, devem ser "aprendidos" para melhor se ajustarem ao conjunto de dados em questão. Para grandes volumes de dados, a normalização dos lotes é uma atitude padrão que trará grandes benefícios ao modelo (SUGIYAMA, KRAULEDAT, MÜLLER, 2007; IOFFE, SZEGEDY, 2015).

Em aplicações práticas verifica-se que manter a taxa de aprendizagem constante para todos os sucessivos *epochs* não contribui para a rápida convergência do modelo. Por um processo intuitivo, quando o modelo encontra-se longe do mínimo local, pode-se aumentar o tamanho dos passos para buscar uma direção preferencial, porém, à medida que se aproxima do mínimo local, utilizar passos menores colabora com a suavização e convergência do modelo (BOWLING, VELOSO, 2002; ZEILER, 2012).

Para adaptar a taxa de aprendizagem existem diversos modos, sendo que um modo bem comum é dividir esta taxa pelo número de *epochs* realizados até o momento, caracterizando uma forma heurística de adaptação das taxas pela sua diminuição progressiva conforme o número de *epochs* cresce (BOWLING, VELOSO, 2002; SMITH *et al.*, 2017).

Outra tratativa no processo de aprendizagem preconiza que os parâmetros devem possuir taxas de aprendizagem independentes entre si quando estes parâmetros não possuem relações com os demais. Já no caso que os parâmetros estão intimamente ligados entre si, torna-se vantajoso estabelecer uma relação entre os pesos em razão da dependência existente, visto que tratá-los de forma independente prejudicará a convergência global (SALIMANS *et al.*, 2016; WEN *et al.*, 2018).

Tomando um exemplo de um parâmetro que sempre move em direção à um mínimo local e outro parâmetro que apresenta uma grande variação simultânea, cuja média das flutuações das direções é próxima de zero, pode-se negligenciar tal parâmetro e adotar o caminho percorrido pelo parâmetro cuja tendência mostra-se mais promissora (JACOBS, 1988; LOWD, DOMINGOS, 2007).

Como redes neurais profundas possuem, em geral, um grande número de parâmetros, pode-se flexibilizar o aprendizado utilizando um número efetivo de parâmetros. Esta flexibilização não está relacionada com a capacidade do modelo de desempenhar análises concisas nos dados de treinamento, mas nos dados externos cujo modelo será implantado para análise e, muitas das vezes, está análise é prejudicada justamente pelo problema de sobre-ajuste

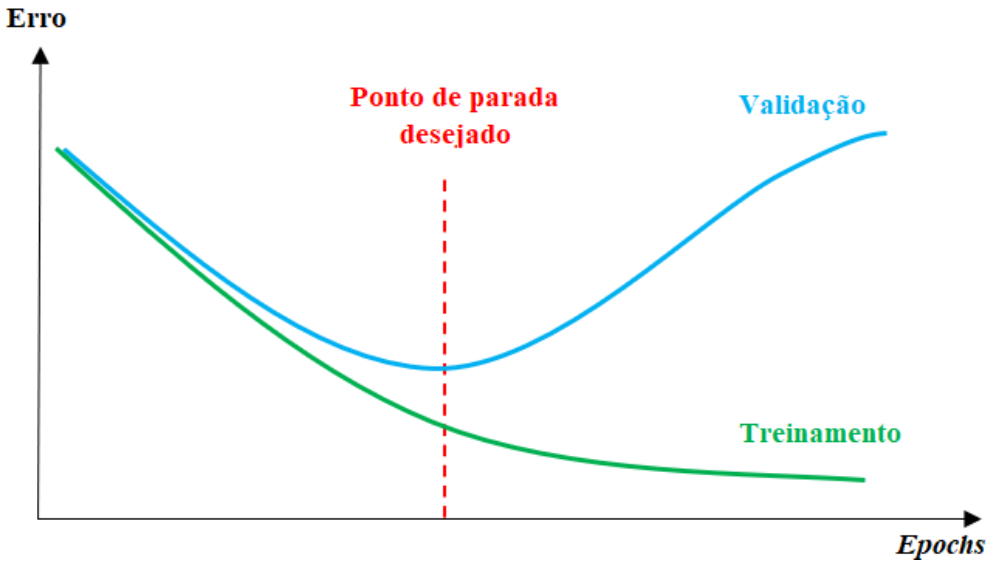
nestes dados de treinamento (SPINOSA, CARVALHO, GAMA, 2007; FEURER, HUTTER, 2019).

Ou seja, um modo de reduzir o sobreajuste é induzir a rede neural para trabalhar com um número reduzido de parâmetros, permitindo à rede experimentar diferentes abordagens encontradas em cada um dos *mini-batches*. Para isso, pode-se aplicar um método de controle e incremento de flexibilização que baseia-se na escolha aleatória de alguns nós para ignorar tanto suas entradas como suas saídas. Com isso, para uma mesma rede neural, pode-se obter uma generalização benéfica ao modelo principalmente por reduzir o sobreajuste (COGSWELL *et al.*, 2015; HA *et al.*, 2019; GARBIN, ZHU, MARQUES, 2020).

Outro modo para contornar o *sobreajuste* consiste em estabelecer um momento adequado para parar o treinamento antes que o modelo sobreponha os dados de treinamento e tenha sua performance prejudicada nos dados de validação. A cada *epoch* concluída, computa-se o erro do treinamento e da validação de dados e, a partir do momento que o erro de validação começa a subir, cria-se um ponto candidato para parada do treinamento (ZHANG, YU, 2005; YAO, ROSASCO, CAPONNETTO, 2007, PRECHELT, 2012).

Porém, em alguns casos, pode ser que o erro de validação aumente por um período mas depois volte a cair, cuja parada precoce poderia prejudicar o aproveitamento do treinamento. Este parâmetro, por sua vez, é conhecido como parâmetro de paciência. A partir do momento que o erro mostrou clara tendência de aumento, para-se o processo de aprendizagem ali e adota-se os parâmetros neste instante, manobra conhecida como parada antecipada (PRECHELT, 1998; ZHANG, YU, 2005; YAO, ROSASCO, CAPONNETTO, 2007). A Figura 27 ilustra o comportamento do erro em relação ao número de *epochs*, bem como o critério de parada adotado.

Figura 27 - Critério de parada do treinamento em função do custo e do número de *epochs*



Fonte: Adaptado (ARVETTI, GINI, FOLGHERAITER, 2007)

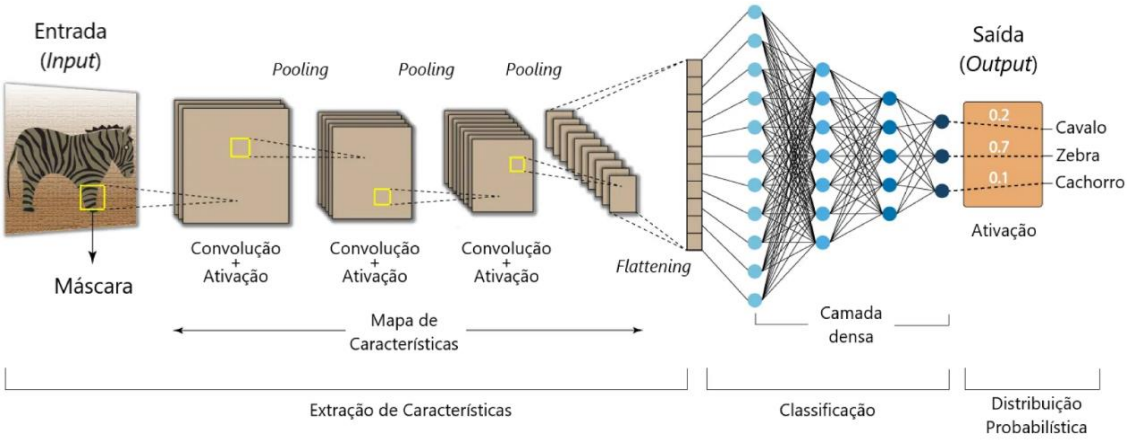
Foram mencionados diversos parâmetros inerentes às redes neurais e o modo qualitativo como cada um contribui para o treinamento, porém, alguns parâmetros em especial são mencionados como hiperparâmetros em razão de uma característica especial. Hiperparâmetros são uma classe de parâmetros que não são adquiridos e não podem ser desenvolvidos no treinamento, ou seja, precisam ser pré-definidos antes do início do treinamento pelo desenvolvedor do modelo (MACKAY, 1996; BENGIO, 2000).

Alguns exemplos de hiperparâmetros vistos anteriormente são: o número de camadas e de neurônios, a função de ativação, a taxa de aprendizagem e sua atenuação, o tamanho dos lotes, a inicialização dos pesos, entre outros. Cabe ressaltar que a alteração destes parâmetros pode mudar todas as características do processo de aprendizagem, sendo que, a escolha mais adequada da combinação dos hiperparâmetros não é trivial e requer vivência e experiência no desenvolvimento de redes neurais (MACKAY, 2000; SMITH, 2018).

3.3.6 Desenvolvimento das etapas

Como mencionado em seções anteriores, dada uma imagem para treinamento do modelo, existe uma série de eventos em cascata que realizam, individualmente, alguma operação almejando a extração de características e a classificação, detecção e localização de algum objeto de desejo. A fim de ilustrar como este processo ocorre em uma escala macroscópica, pode-se observar a Figura 28.

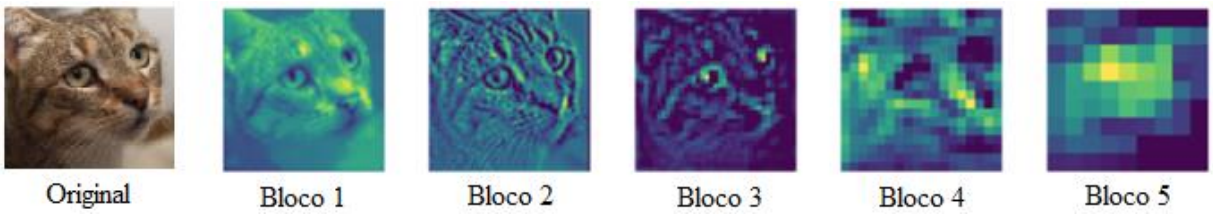
Figura 28 - Estrutura genérica de uma rede neural convolucional na classificação de imagens



Fonte: Adaptado (DEVELOPERS BREACH, s.d)

Inicialmente a imagem é submetida à uma operação de convolução utilizando uma máscara e, desta operação, surge o que é conhecido como mapa de características. As dimensões e características destes mapas são diretamente dependentes da máscara utilizada no processo de convolução, bem como de parâmetros como *padding* e *stride* utilizados (DUMOLIN, VISIN, 2016; ZHANG, PENG, 2019). Alguns mapas de características podem ser observados na Figura 29 para ilustrar, visualmente, como o modelo percebe estes dados.

Figura 29 - Exemplo de mapas de características



Fonte: Adaptado (DERTAT, 2017)

Observando a Figura 29 nota-se que inicialmente o mapa de características retém a maior parte de informações presentes na imagem, mas à medida que a imagem se aprofunda na rede neural, os mapas de características geram representações abstratas que, visualmente, tornam-se pouco interpretáveis.

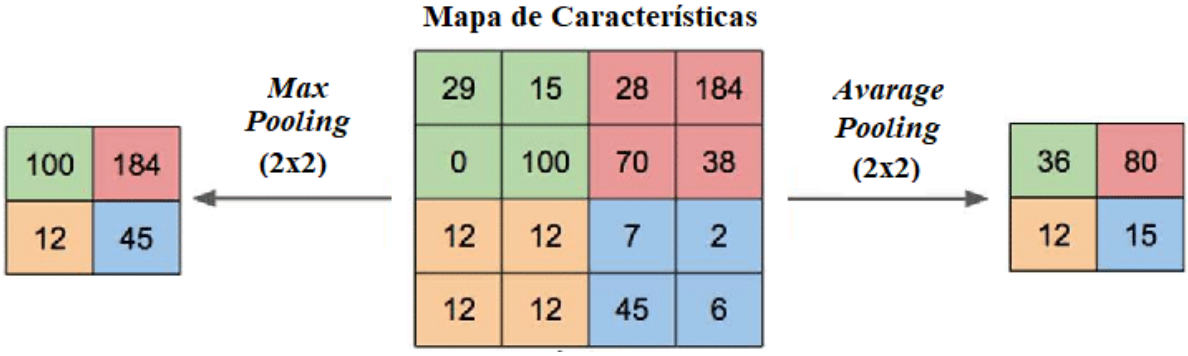
Então, após o processo de convolução, pode-se normalizar os lotes a fim de homogeneizar suas distribuições, realizar as operações de *dropout* e aplicar a função de ativação. Cabe ressaltar que a função de ativação *ReLU* é uma das mais utilizadas para estas finalidades(IOFFE, SZEGEDY, 2015; LI *et al.*, 2019). Em seguida, encontra-se a camada responsável pelo que é conhecido como amostragem.

A camada de amostragem transforma os mapas de características com a finalidade de realçar as características mais expressivas da imagem e, também, para reduzir suas dimensões.

Esta redução de dimensões acarreta diretamente a redução de parâmetros necessários e, consequente, reduz o sobreajuste e o esforço computacional (SRIVASTAVA *et al.*, 2014; YU *et al.*, 2014).

De modo geral, dois tipos de amostragem são frequentemente utilizados, o *max pooling* e o *avarege pooling*. No caso do *max pooling*, extrai-se o maior valor de uma poção do mapa de características, enquanto no *avarage pooling* realiza-se uma média de todos esses valores (ZEILER, FERGUS, 2013; WANG *et al.*, 2018). A fim de ilustrar estas duas abordagens, elaborou-se a Figura 30.

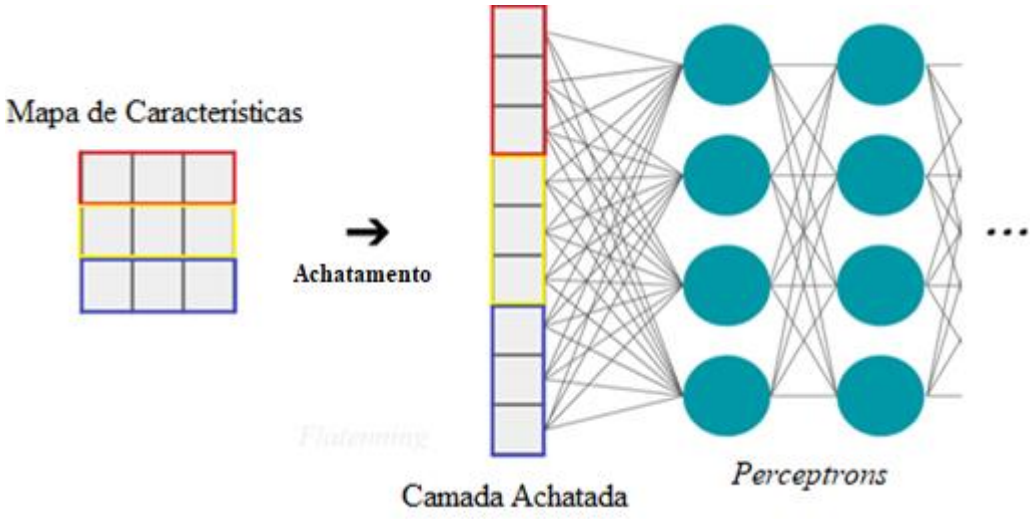
Figura 30 - Exemplo da aplicação de *max pooling* e *avarage pooling*



Fonte: Adaptado (YANI *et al.*, 2019)

Essa sequência de eventos é repetida várias vezes em razão do número de camadas existentes e, no final do processo, ocorre o que é conhecido como achatamento. Este processo consiste na conversão dos dados, até então dispostos em arranjos bidimensionais, para um arranjo unidimensional, ou seja, para um vetor de características (BRAHMA, WU, SHE, 2015; CHAKRABARTY, 2018). A camada achatada representa o fim da sequência de extração de características e estabelece o contato e a alimentação das camadas densas para classificação da imagem, por exemplo. A Figura 31 ilustra o processo de achatamento e a conexão com os *perceptrons* da camada densa.

Figura 31 - Exemplo do achatamento do mapa e conexão com a camada densa



Fonte: Adaptado (JEONG, 2019)

Cabe ressaltar que muitas das vezes essas camadas densas são totalmente conectadas, como ilustrado no exemplo da Figura 31. Nestas camadas regularmente também é aplicado o *dropout* para reduzir o sobreajuste e regularização da rede neural (GAL, GHAHRAMANI, 2016).

No final do processo, para o cenário de múltipla classificação, geralmente encontra-se a camada *Soft-Max* de ativação. Essa camada, por sua vez, é a responsável pela saída e classificação da imagem por meio de uma distribuição probabilística, o qual utiliza valores decimais entre zero e um referentes à probabilidade do objeto estar em uma classe específica, cuja soma de todos os valores é unitária (DUNNE, CAMPBELL, 1997; LIU *et al.*, 2016).

3.4 YOU ONLY LOOK ONCE (YOLO)

Y*OLO,* termo abreviado para *You Only Look Once*, foi um algoritmo desenvolvido por Joseph Redmon *et. al.* em maio do ano de 2016 que utiliza redes neurais convolucionais para detecção de múltiplos objetos, predição de classes e localização de suas posições. Desde seu lançamento, o *YOLO* tornou-se referência e um dos mais populares e favoritos algoritmos para detecção de objetos em tempo real (REDMON *et al.*, 2016; SHAFIEE *et al.*, 2017).

Esta primeira versão representou uma grande evolução no cenário e, em dezembro de 2017, Joseph introduziu uma versão otimizada que ficou conhecida como *YOLO9000*. No ano seguinte, em abril de 2018, Joseph introduziu a versão mais popular e estável do algoritmo denominada *YOLOv3* (REDMON, FARHADI, 2017; REDMON, FARHADI, 2018).

Porém, em decorrência de um afastamento na busca pela otimização contínua do algoritmo por parte do Joseph e sua equipe, Alexey Bochkovskiy introduziu a quarta versão do

algoritmo em, *YOLOv4*, em abril de 2020, com grandes mudanças que superaram a performance da versão antecessora (BOCHKOVSKIY, WANG, LIAO, 2020).

Pouco tempo depois, em junho de 2020, outro desenvolvedor chamado Glenn Jocher introduziu a quinta versão do algoritmo denominada *YOLOv5*. Muito ainda é discutido à respeito desta versão no que tange à performance e à praticidade comparada com a versão 4, porém, apesar da comparação em altos níveis entre as versões, ambas mostram-se excelentes para a detecção de objetos em tempo real (CHEN *et al.*, 2021; LIU *et al.*, 22021; THUAN, 2021).

3.4.1 Características do algoritmo YOLOv5

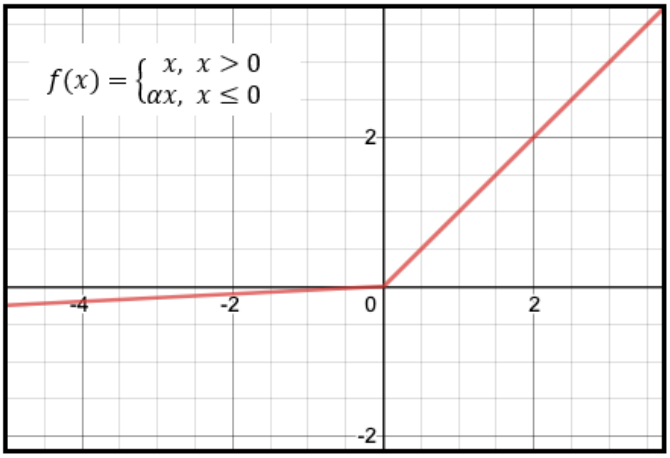
As camadas responsáveis pela extração de características são mencionadas como a espinha dorsal do modelo e, no caso da quinta versão, utiliza-se uma rede conhecida como *Cross Stage Parcial Networks* (CSPNet), cujo autor retrata grandes reduções de esforços computacionais para níveis equivalentes de acurácia (CHEN *et al.*, 2020; VISHWAKARMA, VENNELAKANTI, 2020).

Já o pescoço do modelo é geralmente utilizado para gerar pirâmides de características, contribuindo para que o modelo possua melhores desempenhos em relação à variação de escala nos dados de teste. No caso do *YOLOv5*, utiliza-se para isto uma rede conhecida como *Path Aggregation Network* (PANet) (LIU *et al.*, 2018; WANG *et al.*, 2021).

No final do modelo utiliza-se a cabeça para desempenhar a detecção final das características, aplicando *anchor boxes* nas características e gerando a probabilidade das classes, pontuações e as *bounding boxes*. Cabe ressaltar que esta porção do modelo utilizada no *YOLOv5* é a mesma utilizada desde o desenvolvimento do *YOLOv3* (HURTIK *et al.*, 2020; FANG, LIU, LI, 2021).

A função de ativação sigmoide foi escolhida pelo desenvolvedor na camada final de detecção, enquanto nas camadas intermediárias ocultas, a função escolhida é conhecida como *Leaky ReLU*. Esta função é uma variação baseada na função de ativação *ReLU* que admite uma pequena inclinação para valores negativos, expressa por um coeficiente α, visando aumentar a abrangência de ativação (XU *et al.*, 2015; DUBEY, 2019). Seu comportamento pode ser observado na Figura 32.

Figura 32 - Comportamento da função *Leaky ReLU* de ativação



Fonte: Adaptado (SINGH, c2021)

De modo padrão, o algoritmo utiliza a *Stochastic Gradient Descent* (SGD) como função de otimização, porém, pode-se utilizar outra função conhecida como Função de Estimação Adaptativa de Momento (*ADAM – Adaptative Moment Estimation*, da sigla em inglês), por meio de uma simples menção no código (BOTTOU, 2010; BOTTOU, 2012).

Já para a função de custo, pode-se também escolher utilizar uma função conhecida como *Binary Cross-Entropy with Logits Loss Function*, já disponível na biblioteca *PyTorch* utilizada pelo algoritmo, ou então utilizar uma função conhecida como *Focal Loss Function* também por menções no código (LIN *et al.*, 2017; ZHANG, SABUNCU, 2018).

3.4.2 Detecção de objetos

A classificação de uma imagem é uma tarefa desempenhada pelo algoritmo com a finalidade de, dado uma série de classes, definir qual delas melhor se encaixam aos padrões analisados. A localização, por sua vez, torna o algoritmo responsável por criar uma *bounding box* na posição que o objeto encontra-se na imagem. Já o problema envolvendo a detecção abrange múltiplos objetos, geralmente envolvendo diversas classes em uma mesma imagem, cujo algoritmo deve ser capaz de classificá-los e localizá-los de modo simultâneo e, muitas das vezes, em tempo real (HARZALLAK, JURIE, SCHMID, 2009; REDMON *et al.*, 2016; LONG *et al.*, 2017). A Figura 33 demonstra a diferença entre a detecção de múltiplos objetos.

Figura 33 - Diferença entre classificação, localização e detecção de objetos

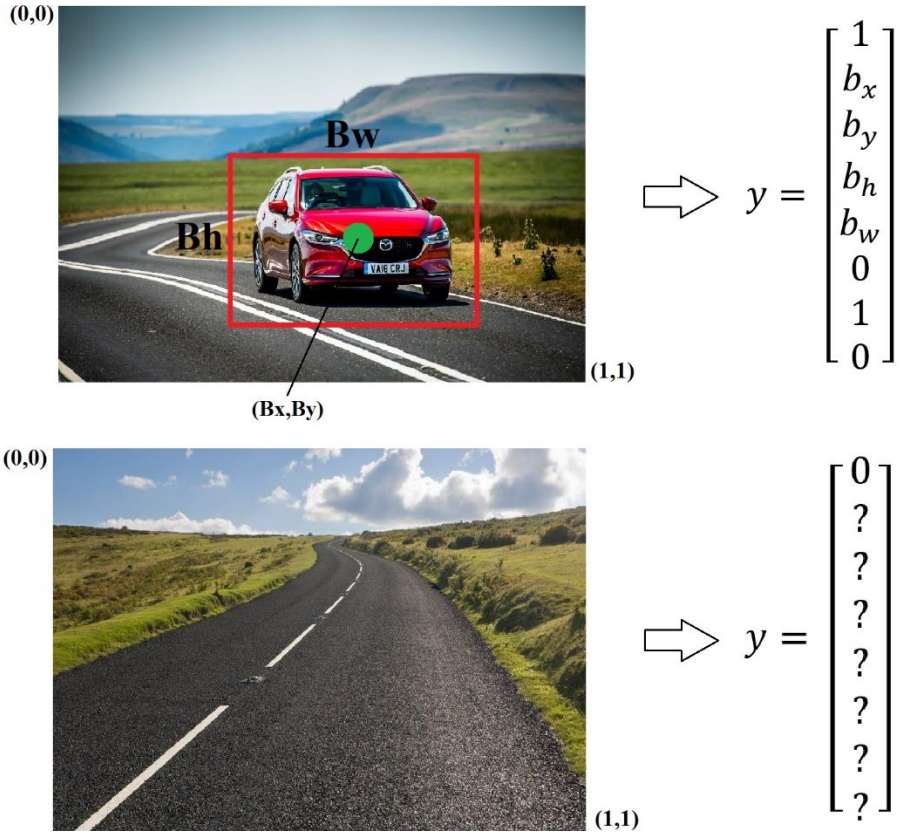
Fonte: Adaptado (DHAMI, 2021)

No problema envolvendo a classificação e localização de objetos, a saída da última camada de classificação terá, além da probabilidade de cada uma das classes, outros quatro termos referentes à parametrização das *bounding boxes* armazenados em um vetor conhecido como vetor alvo (JUDD, SMITH, WEISHEIMER, 2007; ĆOROVIĆ *et al.*, 2018). Supondo um cenário com três classes possíveis para classificação dos objetos, este vetor assumiria a forma ilustrada na Equação 4:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{4}$$

Os elementos c_1, c_2 e c_3 ilustrados na Equação 4 denotam as classes existentes para distinção e classificação pelo algoritmo enquanto b_x, b_y, b_h, e b_w representam a localização do centro, da largura e da altura da *bounding box*. Já o elemento P_c refere-se à existência, ou não, de alguma classe existir na imagem de análise (SHASTRY, 2018). A Figura 34 ilustra o comportamento deste vetor para os cenários onde existem e não existem objetos identificados.

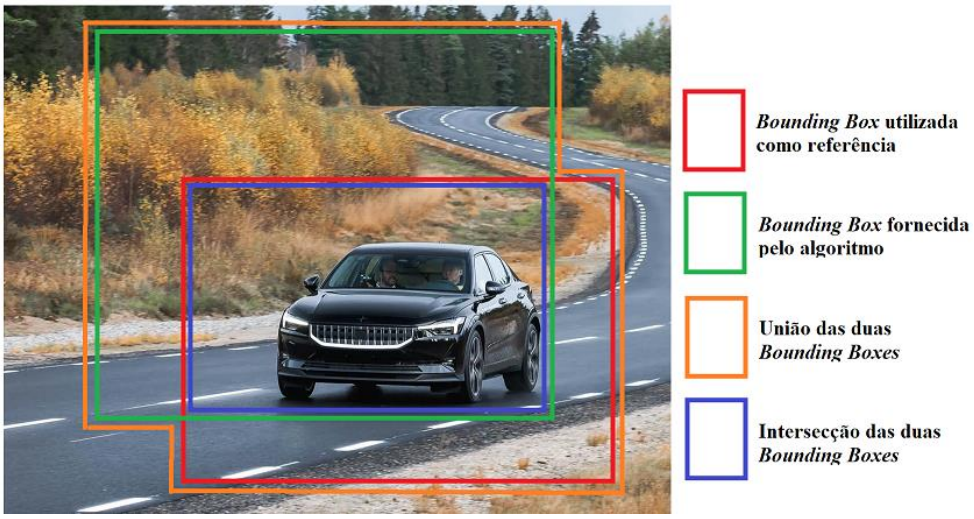
Figura 34 - Exemplo do comportamento do vetor alvo para dois cenários



Fonte: Adaptado (HARISSON, 2016; IS THE MAZDA…, 2018)

Uma importante função conhecida por *Intersection over Union* (IoU) é utilizada para avaliar a posição das *bounding boxes* em relação ao local desejado por meio de uma relação entre a área de união e de intersecção. A fim de exemplificar o modo como se relacionam as *bounding boxes*, têm-se a Figura 35.

Figura 35 - Exemplo da relação entre as *bounding boxes*



Fonte: Adaptado (VALENZUELA, 2019)

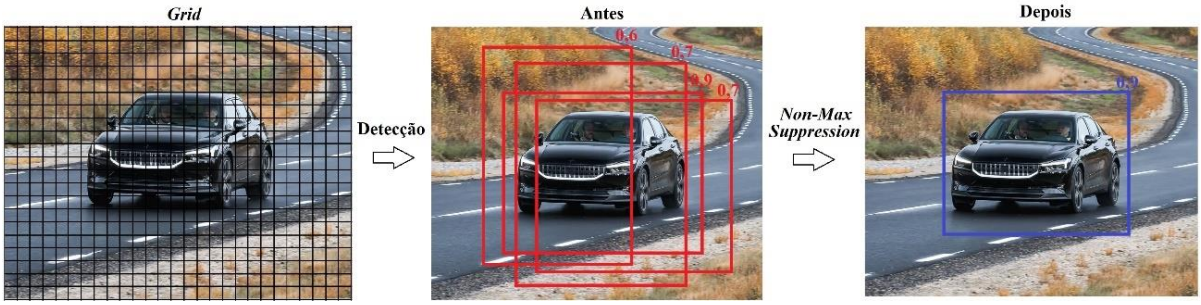
O papel da IoU está em avaliar a união e a intersecção das duas *bounding boxes*, cuja relação está expressa na Equação 5. Convencionalmente, atribui-se um *threshold* com valor superior à 50% de correspondência para julgamento e validação da posição correta da *bounding box* (BARBANERA, DEZANICIANCAGLINI, DELIGUORO, 1995; REZATOFIGH *et al.*, 2019).

$$IoU = \frac{Área\ de\ intersecção}{Área\ de\ união} \tag{5}$$

Outra questão sensível desta problemática tange à múltipla detecção do mesmo objeto, Isto acontece em decorrência de uma única célula analisada predizer que o objeto se encontra naquele local e as células vizinhas possuírem a mesma visão. Para a solução deste caso, utiliza-se o que é conhecido como *non-max suppression*, que atua "limpando", ou então, suprimindo as detecções para que somente seja contabilizado a exata quantidade de objetos existentes (NEUBECK, VAN GOOL, 2006; HOSAN, BENENSON, SCHIELE, 2017).

Isto é feito analisando a maior probabilidade de localização do objeto alcançada pelas *bounding boxes* próximas e associando-as com as suas respectivas uniões, ou seja, levando em consideração os valores descritos pela IoU. Em suma, as *bounding boxes* proximas que possuem o valor de P_c menor ou igual à 0,6, ou o valor de IoU menor ou igual à 0,5 são descartadas (NEUBECK, VAN GOOL, 2006; HOSAN, BENENSON, SCHIELE, 2017). Na Figura 36 pode-se observar um exemplo do efeito resultante da aplicação de *non-max supression*.

Figura 36 - Exemplo da aplicação da *non-max suppression*



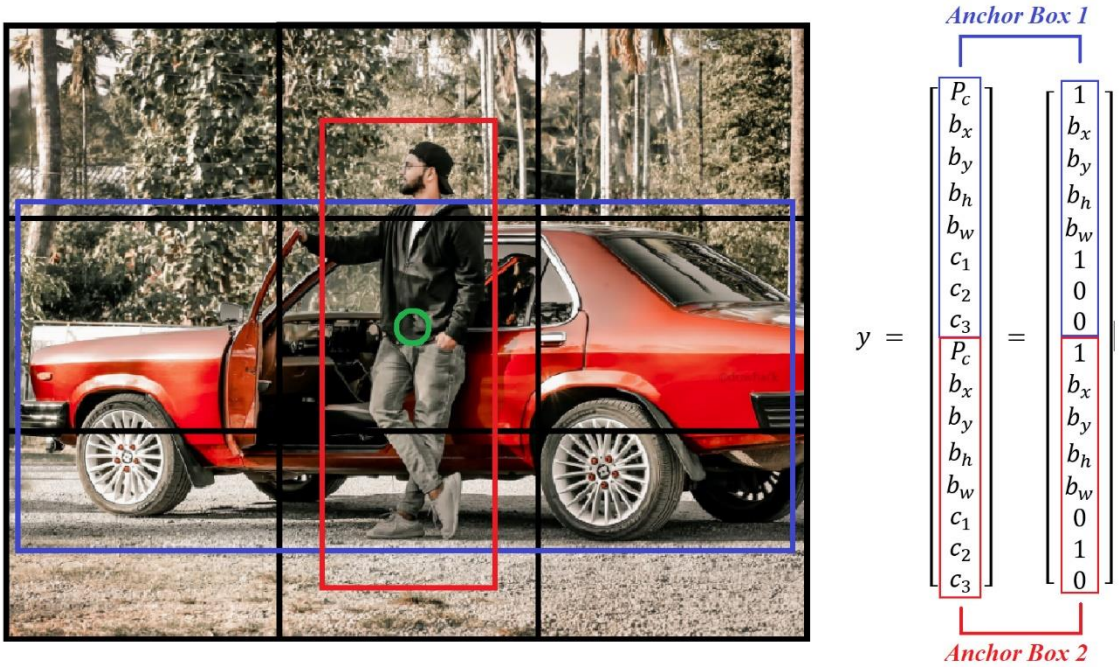
Fonte: Adaptado (VALENZUELA, 2019)

Outro grande ponto de atenção na classificação de imagens refere-se ao princípio que cada célula da imagem é capaz de detectar somente um único objeto, porém, é muito comum existirem objetos sobrepostos ou interferentes em cenários reais (REF).

Para solucionar este ponto, utiliza-se o que é conhecido como *anchor boxes*, algoritmo que permite a associação de duas ou mais predições para a mesma célula. Para isso, torna-se necessário empilhá-las no *target vector* a fim de permitir que, para uma mesma célula, haja a

possibilidade de detecção de duas ou mais classes simultaneamente (CHENG *et al.*, 2020; ZHONG *et al.*, 2020). A Figura 37 ilustra um exemplo da sobreposição de classes sob um mesmo ponto central e as *anchor boxes* associadas ao *target vector*.

Figura 37 - Exemplo do resultado da aplicação de *anchor boxes*



Fonte: Adaptado (MATIĆ *et al.*, 2018)

4 MATERIAIS E MÉTODOS

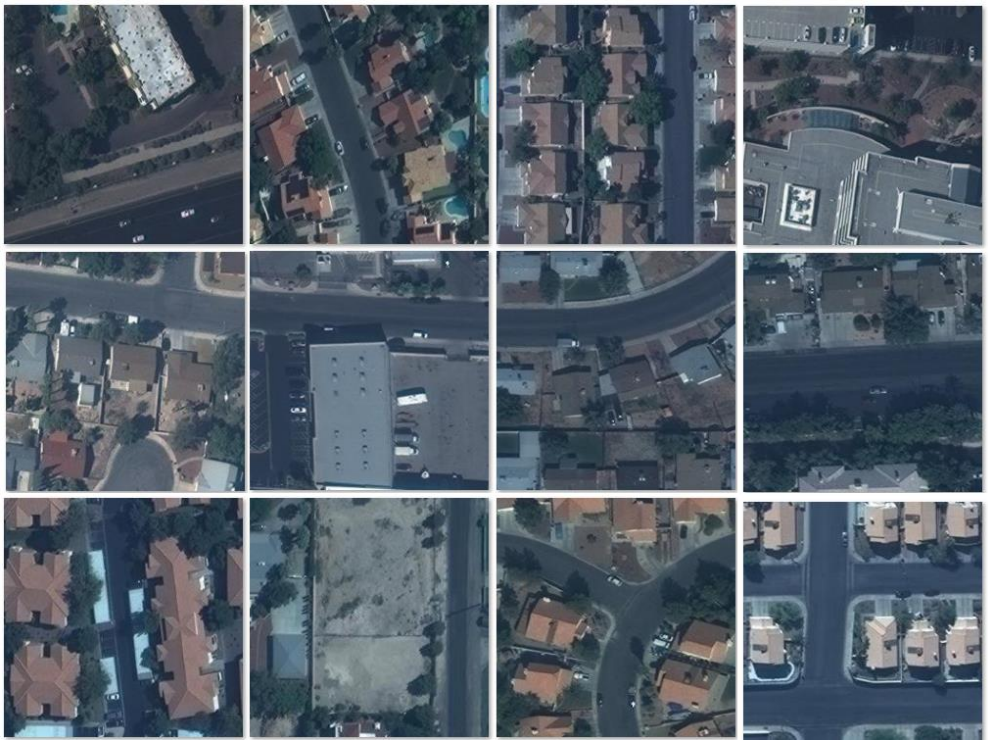
4.1 CONJUNTO DE DADOS

Considerando a tratativa envolvendo a detecção de edificações em imagens aéreas, torna-se necessário fornecer dados ao modelo computacional que representem a problemática envolvida. Para isso, face às distintas abordagens que poderiam ser adotadas, optou-se por escolher um conjunto de dados pré-processado já existente.

O conjunto de dados utilizado para o treinamento, validação e teste do modelo é conhecido como *Aerial Imagery for Roof Segmentation* (AIRS) e, como os próprios autores o descrevem, trata-se de um conjunto de dados de larga escala para automatização do mapeamento de edificações (CHEN *et al.*, 2020).

Este conjunto de dados, de disponibilização pública e gratuita, reúne cerca de 220.000 imagens aéreas ortorretificadas de edificações da cidade de Christchurch, em Nova Zelândia, com uma resolução espacial de 0,075m. Também cabe ressaltar que todas as imagens possuem as mesmas dimensões (300x300 pixels) e encontram-se previamente ajustadas, com certo grau de paralelismo, entre os telhados e as margens das imagens. A Figura 38 ilustra algumas destas imagens.

Figura 38 - Exemplo das imagens utilizadas do conjunto de dados AIRS



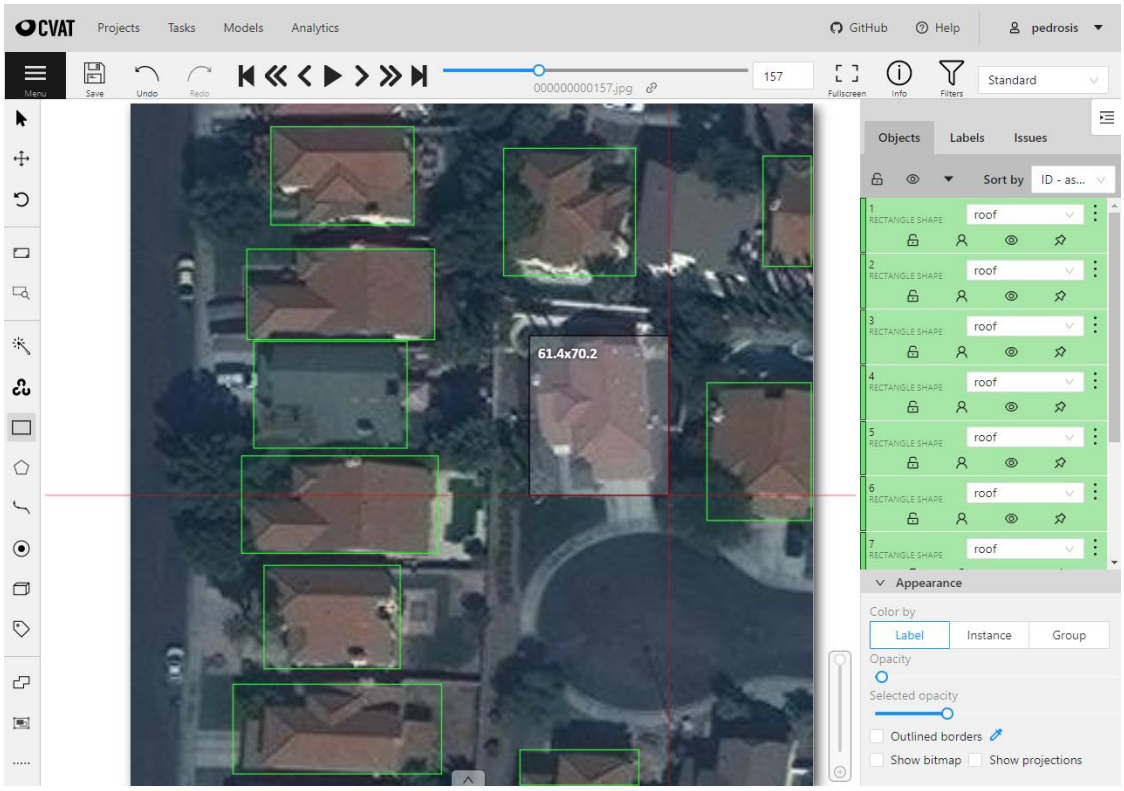
Fonte: Adaptado (AIRS, 2016).

Porém, também utilizou-se imagens aéreas obtidas por meio do *software Google Earth* para diversas regiões de interesse, que serão apresentadas ao modelo após o treinamento com as imagens disponíveis no AIRS. Estas imagens personalizadas contribuirão com a análise da robustez e da versatilidade do modelo, cuja abordagem contribuirá com a avaliação do poder de replicação de conhecimento do modelo.

Em posse dos dados, o próximo passo consiste na criação e a atribuição de rótulos aos dados. Para isso, utilizou-se outra ferramenta pública e gratuita desenvolvida pela Intel em 2018 conhecida como *Computer Vision Annotation Tool* (CVAT).

Escolhido o nome dos rótulos e as imagens, que no caso do presente trabalho foi um único rótulo denominado "*roof*" para 1000 imagens, iniciou-se o processo de rotulação dos dados. A Figura 39 mostra como a rotulação de cada telhado é feita para cada uma destas imagens.

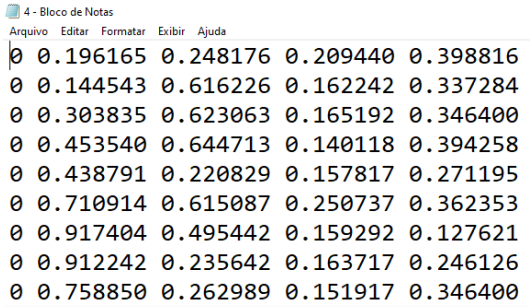
Figura 39 - Exemplo da rotulação dos telhados em cada imagem



Fonte: Elaborado pelo autor.

Ao final do processo é gerado, pela própria ferramenta, vários arquivos no formato *.txt* contendo informações referentes à cada uma das imagens A Figura 40 ilustra o conteúdo de um destes arquivos.

Figura 40 - Exemplo de um arquivo *.txt* gerado após a rotulação



Fonte: Elaborado pelo autor.

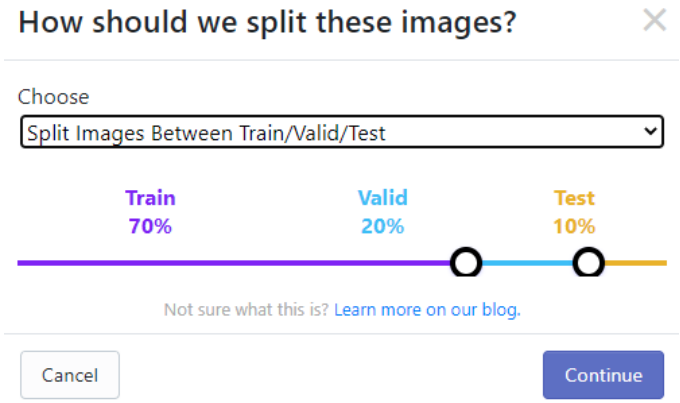
Observa-se que nestes arquivos existem 5 colunas de dados. Cada coluna representa um determinado parâmetro, sendo que, no caso da primeira coluna, comum para todas as linhas pelo número 0, denota que todos os elementos compactuam a mesma classe. Esta classe é única para este estudo e foi selecionada no início do processo como *roof*.

Já a segunda coluna denota a posição do centro do eixo transversal a partir do vértice superior esquerdo do registro, enquanto a terceira coluna denota a posição do centro do eixo longitudinal também a partir da mesma referência. A quarta coluna, por sua vez, denota a largura enquanto a quinta e última coluna denota o comprimento do polígono. Deste modo, a posição de cada rótulo, bem como suas dimensões, encontram-se bem definidas para cada registro.

Feito isso, utilizou-se uma plataforma conhecida como *Roboflow* para verificar se o conjunto de dados possuía imagens sem rótulos, duplicadas ou fora dos padrões das demais. Também realizou-se a distribuição das imagens, pré-processamentos e expansão do conjunto de dados. Cabe ressaltar que esta plataforma disponibiliza várias ferramentas de forma gratuita mediante cadastro.

Nesta plataforma, pode-se selecionar a quantidade de dados que serão utilizados para treinamento, validação e teste do modelo por meio de uma simples seleção, cuja proporção selecionada foi de 70%, 20% e 10%, respectivamente. A Figura 41 ilustra como é feita esta seleção.

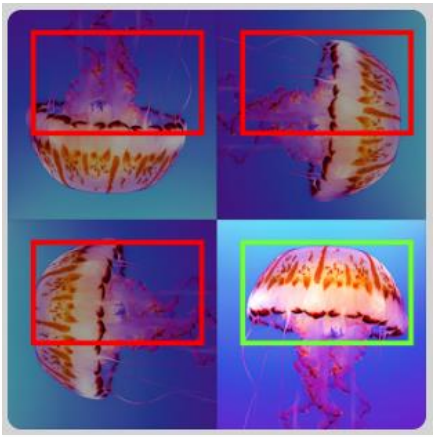
Figura 41 - Exemplo da proporção de subdivisão do conjunto de dados



Fonte: Adaptado (ROBOFLOW, c2021)

Como parte constituinte do pré-processamento do conjunto de dados, utilizou-se uma ferramenta de auto orientação para corrigir o desencontro das anotações, como ilustrado na Figura 42.

Figura 42 - Exemplo da aplicação da ferramenta de auto-orientação



Fonte: Adaptado (ROBOFLOW, c2021)

Também utilizou-se algumas ferramentas para expansão do conjunto de dados. Dentre essas ferramentas, utilizou-se rotações, saturações e ruídos, majoritariamente. A Figura 43 ilustra algumas das possibilidades disponíveis gratuitamente para utilização.

Figura 43 - Exemplos das possibilidades para expansão do conjunto de dados

Fonte: Adaptado (ROBOFLOW, c2021)

Também cabe pontuar que estas manobras utilizaram o conjunto de dados base para criar novas imagens, aumentando o volume de amostras de treinamento e, consequentemente, contribuir para a robustez do modelo. Por fim, os dados foram exportados em nuvem e um link foi gerado para sua importação no algoritmo.

4.2 GOOGLE COLABORATORY (GOOGLE COLAB)

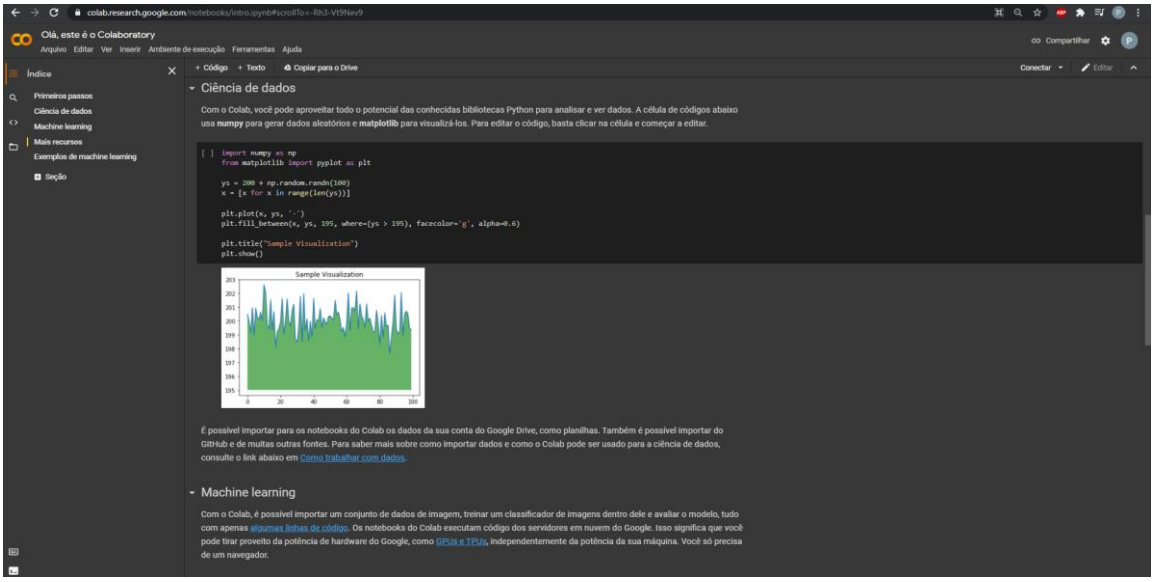
O desenvolvimento do modelo de aprendizado de máquina foi realizado utilizando a linguagem Python de programação. A razão pela qual escolheu-se esta linguagem está relacionada, entre outros quesitos (como a prévia familiaridade), na ascensão e notoriedade nas aplicações envolvendo aprendizado de máquina nos últimos anos. Para isso, optou-se em utilizar um ambiente conhecido como *Google Colaboratory* ou então *Google Colab*, como é usualmente mencionado.

O *Google Colab* é um ambiente que permite, entre outras vantagens, o desenvolvimento de *notebooks* para escrita do código diretamente do navegador sem a necessidade de instalação, ou mesmo de maiores configurações ou adequações prévias no ponto de acesso, além da fácil integração com ambientes em nuvem.

Porém, um dos maiores fatores que levaram à escolha do *Google Colab* como ambiente de desenvolvimento do modelo foi a disponibilidade gratuita, mesmo que limitada, de unidades

de processamento gráfico (GPUs) e de unidades de processamento de tensores (TPUs). Estas unidades otimizaram o tempo necessário para processar as solicitações impostas pelo modelo visto que, o computador utilizado no desenvolvimento deste trabalho, não possui grande capacidade de processamento comparado às unidades disponibilizadas pelo *Google Colab*. A Figura 44 exemplifica um ambiente de execução *Google Colab.*

Figura 44 - Exemplo do ambiente de execução do *Google Colab*



Fonte: GOOGLE, c2021.

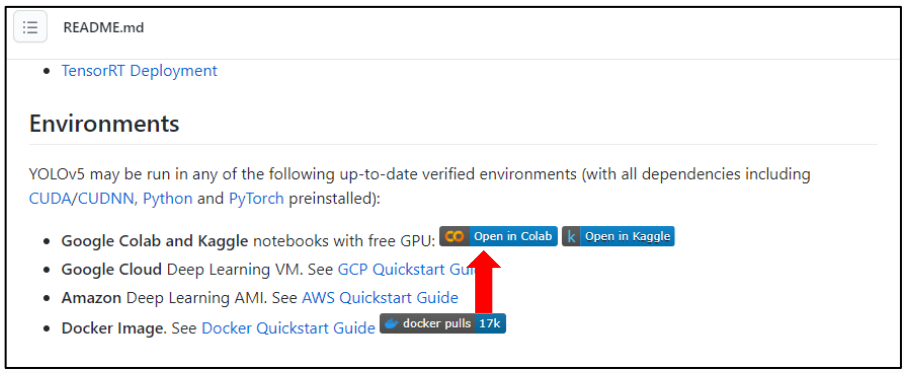
Portanto, com a familiarização concluída e o ambiente pronto para receber os próximos passos, foi iniciada o desenvolvimento do modelo computacional utilizando o algoritmo *You Only Look Once (YOLO)*.

4.3 YOU ONLY LOOK ONCE

O algoritmo *You Only Look Once* na sua 5ª versão, ou simplesmente mencionado como *YOLOv5* (ULTRALYTICS, c2021), é um algoritmo utilizado para detecção de objetos desenvolvido e disponibilizado gratuitamente pela *Ultralytics*© no *GitHub*©[©]. O *GitHub*©, por sua vez, pode ser descrito como uma plataforma em nuvem utilizada para disponibilização de códigos e demais arquivos para seus usuários.

Isto posto, decidiu-se utilizá-lo como ferramenta para detecção das edificações a fim de obter, utilizando um dos mais recentes e renomados recursos disponíveis, a melhor resposta possível para a tratativa em questão. O primeiro passo consiste na visita ao *GitHub*© para aquisição do código disponibilizado, cujo procedimento pode ser melhor visualizado na Figura 45.

Figura 45 - Acesso ao código-fonte por meio do *Google Colab* no *GitHub*©



Fonte: GITHUB, 2021.

Após o redirecionamento ao *Google Colab*, o próximo passo correspondeu à preparação do ambiente. Isto foi feito redirecionando o diretório, confirmando os requisitos necessários para execução do código, importação dos recursos necessários e informações sobre a unidade de processamento utilizada.

Toda a arquitetura do algoritmo encontra-se disponível para consulta no site do desenvolvedor que, por se tratar de um algoritmo de código aberto, permite acesso à todos os hiperparâmetros de modo explícito.

Ressalta-se aqui a utilização de um *framework* de código-aberto desenvolvido pelo *Facebook's AI Research Lab* - FAIR (FACEBOOK, c2021) em 2017 conhecido como *PyTorch*, e, desde então, amplamente utilizado para aplicações do gênero. Também cabe pontuar que a GPU disponibilizada para o desenvolvimento do modelo pelo *Google Colab* trata-se de uma *NVIDIA® Tesla T4* (NVIDIA, c2021).

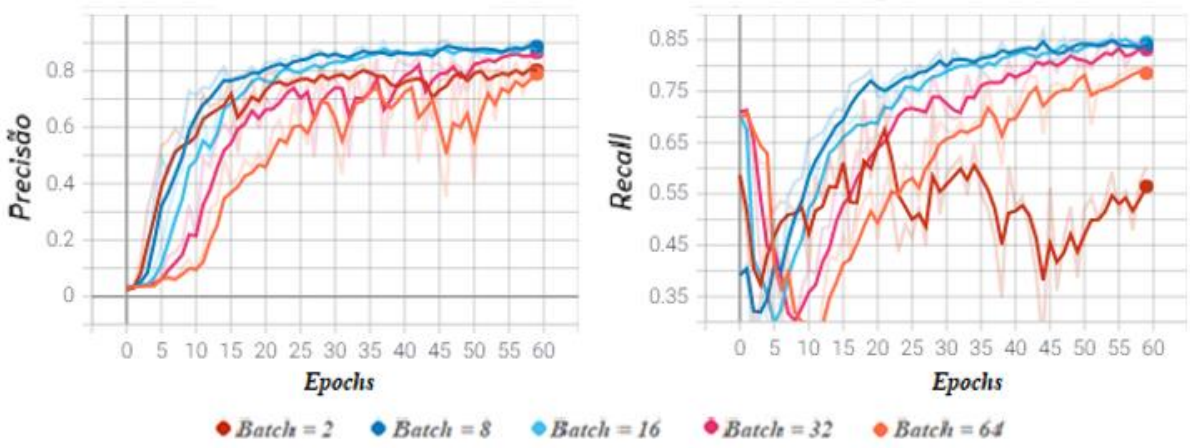
Cabe pontuar também que, durante a implementação do algoritmo, optou-se por conservar todos os hiperparâmetros genéricos que acompanham o modelo disponibilizado pelo desenvolvedor. Assim, será possível avaliar como o modelo se comportará frente ao conjunto de dados escolhido sem que haja intervenções durante sua implementação.

5 RESULTADOS E DISCUSSÃO

A fim de alcançar a melhor performance possível, avaliou-se o comportamento de diversos hiperparâmetros com o conjunto de dados disponibilizado. Em razão da enorme possibilidade de customização do modelo, bem como as diversas possibilidades de abordagem nos dados de treinamento e de teste, decidiu-se então avaliá-los individualmente.

Utilizando-se, inicialmente, 60 *epochs*, variou-se o número de *batches* e manteve-se todos os demais parâmetros fixos. A Figura 46 mostra uma relação de gráficos que ilustram, individualmente, o comportamento da precisão e do *recall*, em relação ao número de *epochs*, para cinco valores distintos de *batches*.

Figura 46 - Comportamento da precisão e do *recall* variando o número de batches



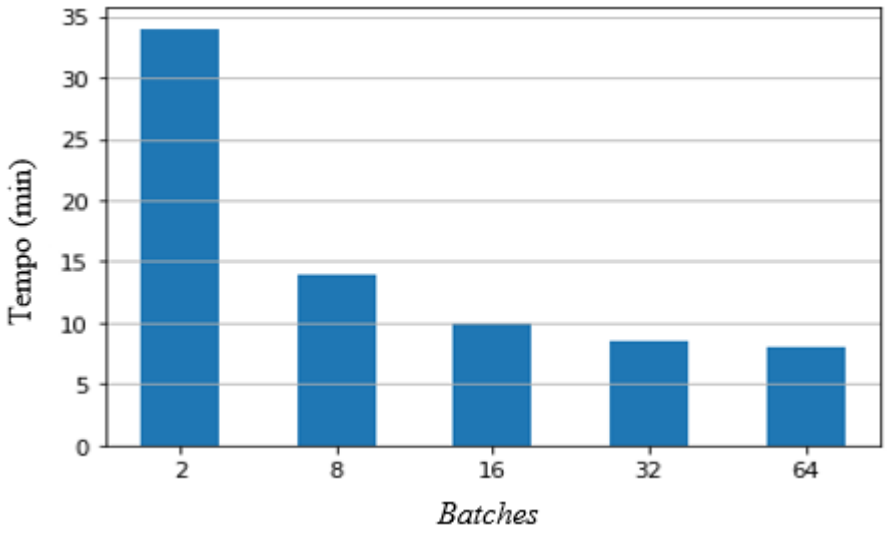
Fonte: Elaborado pelo autor.

A precisão e o *recall*, ilustrados em função do número de *epochs* mostram, respectivamente, a porção de objetos detectados corretamente em relação à todos os objetos detectados e a capacidade do modelo de detectar tais objetos. Logo, para a análise em questão, deseja-se sempre os maiores valores possíveis para esses dois parâmetros e, por isso, considerou-se o valor de *batch* igual a 16, adequado para este conjunto de dados sob estas condições.

Nota-se também que os piores desempenhos foram observados nos valores de *batch* extremos (2 e 64) para o intervalo adotado que, além de resultarem em baixos valores de precisão e *recall* para todas as *epochs*, apresentaram ascendências rebuscadas e inviabilizaram sua utilização. Já para valores de *batch* intermediários, como 8 e 16, notou-se comportamentos similares, cujo critério de seleção levou-se em conta, entre outros parâmetros, a estabilidade de ascensão e os melhores valores de precisão e *recall* para um mesmo *epoch*.

Cabe ressaltar que os tempos necessários para o treinamento de cada um desses cenários também foram ponderados em busca da solução mais ágil e, para ilustrar este comportamento, elaborou-se a Figura 47.

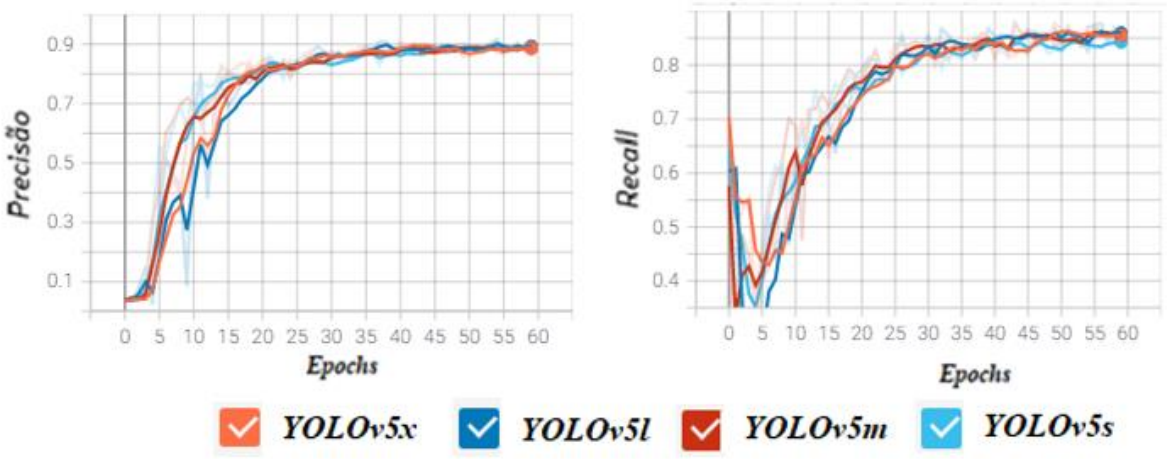
Figura 47 - Relação entre o tempo de treinamento e o número de *batches*



Fonte: Elaborado pelo autor.

Também decidiu-se avaliar quais dos modelos disponíveis no YOLOv5 melhor se adaptaria ao conjunto de dados e hiperparâmetros estipulado e, para isso, variou-se o treinamento utilizando os quatro modelos disponíveis (YOLOv5s, YOLOv5m, YOLOv5l e YOLOv5x). A Figura 48 ilustra o comportamento da precisão e recall destes modelos em função do número de *epochs*.

Figura 48 - Comportamento da precisão e do *recall* variando o modelo utilizado



Fonte: Elaborado pelo autor.

Apesar de todos os modelos alcançarem precisões e *recalls* semelhantes nos maiores valores de *epoch*, nota-se além de maiores estabilidades nos modelos YOLOv5m e YOLOv5s, quantidades significativamente menores de camadas e pesos, implicando em treinamentos mais

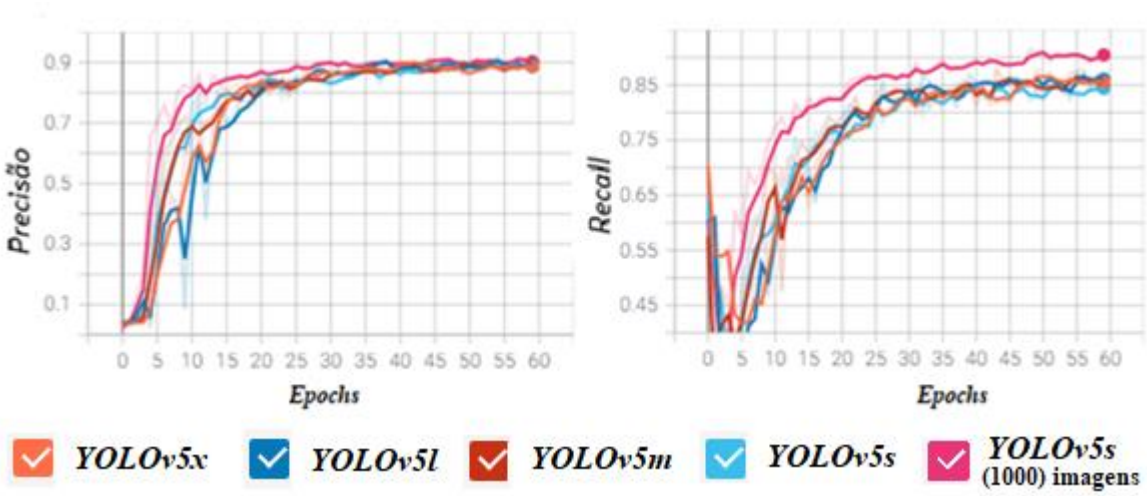
rápidos e menos propícios à sobreajustes. Portanto, na análise de outros parâmetros adotou-se o modelo YOLOv5s por demonstrar equilíbrio entre performance e recursos computacionais.

Estas análises levaram em consideração um conjunto de dados contendo 500 imagens rotuladas e, a fim de otimizar o desempenho do modelo, optou-se por variar este volume para avaliar sua eficiência em razão do volume de dados disponíveis. Notou-se que para pequenos conjuntos de dados, utilizando cerca de 100 imagens distribuídas entre treinamentos e validação, a precisão e o *recall* do modelo eram significativamente comprometidas. Neste cenário obteve-se cerca de 60% de precisão e 55% de *recall*, fatores que demonstraram que o modelo estava, no âmbito do volume de dados, pouco alimentado, prejudicando as métricas obtidas.

Então, aumentou-se este volume para um total de 1000 imagens, distribuídas em 70% em dados de treinamento, 20% em dados de validação e 10% em dados de teste. Neste cenário, houve um aumento pouco pronunciado nos valores de precisão (menor que 1%), porém com acentuação nítida nos valores de *recall* (cerca de 5%). Cabe ressaltar que, na abordagem utilizada para detecção, o valor de *recall* influencia de modo tão significativo quanto a precisão, fator que culminou na escolha do maior conjunto de dados para treinamento do modelo.

A Figura 49 ilustra o desempenho desta abordagem em relação aos dados anteriores, mostrada pela curva de coloração rosa.

Figura 49 - Influência do aumento do volume de dados na curva de coloração rosa

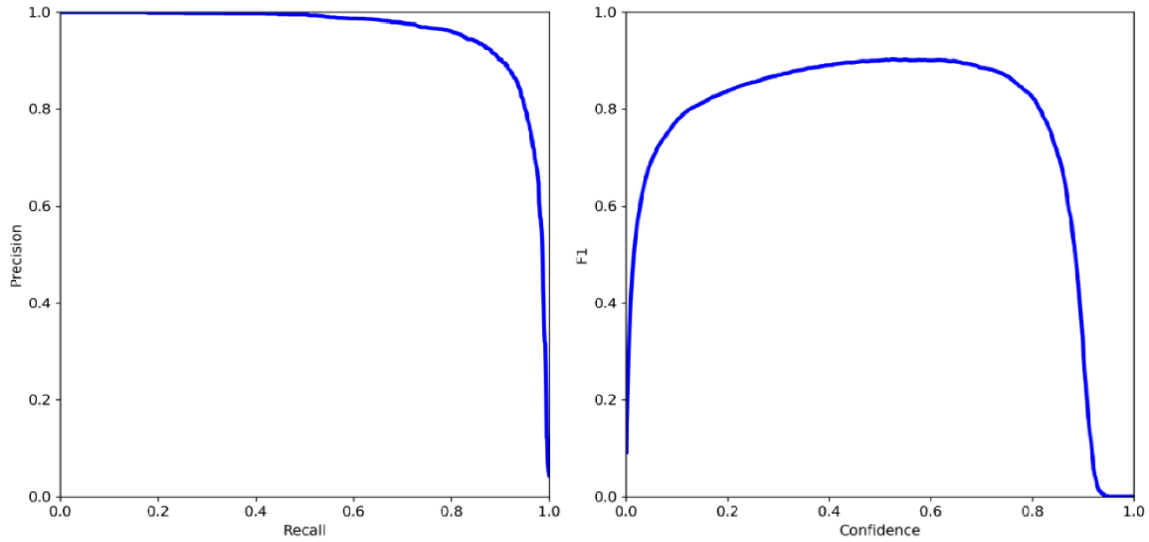


Fonte: Elaborado pelo autor.

A atuação do modelo frente aos níveis de precisão e *recall* pode ser melhor observado quando relacionamos os dois parâmetros por meio de um gráfico. No primeiro gráfico ilustrado na Figura 50, têm-se a relação direta entre estas duas grandezas.

Já no segundo gráfico da Figura 50, observa-se o comportamento de um parâmetro conhecido como F_1 em relação à confiança do modelo. Por se tratar da média harmônica entre os valores de precisão e *recall* deseja-se, em suma, obter o valor de F_1 o mais próximo possível do valor unitário, levando a detecção ao nível de perfeição. Ambas as curvas ilustradas na Figura 50 foram geradas considerando o último treinamento realizado, ou seja, utilizando o modelo embarcado no YOLOv5s, com todos os hiperparâmetros preestabelecidos inicialmente pelo modelo, utilizando 1000 imagens no conjunto de dados e considerando *batch* igual à 16.

Figura 50 – Relações entre precisão, *recall*, a pontuação F_1 e a confiança do modelo

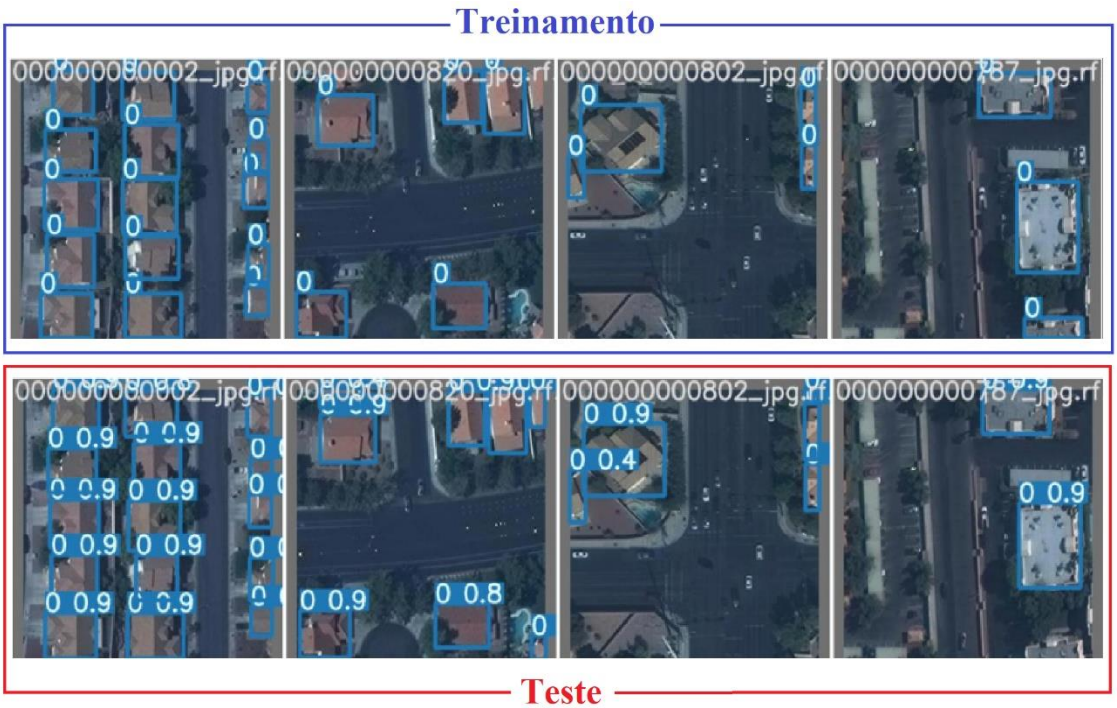


Fonte: Elaborado pelo autor.

Deste modo, observa-se que, para o nível de treinamento empregado, bem como pela utilização de hiperparâmetros genéricos, o modelo apresentou pontuações que podem ser consideradas suficientes para a análise esperada dos resultados em ambientes diversos aos apresentados ao algoritmo. O cenário ideal envolvendo a precisão e *recall* ao patamar considerado satisfatório para a análise sempre será objeto de desejo e, levar o modelo próximo desta meta, condiz com a busca pelo aperfeiçoamento contínuo em busca de resultados aprimorados.

A fim de ilustrar inicialmente o desempenho do modelo, elaborou-se a Figura 51. Esta figura compara os dados utilizados no treinamento do modelo, manualmente rotulados, com as predições desempenhadas nestes mesmos dados.

Figura 51 - Performance do modelo nos dados utilizados para treinamento



Fonte: Elaborado pelo autor.

Observa-se que o modelo classificou e localizou as edificações com êxito quando comparado à análise visual humana. Tal desempenho retrata a capacidade do modelo de aprender, com relativa eficácia, os padrões necessários para caracterizar uma edificação com base no conjunto de dados fornecido, justificando a precisão e o *recall* obtidos durante o treinamento.

Porém, o objetivo do modelo é desempenhar esta mesma performance em dados jamais vistos anteriormente, generalizando e perpetuando o aprendizado. Para avaliar este comportamento, solicitou-se ao modelo algumas predições com base em um conjunto de dados de teste.

As imagens utilizadas neste teste foram obtidas de mesmo modo que as imagens utilizadas para treinamento e validação do modelo, porém, cabe ressaltar que tratam-se de imagens diferentes, que não sofreram rotulação prévia e não foram “vistas” pelo modelo até então. A fim de ilustrar o desempenho do modelo nesses dados, elaborou-se a Figura 52.

Figura 52 - Exemplo de algumas imagens utilizadas no teste do modelo

Fonte: Elaborado pelo Autor

Analisando a Figura 52, observa-se que o desempenho do modelo pode ser comparado com a percepção humana natural para distinção de edificações próximas à pavimentos e vegetação, por exemplo.

No entanto, nota-se dificuldades do modelo na distinção das edificações quando as amostras possuem estacionamentos, galpões e solos expostos, por exemplo. Estas dificuldade já eram esperadas em razão do baixo número de amostras destas estruturas. Muitas das vezes essa distinção torna-se nebulosa inclusive para a análise humana. A própria rotulação das edificações no processo de treinamento pode ter sido prejudicada por fatores do observador que, inevitavelmente, refletem no modo como o modelo interpreta essas regiões.

Também é pertinente ressaltar a múltipla detecção da mesma edificação pelo modelo que, por apresentar telhados segmentados ou então dispostos em formatos irregulares, causam a interpretação de um número maior de edificações. Tal fenômeno pode ser discutido tanto no âmbito do valor de *threshold* da *IoU*, quanto na ausência de exemplos envolvendo telhados irregulares, que não eram abundantes no conjunto de dados utilizados no treinamento. De mesmo modo também a detecção de falsos positivos e, para esta discussão, elaborou-se a Figura 53.

Figura 53 - Exemplos de falsos positivos e incoerência na detecção

Fonte: Elaborado pelo autor.

Observando a Figura 53, nota-se nítidas imprecisões do modelo perante alguns dos casos mencionados anteriormente. Regiões que possuem breve semelhanças com edificações tornam-se propícias para tornarem-se falsos positivos, fator que poderia ser relevado caso tal semelhança confundisse um observador humano.

Porém, em alguns casos, estes falsos positivos mostram-se nitidamente incoerentes e seriam facilmente percebidos em uma análise visual. Este comportamento aparece principalmente em virtude do alto valor de *recall* adquirido pelo modelo durante o treinamento, refletindo no incremento da sensibilidade de detecção.

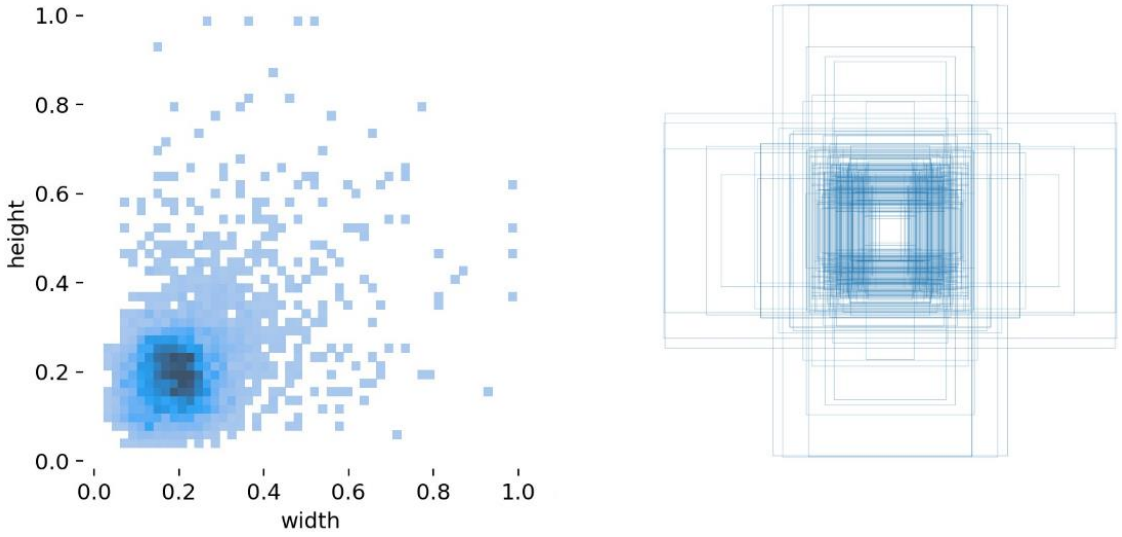
No entanto, em razão da natureza do problema, deseja-se manter o valor de *recall* alto o suficiente para que todas as edificações sejam identificadas. Este caso difere-se de um

diagnóstico clínico de tumores por análise de imagens, por exemplo, onde geralmente pondera-se, em maior instância, a precisão do diagnóstico do que o volume de detecções em uma mesma amostra.

Claro que, para o modelo atingir maiores níveis de eficiência, deve-se ajustá-lo até atingir precisões e *recall* considerados satisfatórios para a aplicação em questão. No caso do exemplo analisado, para o grau de eficácia esperado em razão do nível de treinamento empregado, pode-se admitir certo volume de falsos positivos em prol da detecção do maior número possível de edificações em cada imagem. Deste modo, os níveis de precisão e *recall* (90,8% e 91,2%, respectivamente) satisfizeram as expectativas do autor e foram considerados adequados para a sequência das análises.

Para entender alguns desses comportamentos do modelo, elaborou-se a Figura 54. Esta figura relaciona as dimensões das *bounding boxes* utilizadas durante os testes.

Figura 54 - Distribuição das dimensões das *bounding boxes* durante as detecções



Fonte: Elaborado pelo autor.

Analisando o gráfico esquerdo da Figura 54, observa-se que há uma concentração das dimensões das *bounding boxes* fornecidas pelo modelo durante os testes que, em sua grande maioria, concentraram-se em torno de 20% da dimensão das imagens de teste. Isso implica que a maior parte das *bounding boxes* possuíam estas dimensões, fator que induz a direção de detecção do modelo para edificações que possuem estas dimensões nas imagens analisadas.

Outro aspecto importante diz respeito às *bounding boxes* que apresentaram grandes dimensões que, apesar de escassas, estão presentes no conjunto de dados e são computadas pelo modelo. O fato de estarem presentes na análise induz o modelo à preparação para sua detecção

e, consequentemente, amplia o campo de atuação, colaborando tanto para a robustez da análise quanto para a imprecisão em alguns casos específicos.

Ademais, a ilustração presente na porção direita da Figura 54 representa a distribuição geométrica dessas *bounding boxes*, ilustrando de modo visual como esta frequência de incidência foi representada nos dados de teste.

A fim de experimentar o desempenho alcançado pelo modelo, optou-se por implementá-lo em diversos cenários envolvendo imagens aéreas variadas. Como comentado anteriormente, a premissa deste modelo é detectar o maior número possível de edificações em uma imagem aérea, auxiliando e automatizando a detecção manual, visando principalmente a redução de esforços humanos sem abrir mão da confiabilidade da análise.

Para avaliar o desempenho, a robustez e a capacidade de replicar o conhecimento de modo genérico, utilizou-se imagens de variadas resoluções, dimensões, ampliações e características, envolvendo desde regiões conhecidas até aleatórias do globo terrestre. Algumas destas imagens utilizadas nesse teste podem ser observadas nas Figuras 55 e 56.

Figura 55 - Exemplo da implementação do modelo em cenários diversos

Fonte: Elaborado pelo autor.

Figura 56 - Exemplo de outra avaliação do modelo em cenários diversos

Fonte: Elaborado pelo autor.

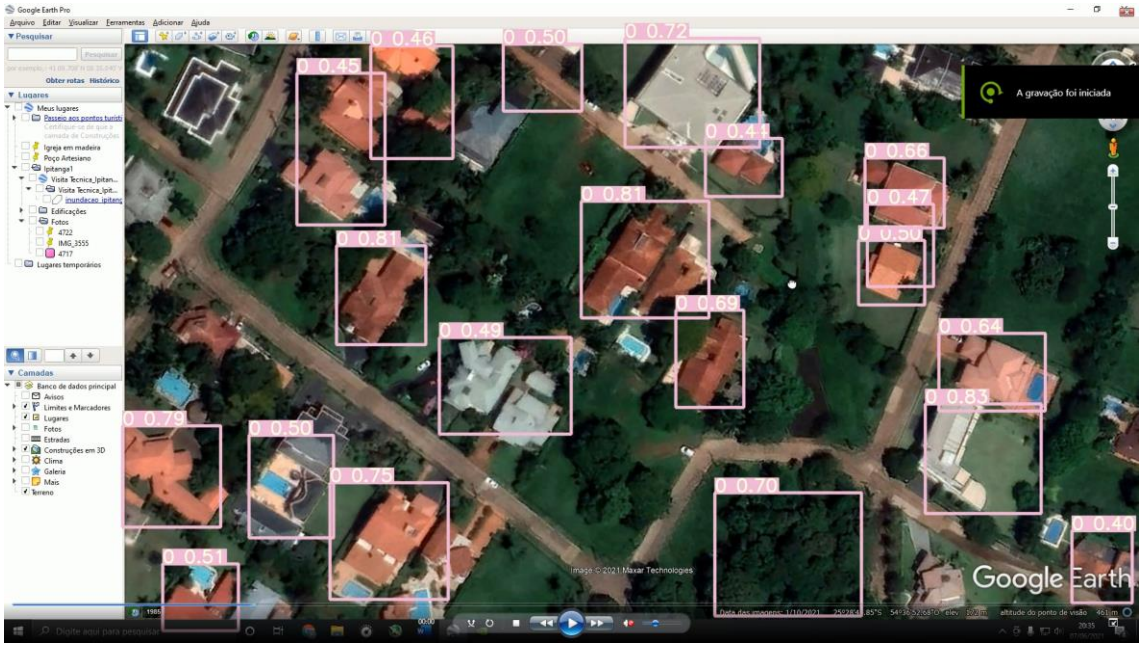
Analisando as figuras mostradas anteriormente, é possível observar que o modelo desempenhou classificações esperadas para o nível de treinamento empregado. Mesmo a utilização de parâmetros genéricos do algoritmo já permite alcançar os níveis de precisão e *recall* responsáveis pela distinção e localização das edificações em meio à estradas e vegetação, por exemplo.

Nota-se, no entanto, a mesma problemática utilizada nos dados anteriores de treinamento: detecções de falsos positivos e a não detecção de edificações que estavam nítidas ao olhar humano.

Uma grande ferramenta do YOLO diz respeito à detecção de objetos em tempo real. Isso significa que as imagens podem ser analisadas não só em abordagens estáticas, mas em tempo real com dispositivos de captura de imagens atrelados ao algoritmo. Essa condição permite a integração do modelo com diversos recursos disponíveis para auxiliar os usuários na detecção de edificações, como por exemplo no emprego de *drones.*

Para avaliar esta capacidade e o desempenho do modelo, elaborou-se alguns vídeos navegando por imagens de satélite enquanto o modelo detectava as edificações presentes na tela do usuário, simulando o que seria visto por um *drone* durante um voo de varredura por uma região de interesse. A Figura 57 ilustra o que foi observado em uma destas análises.

Figura 57 - Exemplo da análise de vídeos pelo algoritmo



Fonte: Elaborado pelo autor.

Durante a análise do vídeo, a detecção de edificações mostra-se condizente com as abordagens estáticas de imagens, refletindo comportamentos semelhantes em relação aos êxitos e aos erros cometidos pelo modelo. No entanto, ressalta-se a grande velocidade com que as detecções são realizadas a medida que as novas edificações aparecem, não havendo penalidades na taxa de quadros por segundo. Ou seja, o modelo mostrou-se capaz de desempenhar análises em tempo real consideradas satisfatórias para o nível esperado, superando a taxa de atualização de 60 Hz nas solicitações deste estudo.

Em relação à sua eficácia geral, diversos fatores culminam para que o resultado obtido não reflita de modo perfeito a capacidade de detecção das edificações. O conjunto de dados utilizado no treinamento do modelo foi obtido de uma única região, fator que contribui para que o modelo tenha baixa percepção de edificações que apresentam arquiteturas diversas, disposições geográficas e aspectos envolvendo pavimentos (asfáltico e de terra) diferentes dos dados empregados no treinamento, por exemplo. Áreas rurais ou grandes centros, que apresentam extremos frente à densidade demográfica, são outros exemplos da complexidade envolvida neste tipo de detecção.

Portanto, para elevar a eficácia do modelo e torná-lo mais robusto frente às novas solicitações, diversas abordagens podem ser empregadas. Algumas destas abordagens dizem respeito à:

- Utilização de dados de treinamento que melhor se aproximam aos dados que serão empregados nas imagens de teste;

- Aumento do volume total do conjunto de dados envolvendo diversas regiões diferentes;
- Desenvolvimentos dos pesos personalizados ao invés da utilização de pesos genéricos pré estipulados;
- Adequação e testes de outras funções de ativação;
- Emprego de filtros em busca de maiores evidências de bordas e geometrias.

Mesmo com tantas variáveis envolvidas em um problema tão complexo, notou-se que o modelo conseguiu desempenhar resultados satisfatórios para os patamares inicialmente estipulados, principalmente em razão da simplicidade de implementação quando comparado à construção completa de algoritmos. Análises, que até décadas passadas despendiam de grandes esforços por parte do desenvolvedor, podem ser realizadas, hoje, com relativa simplicidade, demonstrando que a evolução no segmento está contribuindo para que todas estas ferramentas estejam cada vez mais próximas dos usuários comuns.

Porém, o aperfeiçoamento das análises depende da otimização tanto do modelo computacional quando do conjunto de dados utilizados no treinamento. Levando em consideração o cenário de população residente a jusante de barragens, a omissão de sua detecção poderá implicar em risco à vida ou à propriedade e, portanto, a utilização de modelos com hiperparâmetros genéricos e conjunto de imagens diferentes da realidade dos empreendimentos, implica diretamente na necessidade de supervisão humana dos resultados obtidos.

Apesar destes resultados refletirem a capacidade do modelo em replicar o conhecimento adquirido no conjunto de dados fornecido para o nível de treinamento empregado, tal condição não é suficiente para que o modelo seja capaz de desempenhar estas detecções com a confiabilidade necessária para assegurar o mapeamento completo da região.

No entanto, o desenvolvimento da atividade contribuiu para a visão técnica do autor e sanou o anseio que cerceava o trabalho. As diversas possibilidades de otimização do modelo trazem futuras expectativas e colaboram para fomentar o desenvolvimento da ciência e da cultura de segurança embarcada neste trabalho.

6 CONCLUSÃO

A Inteligência Artificial, hoje presente de modo substancial no cotidiano, auxilia e simplifica cada vez mais abordagens que até então eram dispendiosas, ou até intratáveis. A visão computacional, por sua vez, contribui, de modo ascendente e acelerado, em análises nos mais diferentes escopos possíveis, como na medicina, ciências aeroespaciais e até mesmo em redes sociais, por exemplo.

O manejo destas ferramentas vem gradualmente tornando-se parte essencial do cotidiano e as soluções tecnológicas envolvendo estas áreas abrem espaços que contribuem em setores envolvendo desde o conforto até a própria segurança das pessoas.

A concepção de uma ferramenta que realizasse uma varredura em uma imagem aérea e identificasse pontos de interesse é objeto de amplo estudo, principalmente por envolver diversas tratativas possíveis. Neste cenário, a identificação e localização de edificações situadas a jusante de barragens em imagens aéreas constitui uma destas aplicações e contribui para a rápida atuação de entidades de proteção e de resgate envolvendo cenários emergenciais destas estruturas.

Assim, tendo em vista a versatilidade desta ferramenta, a implementação do algoritmo transpareceu a complexidade envolvendo a detecção de edificações, porém, ilustrou a capacidade e, principalmente, a praticidade no desenvolvimento de modelos computacionais dessa natureza nos dias atuais.

Deste modo, a partir dos resultados obtidos, verificou-se que o modelo, mesmo com certos pontos de incoerência, é capaz de distinguir e detectar as edificações em diversas solicitações impostas, desempenho que pode ser optimizado na adequação dos parâmetros genéricos empregados e no conjunto de dados utilizado no desenvolvimento do aprendizado.

Diversos são as contribuições que compõem este resultado final. O conjunto de dados utilizado, por exemplo, foi formado a partir de uma única região que possui características próprias, seja na vegetação, nas singularidades das edificações ou no próprio modo como foram coletadas. Os parâmetros utilizados no treinamento, como os pesos, foram estipulados para um cenário genérico, cuja adequação à problemática em questão poderia surtir melhores efeitos na detecção.

Em meio à uma série de variantes, nota-se grandes oportunidades de aprimoramento do modelo para equivalência ao discernimento humano em tratativas como esta. A lapidação do modelo consiste em uma etapa delicada, porém essencial para atingir resultados confiáveis e precisos. Tais oportunidades devem ser exploradas a fim de propiciar, ao final das análises,

maior confiabilidade e agilidade nas detecções, requerendo progressivamente menores contribuições humanas.

Portanto, apesar do resultado obtido não retratar com fidelidade o mesmo desempenho que seria alcançado por olhares humanos, principalmente em razão dos hiperparâmetros e do conjunto de dados utilizado, deve-se ponderar a versatilidade e a gama de otimizações possíveis que rodeiam esta análise.

Dentre as possibilidades para elevar o estudo à níveis de excelência, cabe ressaltar alguns dos próximos passos. A aquisição de dados para o treinamento condizentes com os dados que o modelo será empregado constitui uma parte fundamental para o sucesso da análise, cujo constante aprimoramento culminará progressivamente ao resultado esperado. Explorar as diversas combinações hiperparâmetros, como por exemplo as funções de ativação, número de camadas e neurônios e a taxa de aprendizagem, podem resultar em melhores generalizações e aumentar a eficácia das detecções.

Cabe ressaltar a grandiosidade que cerceia tais ferramentas da visão computacional, cuja análise pode ser vinculada à diversos dispositivos e levada à campo. Um exemplo desta aplicação baseia-se no acoplamento desta ferramenta em *drones*, buscando o mapeamento, a contabilização e o acompanhamento de áreas impactadas par auxiliar no resgate e, principalmente, na prevenção de danos.

Por fim, pontua-se que o desenvolvimento deste trabalho colaborou com a ambientação das ferramentas utilizadas no campo da visão computacional, abrindo o campo de possibilidades para aprimoramentos e consolidando a metodologia utilizada para detecção de edificações no campo da visão computacional. Muito foi absorvido em relação à este cenário tão enfatizado atualmente e, participar, mesmo que de modo singelo desse desenvolvimento, colabora para a otimização, praticidade e evolução desta área em todas suas linhas de atuação.

REFERÊNCIAS BIBLIOGRÁFICAS

ACHARYA, T.; RAY, A. R. **Image Processing: Principles and Applications.** 1 ed. Hoboken, New Jersey: Wiley-Interscience, 2005. 452 p.

ADELSON, E. H. *et al.* Pyramid methods in image processing. **RCA engineer**, v. 29, n. 6, p. 33-41, 1984.

AIRS – Aerial Iagery for Roof Segmentation. **Air Dataset**. [*S.l.*], 2016. Disponível em: https://www.airs-dataset.com/. Acesso em: 5 de maio de 2021.

AL-AMRI, S. S.; KALYANKAR, N. V.;KHAMITKAR, S. D. Image segmentation by using edge detection. **International journal on computer science and engineering**, v. 2, n. 3, p. 804-807, 2010.

ANDRYCHOWICZ, M. *et al.* Learning to learn by gradient descent by gradient descent. **Advances in neural information processing systems**. p. 3981-3989, 2016.

ANTWEILER, W; COPELAND, B. R.; TAYLOR, M. S. Is free trade good for the environment?. **American economic review**, v. 91, n. 4, p. 877-908, 2001.

AQEL, M. J.; ALQADI, Z. RGB Color Image Encryption-Decryption Using Image Segmentation and Matrix Multiplication. **International Journal of Engineering & Technology.** v. 7, n. 3.13, p. 104-107, 2018.

ARVETTI, M.; GINI, G.; FOLGHERAITER, M.. Classification of EMG signals through wavelet analysis and neural networks for controlling an active hand prosthesis. *In:* 2007 IEEE 10th international conference on Rehabilitation Robotics, 2007, Noordwijk, Netherlands. **Paper** […] Noordwijk: IEEE, 2007. p. 531-536. doi: 10.1109/ICORR.2007.4428476.

ASIMOV, I. **The road to infinity.** 1 ed. United States: Avon Books, 1981. 233 p.

AVRIEL, M.; ZANG, I. Generalized arcwise-connected functions and characterizations of local-global minimum properties. **Journal of Optimization Theory and Applications**, v. 32, n. 4, p. 407-425, 1980.

AZIZ, A. R. A.; WONG, K. -F. V. A neural-network approach to the determination of aquifer parameters. **Groundwater**, v. 30, n. 2, p. 164-166, 1992.

BAI, C; ZENG, G. L.; GULLBERG, G. T. A slice-by-slice blurring model and kernel evaluation using the Klein-Nishina formula for 3D scatter compensation in parallel and converging beam SPECT. **Physics in Medicine & Biology**, v. 45, n. 5, p. 1275, 2000.

BARBANERA, F.; DEZANICIANCAGLINI, M.; DELIGUORO, U. Intersection and union types: syntax and semantics. **Information and Computation**, v. 119, n. 2, p. 202-230, 1995.

BENGIO, Y. *et al.* Convex neural networks. *In*: Weiss, Y., Scholkopf, B., & Platt, J. (Eds.), **Advances in Neural Information Processing Systems 18**, 1 ed. Cambridge, Massachusetts: MIT Press, 2006. p. 123–130.

BENGIO, Y. Gradient-based optimization of hyperparameters. **Neural computation**, v. 12, n. 8, p. 1889-1900, 2000.

BERNARDOS, P. G.; VOSNIAKOS, G. -C. Optimizing feedforward artificial neural network architecture. **Engineering Applications of Artificial Intelligence.** v. 20, n. 3, p. 365-382, 2007.

BLUM, M. *et al.* A learned feature descriptor for object recognition in rgb-d data. In: **IEEE International Conference on Robotics and Automation,** 2012, Saint Paul, Minnesota. **Paper** […] Saint Paul: IEEE, 2012. p. 1298-1303.

BOCHKOVSKIY, A.; WANG, C. -Y.; LIAO, H. -Y. M. Yolov4: Optimal speed and accuracy of object detection. **arXiv preprint arXiv:2004.10934**, p. 1-17, 2020.

BORJI, A. *et al.* Salient object detection: A benchmark. **IEEE transactions on image processing**, v. 24, n. 12, p. 5706-5722, 2015.

BOTTOU, L. Large-Scale Machine Learning with Stochastic Gradient Descent. *In:* 19th International Conference on Computational Statistics, 2010, Paris, France. **Anais** […] Paris: Physica-Verlag, p. 177-186. doi: https://doi.org/10.1007/978-3-7908-2604-3_16.

BOTTOU, L. Stochastic Gradient Descent Tricks. *In:* MONTAVON, G. ORR, G. B.; MÜLLER, K. -R (Ed). **Neural Networks:** Tricks of the trade. 2 ed. Cornell: Springer, Berlin, Heidelberg, 2012. Big Learning in Deep Neural Networks, p. 421-436.

BOWLING, M; VELOSO, M. Multiagent learning using a variable learning rate. **Artificial Intelligence**, v. 136, n. 2, p. 215-250, 2002.

BRAHMA, P. P.; WU, D.; SHE, Y. Why deep learning works: A manifold disentanglement perspective. **IEEE transactions on neural networks and learning systems**, v. 27, n. 10, p. 1997-2008, 2015.

BRASIL. **Lei** nº 12.334, de 20 de setembro de 2010. Estabelece a Política Nacional de Segurança de Barragens destinadas à acumulação de água para quaisquer usos, à disposição final ou temporária de rejeitos e à acumulação de resíduos industriais, cria o Sistema Nacional de Informações sobre Segurança de Barragens e altera a redação do art. 35 da Lei no 9.433, de 8 de janeiro de 1997, e do art. 4o da Lei no 9.984, de 17 de julho de 2000. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2010/lei/l12334.htm. Acesso em: 22 de agosto de 2021.

BROWNLEE, J. What is the Difference Between a Batch and an Epoch in a Neural Network?. **Machine Learning Mastery**, v. 20, p. 1-5, 2018.

CAI, L; ZHU, Y. The challenges of data quality and data quality assessment in the big data era. **Data science journal**, v. 14, n. 2, p. 1-10, 2015.

CHAKRABARTY, N. A deep learning method for the detection of diabetic retinopathy. *In:* 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2018, Gorakhpur. India. **Paper** […] Gorakhpur: IEEE, 2018. p. 1-5. doi: 10.1109/UPCON.2018.8596839.

CHEN, Q. et al. Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings. **ISPRS Journal of Photogrammetry and Remote Sensing**. v. 147, p. 42-55, 2019.

CHEN, T-J. *et al.* A blurring index for medical images. **Journal of digital imaging**, v. 19, n. 2, p. 118, 2006.

CHEN, W. *et al.* YOLO-face: a real-time face detector. **The Visual Computer**, v. 37, p. 1-9, 2020.

CHEN, Y. *et al.* Ship detection in optical sensing images based on YOLOv5. *In*: Twelfth International Conference on Graphics and Image Processing (ICGIP), 2020, Xi'na, China. **Anais** […] Xi'an: International Society for Optics and Photonics, 2021. p. 117200E.

CHEN, Y. *et al.* Ship detection in optical sensing images based on YOLOv5. In: Twelfth International Conference on Graphics and Image Processing (ICGIP), 2020, Xi'na, China. **Anais** […] Xi'an: International Society for Optics and Photonics. p. 117200E. doi: https://doi.org/10.1117/12.2589395.

CHENG, D. *et al.* Person re-identification by multi-channel parts-based cnn with improved triplet loss function. *In:* **IEEE conference on computer vision and pattern recognition**, 2016, Las Vegas, Nevada. **Anais** […] Las Vegas: IEEE, p. 1335-1344.

CHIYCHISAN, I. A new FGPA-based real-time configurable system for medical image processing. *In*: E-Health and Bioengineering Conference (EHB), 2013, Iasi, Romania. **Paper** […] p. 1-4, doi: 10.1109/EHB.2013.6707301.

CHULHEE, L.; LANDGREBE, D. A. Decision boundary feature extraction for neural networks. **IEEE Transactions on Neural Networks**. v. 8, n. 1, p. 75-83, 1997. doi: 10.1109/72.554193.

COELHO, M. Inteligência Artificial e Deep Learning. *In:* UFOP - Laboratório Imobilis Computação Móvel. **Departamento de Computação – DECOM**. Ouro Preto, 28 mar. 2019. Disponível em: http://www2.decom.ufop.br/imobilis/inteligencia-artificial-e-deep-learning/. Acesso em: 28 de abril de 2020.

COGSWELL, M. *et al.* Reducing sobreajuste in deep networks by decorrelating representations. **arXiv preprint arXiv:1511.06068**, p. 1-12, 2015.

Convolution Neural Network (CNN). **Developers Breach,** [s.d.]. Disponível em: https://developersbreach.com/convolution-neural-network-deep-learning/. Acesso em: 23 de abril de 2021.

ĆOROVIĆ, A. *et al.* The real-time detection of traffic participants using YOLO algorithm. *In:* **26th Telecommunications Forum (TELFOR)**, 2018, Belgrade, Serbia. **Paper** […] Belgrade: IEEE, 2018. p. 1-4. doi: 10.1109/TELFOR.2018.8611986.

CRUZ, C. F. **Automatic Analysis of Mammography Images:** enhancement and segmentation techniques. 2011. 152 f. Dissertação (Mestrado em Bioengenharia – Engenharia Biomédica) – Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2011.

DANIEL, C.; TAYLOR, J.; NOWOZIN, S. Learning Step Size Controllers for Robust Neural Network Training**. Proceedings of the AAAI Conference on Artificial Intelligence,** v. 30, n. 1, p. 1519-1525, 2016.

DERTAT, A. Applied deep learning-part 4: Convolutional neural networks. **Towards Data Science**, 2017. Disponível em: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Acesso em: 27 de maio de 2021.

DHAL, K. G. *et al.* Decolorization of Digital Pathology Images: A Comparative Study. *In:* 6th Student Computer Science Research Conference (StuCoSReC), 2019, Koper, Slovenia. **Paper** […] p. 49-52, doi: 10.26493/978-961-7055-82-5.49-52.

DHAMI, D. Object Detection/Localization Algorithms. **Medium,** 2018. Disponível em: https://medium.com/@dhartidhami/object-detection-localization-algorithms-e55b19259cbf. Acesso em: 05 de maio de 2021.

DIETTERICH, T. G.; KONG, E. B. **Machine learning bias, statistical bias, and statistical variance of decision tree algorithms**. 1995. 14 f. Relatório Técnico (Department of Computer Science), Oregon State University, Oregon, 1995.

DUBEY, A. K.; JAIN, V. Comparative study of convolution neural network's relu and leaky-relu activation functions. *In:* MISHRA, S.; SOOD, Y. S.; TOMAR, A. **Applications of Computing, Automation and Wireless Systems in Electrical Engineering**. v. 2. 1260 f. New Delhi: Springer, Singapore, 2019. p. 873-880.

DUMOULIN, V; VISIN, F. A guide to convolution arithmetic for deep learning. **arXiv preprint arXiv:1603.07285**, p. 1-31, 2016.

FACEBOOK ©. **Facebook AI.** [*S.l.:s.n*], Facebook, c2021. Disponível em: https://ai.facebook.com/. Acesso em: 18 de junho de 2021.

FANG, J; LIU, Q.; LI, J.. A Deployment Scheme of YOLOv5 with Inference Optimizations Based on the Triton Inference Server. *In:* IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), 2021, Chengdu, China. **Anais** […] Chengdu: IEEE, p. 441-445. doi: 10.1109/ICCCBDA51879.2021.9442557.

FANG, M.; HAUSLER, G. Class of transforms invariant under shift, rotation and scaling. **Applied Optics.** v. 29, n. 5, p. 704-708, 1990.

FERGUS, R.; PERONA, P.; ZISSERMAN, A. A Visual Category Filter for Google Images. In: European Conference on Computer Vision, 2004, Prague, Czech Republic. **Paper** […] Prague: Springer, Berlin, v. 3021, p. 242-256. doi: https://doi.org/10.1007/978-3-540-24670-1_19.

FERRARIS, V. *et al.* Robust Fusion of Multiband Images With Different Spatial and Spectral Resolutions for Change Detection. **IEEE Transactions on Computational Imaging**, v. 3, n. 2, p. 175-186, 2017.

FEURER, M.; HUTTER, F. Hyperparameter optimization. *In:* HUTTER, F.; KOTTHOFF, L. VANSCHOREN, K. **Automated machine learning:** Methods, Systems, Challenges. Berkeley: Springer, Cham, 2019. Cap 1, p. 3-33.

GAL, Y.; GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *In:* **Machine Learning Research - International Conference on Machine Learning**, 2016, New York, New York. **Anais** […] New York: PMLR, v. 48, p. 1050-1059, 2016.

GALOOGAHI, H. K.; SIM, T.; LUCEY, S. Multi-channel correlation filters. *In:* **IEEE international conference on computer vision**, 2013, Sidney, New South Waler. **Anais** […] Sidney: IEEE, p. 3072-3079.

GARBIN, C.; ZHU, X.; MARQUES, O. Dropout vs. batch normalization: an empirical study of their impact to deep learning. **Multimedia Tools and Applications**, v. 79, n. 19-20, p. 12777-12815, 2020.

GEDRAITE, E. S.; HADAD, M. Investigation on the effect of a Gaussian Blur in image filtering and segmentation. **Proceedings ELMAR-2011**. p. 393-396, 2011.

GOOGLE, INC. **Google Colaboratory**. [*S.l.*:*s.n*], GOOGLE, c2021. Disponível em: https://colab.research.google.com. Acesso em: 18 de junho de 2021.

GORGENS, E. B. *et al.* Estimação do volume de árvores utilizando redes neurais artificiais. **Revista Árvore**, v. 33, p. 1141-1147, 2009.

GOYAL, P. *et al.* Accurate, large minibatch sgd: Training imagenet in 1 hour. **arXiv preprint arXiv:1706.02677**, p. 1-12, 2017.

GRACE, K. *et al.* When will AI exceed human performance? Evidence from AI experts. **Journal of Artificial Intelligence Research**, v. 62, p. 729-754, 2018.

GRATTAROLA, D. **Deep Feature Extraction for Sample-Efficient Reinforcement Learning**. 2017. Dissertação (Mestrado em Ciência da Computação e Engenharia) – Politecnico di Milano, Milão, Itália, 2017.

HA, C. *et al.* Eliminating sobreajuste of probabilistic topic models on short and noisy text: The role of dropout. **International Journal of Approximate Reasoning**, v. 112, p. 85-104, 2019.

HAGAN, M. T.; DEMUTH, H. B. Neural networks for control. *In:* American control conference (cat. No. 99CH36251), 1999, San Diego, California, USA. **Paper** […] San Diego: IEEE, 1999. p. 1642-1656.

HAN, S. *et al.* Learning both weights and connections for efficient neural networks. **arXiv preprint arXiv:1506.02626**. p. 1-9, 2015.

HARRISON, J. Are roads safer with no central white lines? **BBC News**, United Kingdom, 3 fev. 2016. Disponível em: https://www.bbc.com/news/uk-35480736. Acesso em: 27 de maio de 2021.

HARZALLAH, H.; JURIE, F; SCHMID, C. Combining efficient object localization and image classification. *In:* **IEEE 12th international conference on computer vision**, 2009, Kyoto, Japan. **Anais** […] Kyoto: IEEE, p. 237-244.

HAYKIN, S. **Redes neurais: princípios e prática**. 2 ed. Ontário: Bookman Editora, 2007. 898 p.

HE, K.; SUN, J.; TANG, X. Guided Image Filtering. In: European conference on computer vision, 2010. **Paper** […] Crete, Greece: Springer, Berlin, Heidelberg, v. 6311, p. 1-14.

HE, W.; LI, B.; SONG, D. Decision boundary analysis of adversarial examples. *In:* International Conference on Learning Representations, 2018, Berkeley, California. **Paper** […] Berkeley: Computer Science Division, 2018. p. 1-15.

HINTONS, G.; SRIVASTAVA, N.; SWERSKY; K. Neural Networks for Machine Learning - Lecture 6a Overview of mini-batch gradientdescente. **Cited on**, v. 14, n. 8, p. 2-33, 2012.

HOSANG, J. *et al.* What makes for effective detection proposals?. **IEEE transactions on pattern analysis and machine intelligence**, v. 38, n. 4, p. 814-830, 2015.

HOSANG, J; BENENSON, R.; SCHIELE, B. Learning non-maximum suppression. In: **IEEE conference on computer vision and pattern recognition**, 2017, Honolulu, Hawaii. **Anais** […] Honolulu: IEEE, p. 4507-4515.

HOY, M. B. Alexa, Siri, Cortana, and more: an introduction to voice assistants. **Medical reference services quarterly**, v. 37, n. 1, p. 81-88, 2018.

HUANG, G. *et al.* Densely connected convolutional networks. *In:* **IEEE conference on computer vision and pattern** recognition, 2017, Venice, Italy. **Anais** […] Venice: IEEE, p. 4700-4708.

HUANG, G.-B. *et al.* Can threshold networks be trained directly?. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 53, n. 3, p. 187-191, 2006.

HUERTAS, A; NEVATIA, R. Detecting buildings in aerial images. **Computer vision, graphics, and image processing**, v. 41, n. 2, p. 131-152, 1988.

HURTIK, P. *et al.* Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3. **arXiv preprint arXiv:2005.13243**, p. 1-18, 2020.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *In:* **32nd - International Conference on Machine Learning**, 2015, Lille, France. **Anais** […] Lille: JMLR.org, p. 448-456.

IS THE MAZDA 6 one of the best built cars on the road? *In*: Articles. **Business First Online**, United Kingdom, 8 nov. 2018. Disponível em: https://www.businessfirstonline.co.uk/articles/is-the-mazda-6-one-of-the-best-built-cars-on-the-road/. Acesso em: 27 de maio de 2021.

JACOBS, R. A. Increased rates of convergence through learning rate adaptation. **Neural networks**, v. 1, n. 4, p. 295-307, 1988.

JEFFCOCK, P. What's the Difference Between AI, Machine Learning, and Deep Learning? jul. **Oracle Big Data Blog.** [*S.l*], 7 jul. 2018. Disponível: https://blogs.oracle.com/bigdata/post/whatx27s-the-difference-between-ai-machine-learning-and-deep-learning. Acesso em: 5 de maio de 2021.

JEONG, J. The Most Intuitive and Easiest Guide for Convolutional Neural Network. **Towards Data Science**, 2019. Disponível em: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480. Acesso em: 27 de maio de 2021

JESUS, E. O.; COSTA JR, R. A utilização de filtros gaussianos na análise de imagens digitais. **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**. v. 3, n. 1, p. 010118-1 – 010118-7, 2015.

JUDD, K.; SMITH, L. A.; WEISHEIMER, A. How good is an ensemble at capturing truth? Using bounding boxes for forecast evaluation. **Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography**, v. 133, n. 626, p. 1309-1325, 2007.

JUNG, H. Y.; LEE, K. M.; LEE, S. U. Toward global minimum through combined local minima. *In:* 10th European Conference on Computer Vision, 2008, Marseille, France. **Paper** […] Marseille: Springer, Berlin, Heidelberg, 2008. v. 5305, p. 298-311. doi: https://doi.org/10.1007/978-3-540-88693-8_22.

KADAM, S. Neural Networks Part 3: Understanding Back propagation & Learning Rates. **Analytics Vidhya,** 2020. Disponível em: https://medium.com/analytics-vidhya/neural-networks-part-3-understanding-back-propagation-learning-rates-3482a981a2f0. Acesso em: 05 de maio de 2021.

KAUFMAN, D. **A inteligência artificial irá suplantar a inteligência humana?** 1 ed. Barueri, SP: Estação das Letras e Cores, 2018. 94 p.

KHOR, S. H. The Gentlest introduction to Tensorflow – Part 2. *In:* Machine Learning, Data Science, Big Data, Analytics, AI – 2016 Aug Tutorials, Overviews – Kdnuggets. **KDnuggets™**. [*S.l*]: 2016. Disponível em: https://www.kdnuggets.com/2016/08/gentlest-introduction-tensorflow-part-2.html/2. Acesso em: 20 de maio de 2021.

KUMAR, S. K. On weight initialization in deep neural networks. **arXiv preprint arXiv:1704.08863.** p. 1-9, 2017.

KUMAR, T.; VERMA, K. A Theory Based on Conversion of RGB image to Gray image. **International Journal of Computer Applications.** v. 7, n. 2, p. 7-10, 2010.

LAMPERT, C. H.; BLASCHKO, M. B.; HOFMANN, T. Beyond sliding windows: Object localization by efficient subwindow Search. In: 2008 IEEE conference on computer vision and pattern recognition, 2008, Anchorage, Alaska, USA. **Paper** […] Anchorage: IEEE, 2008. p. 1-8.

LAWRENCE, S.; GILES, C. L. Sobreajuste and neural networks: conjugate gradient and backpropagation. *In:* IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN, 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 2000, Como, Italy. **Anais** […] Como: IEEE, 2000. p. 114-119. doi: 10.1109/IJCNN.2000.857823.

LESHNO, M. *et al.* Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. **Neural Networks.** v. 6, n. 6, p. 861-867, 1993.

LEVIN, A; LISCHINSKI, D; WEISS, Y. Colorization using optimization. *In:* ACM SIGGRAPH, 2004, **Paper** […] p. 689-694, doi: 10.1145/1186562.1015780.

LI, H. *et al.* A convolutional neural network cascade for face detection. *In:* **IEEE conference on Computer Vision and Pattern Recognition**, 2015, Boston Massachusetts. **Anais** […] Boston: IEEE, p. 5325-5334.

LI, M. *et al.* Efficient mini-batch training for stochastic optimization. *In:* **20th ACM SIGKDD international conference on Knowledge discovery and data mining**. 2014, New York, New York. **Anais** […] New York: Association for Computing Machinery, p. 661-670.

LI, X. *et al.* Understanding the disharmony between dropout and batch normalization by variance shift. *In:* **IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2019, Long Beach, California. **Anais** […] Long Beach: IEEE, p. 2682-2690.

LIN, T. *et al.* Don't use large mini-batches, use local SGD. **arXiv preprint arXiv:1808.07217**, p. 1-40, 2018.

LIN, T. -Y. *et al.* Focal loss for dense object detection. In: **IEEE international conference on computer vision**, 2017, Venice, Italy. **Anais** […] Venice: IEEE, p. 2980-2988.

LIU, K. *et al.* Performance Validation of Yolo Variants for Object Detection. International Conference on Bioinformatics and Intelligent Computing, 2021, New York, New York. **Anais** […] NY: Association for Computing Machinery, p. 239–243. doi: https://doi.org/10.1145/3448748.3448786.

LIU, S. *et al.* Path aggregation network for instance segmentation. *In:* IEEE conference on computer vision and pattern recognition, 2018, Salt Lake City, Utah. **Anais** […] Salt Lake City: IEEE, p. 8759-8768.

LIU, W. *et al.* Large-margin softmax loss for convolutional neural networks. *In:* **33rd International Conference on Machine Learning**, 2016, New York, New York. **Anais** […] New York: JMLR.org, v. 48, p. 7, 2016.

LOBO, L. C. Inteligência artificial, o Futuro da Medicina e a Educação Médica. **Revista Brasileira de Educação Médica**, v. 42, n. 3, p. 3-8, 2018.

LONG, Y. *et al.* Accurate object localization in remote sensing images based on convolutional neural networks. **IEEE Transactions on Geoscience and Remote Sensing**, v. 55, n. 5, p. 2486-2498, 2017.

LOWD, D.; DOMINGOS, P. Efficient weight learning for Markov logic networks. *In:* 11th European conference on principles of data mining and knowledge discovery, 2007, Warsaw, Poland. **Paper** […] Warsaw: Springer, Berlin, Heidelberg, 2007. p. 200-211. doi: https://doi.org/10.1007/978-3-540-74976-9_21.

LU, C.; XU, L. JIA; J. Contrast preserving decolorization, *In:* IEEE International Conference on Computational Photography (ICCP), 2012, Seatle, USA. **Paper** […] Seatle: IEEE, p. 1-7, doi: 10.1109/ICCPhot.2012.6215215.

MACKAY, D. J. C. Hyperparameters: optimize, or integrate out?. *In:* HEIDBREDER, G. R. **Maximum entropy and bayesian methods.** Santa Barbara, California: Springer, Dordrecht, 1996. Bayesian Hyperparameters, p. 43-59.

MATIĆ, V. *et al* . #029 CNN Yolo Algorithm. *In:* Deep Learning. **Master Dara Science**, 26 nov. 2018. Disponível em: http://datahacker.rs/object-detection-yolo-algorithm/. Acesso em: 5 de junho de 2021.

MCCARTHY, J. **What is Artificial Intelligence?** 2004. 14 f. Relatório – Departamento de Ciência da Computação, Universidade de Stanford. Stanford, Califórnia, 2004. Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1090.836&rep=rep1&type=pdf. Acesso em: 06 de maio de 2021.

MOUSAVIAN, A. *et al.* 3D bounding box estimation using deep learning and geometry. *In***: IEEE conference on Computer Vision and Pattern Recognition**, 2017, Honolulu, Hawaii. **Anais** […] Honolulu: IEEE, p. 7074-7082.

MUELLER, J. P.; MASSARON, L. **Machine Learning for dummies**. 2 ed. Hoboken, New Jersey: John Wiley & Sons, Inc, 2021, 464 p.

MÜLLER, B; REINHARDT, J; STRICKLAND, M T. **Neural networks: an introduction**. Durham: Springer Science & Business Media, 1995. 331 p.

NESTMEYER, T; GEHLER, P. V. Reflectance adaptive filtering improves intrinsic image estimation. *In:* **IEEE Conference on Computer Vision and Pattern Recognition**, 2017, Honolulu, Hawaii. **Anais** […] Honolulu: IEEE, p. 6789-6798.

NEUBECK, A; VAN GOOL, L. Efficient non-maximum suppression. In: **18th International Conference on Pattern Recognition (ICPR'06)**, 2006, Hong Kong, China. **Paper** […] Hong Kong: IEEE, p. 850-855. doi: 10.1109/ICPR.2006.479.

NG, A. . Exercise 8: Non-linear SVM classification with kernels. **Machine Learning**. Stantford, Stantford University, 2012. Disponível em: http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html. Acesso em: 25 de abril de 2021.

NÓBREGA, R. A. A. **Detecção da malha viária na periferia urbana de São Paulo utilizando imagens de alta resolução espacial e classificação orientada a objetos**. 2007. 166 f. Tese (Doutorado em Engenharia de Transportes). Universidade de São Paulo, São Paulo, 2007.

NVIDIA©. **NVIDIA® T4**. [*S.l:s:n*], NVIDIA, c2021. Disponível em: https://www.nvidia.com/pt-br/data-center/tesla-t4/. Acesso em 18 de junho de 2021.

OCTAVIAN, R. *et al*. Converting unstructures and semi-structured data into knowledge. *In:* 11th RoEduNet International Conference, 2013, Sinaia, Romania. **Paper** […] Sinaia: IEEE, 2013.

PACHECO, C. A. R; PEREIRA, N. S. Deep Learning: Conceitos e Utilização nas Diversas Áreas do Conhecimento. **Revista Ada Lovelace**, v.2, p. 34-49, 2018.

PADARIAN, J; MINASNY, Budiman; MCBRATNEY, Alex B. Machine learning and soil sciences: A review aided by machine learning tools. **Soil,** v. 6, n. 1, p. 35-52, 2020.

PAPAGEORGIOU, C. P.; OREN, M.; POGGIO, T. A general framework for object detection. *In:* Sixth International Conference on Computer Vision, 1998, Bombay, India (IEEE Cat. No. 98CH36271). **Paper** […] Bombay: IEEE, 1998. p. 555-562.

PASZKE, A. *et al.* Enet: A deep neural network architecture for real-time semantic segmentation. **arXiv preprint arXiv:1606.02147**. p. 1-10, 2016.

PEI, M. *et al.* Genetic Algorithms For Classification and Feature Extraction. *In:* Classification Society Conference, 1995. **Paper** […] 1995. p. 1-28.

PHYTHON AWESOME - **YOLO & RCNN Object Detection and Multi-Object Tracking**. Machine Learning. Disponível em: https://pythonawesome.com/yolo-rcnn-object-detection-and-multi-object-tracking/. Acesso em: 23 de abril de 2021.

PRECHELT, L. Early stopping-but when?. *In:* ORR, G. B.; MÜLLER, K. -R (Ed). *In:* MONTAVON, G. ORR, G. B.; MÜLLER, K. -R (Ed). **Neural Networks:** Tricks of the trade. 2 ed. Cornell: Springer, Berlin, Heidelberg, 2012. Regularization Techniques to Improve Generalization, p. 55-69.

QUEIROZ, J. E. R.; GOMES, H. M. Introdução ao Processamento Digital de Imagens. **RITA: instruções para preparação de documentos em Word**. v. 3, n. 1, p. 11-42, 2006.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. *In*: **IEEE conference on computer vision and pattern recognition**, 2016, Las Vegas, Nevada. **Anais** […] Las Vegas: IEEE, p. 779-788, 2016.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. *In:* IEEE conference on computer vision and pattern recognition, 2017, Honolulu, Hawaii. **Anais** […] Honolulu: IEEE, p. 7263-7271, 2017.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, p. 1-6, 2018.

REED, R.; MARKSII, R. J. **Neural smithing: supervised learning in feedforward artificial neural networks**. 1 ed. Massachusetts: Bradford Book, 1999. 360 p.

REINSEL, D.; GANTZ, J. RYDNIG, J. Data Age 2025: The Evolution of Daa to Life-Critical. **IDC, Seagate**. 2017. 25 p. Disponível em: https://www.import.io/wp-content/uploads/2017/04/Seagate-WP-DataAge2025-March-2017.pdf. Acesso em 5 de maio de 2021.

REZATOFIGHI, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. *In:* IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, Long Beach, California. **Anais** […] Long Beach: IEEE, p. 658-66.

RIEDMILLER, M,; LERNEN, A. M. Multi layer perceptron**. Machine Learning Lab Special Lecture**, University of Freiburg, p. 7-24, 2014.

ROBINSON, G. S. Edge detection by compass gradient masks. **Computer graphics and image processing**, v. 6, n. 5, p. 492-501, 1977.

ROBOFLOW. Roboflow**.** c2021. Disponível em: https://roboflow.com/. Acesso em: 6 de maio de 2021.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65, n. 6, p. 386, 1958.

RUDER, S. An overview of gradient descent optimization algorithms. **arXiv preprint arXiv:1609.04747**. p. 1-14 2016.

SALIMANS, T.; KINGMA, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. **Advances in neural information processing systems**, v. 29, p. 901-909, 2016.

SAMUEL, A. L. *Some Studies in Machine Learning Using the Game of Checkers*. **IBM Journal of Research and Development**, v.3, n. 3, p. 210-229, 1959.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, v. 3, n. 3, p. 210-229, 1959.

SANTURKAR, S. *et al.* How does batch normalization help optimization?. *In:* **32nd international conference on neural information processing systems**, 2018, Montréal, Canada. **Anais** […] Montréal: Curran Associates Inc., p. 2488-2498.

SCHMIDT-HIEBER, J. Nonparametric regression using deep neural networks with ReLU activation function. **The Annals of Statistics**, v. 48, n. 4, p. 1875-1897, 2020.

SCHÖLKOPF, B.; PLATT, J. HOFMANN, T.(Ed.). **Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference**. 1 ed. Massachusetts: Bradford Book, 2007, 1690 p.

SCHORFHEIDE, F. Loss function-based evaluation of DSGE models. **Applied Econometrics,** v. 15, n. 6, p. 645-670, 2000.

SCHWING, A. G.; URTASUN, R. Fully connected deep structured networks**. arXiv preprint arXiv:1503.02351**, p. 1-10, 2015.

SCURI, A. E. **Fundamentos da imagem digital**. Janeiro de 1999. 67 f. Notas de Aula Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1999.

SHAFIEE, M. J. *et al.* Fast YOLO: A fast you only look once system for real-time embedded object detection in video. **arXiv preprint arXiv:1709.05943**, p. 1-3, 2017.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding Machine Learning:** From Theory to Algorithms. 1 ed. New York, New York: Cambridge University Press, 2014. 409 p.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation Functions in Neural Networks. **International Journal of Engineering Applied Sciences and Technology**. v. 4, n. 12, p. 310-316, 2020.

SHASTRY, A. C. Convolutional Neural Networks: Part 2. **Towards Data Science**, 2018. Disponível em: https://towardsdatascience.com/convolutional-neural-networks-part-2-ce80bf2835af. Acesso em: 27 de maio de 2021.

SHEN, J; CASTAN, S. An optimal linear operator for step edge detection. **CVGIP: Graphical models and image processing**, v. 54, n. 2, p. 112-133, 1992.

SHRIVAKSHAN, G. T.; CHANDRASEKAR, C. A comparison of various edge detection techniques used in image processing. **International Journal of Computer Science Issues (IJCSI)**, v. 9, n. 5, p. 269, 2012.

SHURAN, S.; JIANXIONG, X. Sliding Shapes for 3D Object Detection in Depth Images. *In:* 13th International Conference on Computer Vision, 2014, Zurich, Switzerland. **Paper** […] Zurich: Springer, Cham, 2014. p. 634-651.

SHYNK, J. Frequency-domain and multirate adaptive filtering. **IEEE Signal processing magazine**, v. 9, n. 1, p. 14-37, 1992.

SILVA, G. C. **Detecção e contagem de plantas utilizando técnicas de inteligência artificial e machine learning.** 2017. 94 f. TCC (Graduação em Engenharia Eletrônica) – Universidade Federal de Santa Catarina, Florianópolis, SC, 2017.

SILVA, S. R.; SCHIMIDT, F. Redução de variáveis de entrada de redes neurais artificiais a partir de dados de análise de componentes principais na modelagem de oxigênio dissolvido. **Química Nova**, v. 39, p. 273-278, 2016.

SINGH, S. Leaky ReLU as an Activation Function in Neural Networks. **Deep Learning University,** c2021. Disponível em: https://deeplearninguniversity.com/leaky-relu-as-an-activation-function-in-neural-networks/. Acesso em: 1 junho de 2021.

SMITH, L. N. A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. **arXiv preprint arXiv:1803.09820**, p. 1-21, 2018.

SMITH, S. *et al.* Don't decay the learning rate, increase the batch size. **arXiv preprint arXiv:1711.00489**, p. 1-11, 2017.

SOUSA JÚNIOR, M. A. **Segmentação multi-níveis e multi-modelos para imagens de radar e ópticas.** 2005**.** 137 f. Tese (Doutorado em Sensoriamento Remoto) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2007.

SOUZA, G. S. A. **Aprendizado de máquina em aplicações de manejo florestal** . 2019. 51 f. Tese (Doutorado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa. 2019

SPINOSA, E J.; CARVALHO, A. P. L. F.; GAMA, J. Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. *In:* **ACM symposium on Applied computing**, 2007, Seoul, South Korea. **Anais** […] New York: Association for Computing Machinery p. 448-452.

SRIVASTAVA, N. *et al.* Dropout: a simple way to prevent neural networks from sobreajuste. **The journal of machine learning research**, v. 15, n. 1, p. 1929-1958, 2014.

STANFORD – Stanford University. Tutorial 1: Image Filtering. **Stanford Edu**. Califórnia, USA [s.d]. Disponível em: https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html. Acesso em: 20 de maio de 2021.

STARCK, J. -L; MURTAGH, F. D.; BIJAOUI, A. **Image processing and data analysis: the multiscale approach**. 1 ed. Cambridge: Cambridge University Press, 1998. 287 p.

STRENGERT, M; KRAUS, M; ERTL, T. Pyramid methods in GPU-based image processing. **Proceedings vision, modeling, and visualization**. p. 169-176, 2016.

SUDOWE, P; LEIBE, B. Efficient Use of Geometric Constraints for Sliding-Window Object Detection in Video. *In:* 8th International Conference on Computer Vision, 2011, Sophia Antipolis, France. **Paper** […] Sophia Antipolis: Springer, Berlim, 2011. p. 11-20.

SUGIYAMA, M; KRAULEDAT, M; MÜLLER, K. -R. Covariate shift adaptation by importance weighted cross validation. **Journal of Machine Learning Research**, v. 8, n. 5, p. 985-1005, 2007.

SURYANSH, S. Gradient Descent: All you need to know. **Hackernoon.** [*S.l*], 10 mar. 2018. Disponível em: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. Acesso em: 3 de junho de 2021.

SZEGEDY, C; TOSHEV, A; ERHAN, D. Deep neural networks for object detection. 2013.

TEIXEIRA, J. **O que é inteligência artificial.** 3 ed. [*S.l.*]: E-Galáxia, 2019. 62 p.

THUAN, Do. **Evolution of yolo algorithm and yolov5**: the state-of-the-art object detection algorithm. 2021. 61 f. Bachelor's Thesis (Information Technology) Oulu University of Applied Sciences, Oulu, Finland, 2021.

TIAN, Z. *et al.* Fcos: Fully convolutional one-stage object detection. In: IEEE/CVF international conference on computer vision, 2019, Seoul, South Korea. **Anais** […] Seoul: IEEE, p. 9627-9636.

TUMER, K.; GHOSH, J. Analysis of decision boundaries in linearly combined neural classifiers. **Pattern Recognition**. v. 29, n. 2, p. 341-348, 1996.

ULTRALYTICS ©. **YoloV5**. [*S.l.:s.n*], Ultralytics, c2021. Disponível em: https://ultralytics.com/yolov5. Acessp em: 18 de junho de 2021.

VALENZUELA, H. Volvo Polestar electric – The new boy on the road. **MINAPIM MAGAZINE**, Joinville, SC, 24 out. 2019. Disponível em: https://minapim.com/polestar-2-ev-o-novo-garoto-na-estrada-o-eletrico-da-volvo/. Acesso em: 27 de maio de 2021.

VIJAYAN, A. E.; JOHN, A.; SEN, D. Efficient implementation of 8-bit vedic multipliers for image processing application. *In:* International Conference on Contemporary Computing and Informatics (IC3I), 2014, Mysore, Índia. **Paper** […] Mysore: IEEE, p. 544-552, doi: 10.1109/IC3I.2014.7019675.

VISHWAKARMA, R.; VENNELAKANTI, R. CNN Model & Tuning for Global Road Damage Detection. *In:* IEEE International Conference on Big Data (Big Data), 2020, Atlanta, Georgia. **Anais** […] Atlanta: IEEE, p. 5609-5615. doi: 10.1109/BigData50022.2020.9377902.

VORBURGER, T. V. *et al.* Applications of cross-correlation functions. **WEAR**. v. 271, n. 3-4, p. 529-533, 2011.

WANG, S. -H. *et al.* Classification of Alzheimer's disease based on eight-layer convolutional neural network with leaky rectified linear unit and max amostragem. **Journal of medical systems**, v. 42, n. 5, p. 1-11, 2018.

WANG, X. *et al.* Regionlets for generic object detection. *In:* **IEEE international conference on computer vision,** 2013, Portland, Oregon. **Anais** […] Portland: IEEE, p. 17-24.

WANG, X. *et al.* SPB-YOLO: An Efficient Real-Time Detector For Unmanned Aerial Vehicle Images. *In:* International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2021,Jeju Island, South Korea. **Anais** […] Jeju Island: IEEE, p. 099-104. doi: 10.1109/ICAIIC51459.2021.9415214.

WEBER, E. U. From subjective probabilities to decision weights: The effect of asymmetric loss functions on the evaluation of uncertain outcomes and events. **Psychological Bulletin**, v. 115, n. 2, p. 228, 1994.

WEI, Q.; DOBIGEON, N.; TOURNERET, J-Y. Bayesian Fusion of Multi-Band Images. **IEEE Journal of Selected Topics in Signal Processing**. v. 9, n. 6, p. 1117-1127, 2015.

WEN, Y. *et al.* Flipout: Efficient pseudo-independent weight perturbations on mini-batches. **arXiv preprint arXiv:1803.04386**, p. 1-16, 2018.

WU, D. *et al.* Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. **Computers and Electronics in Agriculture**, v. 178, p. 105742, 2020.

WU, Z. *et al.* Recognizing instagram filtered images with feature de-stylization. **Proceedings of the AAAI Conference on Artificial Intelligence.** v. 34, n. 7, p. 12418-12425, 2010.

XIAO, H; RASUL, K; VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. **arXiv preprint arXiv:1708.07747**, 2017.

XU, B. *et al.* Empirical evaluation of rectified activations in convolutional network. **arXiv preprint arXiv:1505.00853**, p. 1-5, 2015.

XU, D.; ANGUELOV, D.; JAIN, A. Pointfusion: Deep sensor fusion for 3D bounding box estimation. *In:* **IEEE conference on computer vision and pattern recognition**, 2018, Salt Lake City, Utah. **Anais** […] Salt Lake City: IEEE, p. 244-253.

YANI, M. *et al.* Application of transfer learning using convolutional neural network method for early detection of terry's nail. **Journal of Physics: Conference Series**. v. 1201, p. 012052, 2019.

YAO, Y.; ROSASCO, L.; CAPONNETTO, A. On early stopping in gradient descent learning. **Constructive Approximation**, v. 26, n. 2, p. 289-315, 2007.

YOU, Y; GITMAN, I; GINSBURG, B. Large batch training of convolutional networks. **arXiv preprint arXiv:1708.03888**, p. 1-8, 2017.

YU, D. *et al.* Mixed amostragem for convolutional neural networks. *In:* International conference on rough sets and knowledge technology, 2014, Shanghai, China. **Paper** […] Shanghai: Springer, Cham, 2014. p. 364-375. doi: 10.1007/978-3-319-11740-9_34.

YU, J. *et al.* Unitbox: An advanced object detection network. *In:* **24th ACM international conference on Multimedia**, 2016, New York, New York. **Anais** […] New York: Association for Computing Machinery, p. 516-520.

ZANIOLO, L.; MARQUES, O. On the use of variable stride in convolutional neural networks. **Multimedia Tools and Applications**, v. 79, n. 19, p. 13581-13598, 2020.

ZEILER, M. D. Adadelta: an adaptive learning rate method. **arXiv preprint arXiv:1212.5701**, p. 1-6, 2012.

ZEILER, Matthew D.; FERGUS, Rob. Stochastic amostragem for regularization of deep convolutional neural networks. **arXiv preprint arXiv:1301.3557**, p. 1-9, 2013.

ZENG, X.; YEUNG, D. S. Sensitivity analysis of multilayer perceptron to input and weight perturbations. **IEEE Transactions on neural networks**, v. 12, n. 6, p. 1358-1366, 2001.

ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. *In:* European conference on computer vision, 2016, Amsterdam, The Netherlands. **Paper** […] Amsterdam: Springer, Cham, v. 9907, p. 649-666. doi: https://doi.org/10.1007/978-3-319-46487-9_40.

ZHANG, T.; YU, B. Boosting with early stopping: Convergence and consistency. **The Annals of Statistics**, v. 33, n. 4, p. 1538-1579, 2005.

ZHANG, Z.; SABUNCU, M. R. Generalized cross entropy loss for training deep neural networks with noisy labels. *In:* 32nd Conference on Neural Information Processing Systems (NeurIPS), Montréal, Canada. **Paper** […] Montréal: Curran Associates Inc., 2018.

ZHANG, Z; PENG, H.. Deeper and wider siamese networks for real-time visual tracking. *In:* **IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2019, Long Beach, California. **Anais** […] Long Beach: IEEE, p. 4591-4600.

ZHAO, K. *et al.* Optimizing the f-measure for threshold-free salient object detection. *In:* **IEEE/CVF International Conference on Computer Vision,** 2019, Seoul, South Korea. **Anais** […] Seoul: IEEE, p. 8849-8857.

ZHONG, Y. *et al.* Anchor box optimization for object detection. In: **IEEE/CVF Winter Conference on Applications of Computer Vision**, 2020, Snowmass village, Colorado. **Anais** […] Snowmass village: IEEE, p. 1286-1294.

ZIOU, D.; TABBONE, S. A. Edge detection techniques-an overview. **Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii**, v. 8, p. 537-559, 1998.