UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Giovanni Cimolin da Silva

DETECÇÃO E CONTAGEM DE PLANTAS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING

Florianópolis
2017

GIOVANNI CIMOLIN DA SILVA

DETECÇÃO E CONTAGEM DE PLANTAS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Eduardo Luiz Ortiz Batista.

Florianópolis, Janeiro de 2018

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

Silva, Giovanni Cimolin da
lva, Giovanni Cimolin da Silva DETECÇÃO E CONTAGEM DE PLANTAS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING / Giovanni Cimolin da Silva ; orientador, Eduardo Luiz Ortiz Batista, 2018.
94 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Visão computacional. 3. Inteligência artificial. 4. Silvicultura. I. Batista, Eduardo Luiz Ortiz. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Giovanni Cimolin da Silva

DETECÇÃO E CONTAGEM DE PLANTAS UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL E MACHINE LEARNING

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia Eletrônica, e aprovada em sua forma final pelo Programa de Graduação da Universidade Federal de Santa Catarina.

Florianópolis, 19 de Janeiro de 2018.

Eduardo Luiz Ortiz Batista
Orientador

Renata Coelho Borges
Profa. Dra.

Lucas Momm Bastos
Engenheiro Mecânico

Florianópolis
2018

Dedicado à todos aqueles que estiveram e estão próximos de mim,
fazendo esta vida valer cada vez mais a pena.

Agradecimentos

Agradeço à minha família por todo apoio que tive em minha trajetória. Ao meu pai Valtair e minha mãe Lúcia, por sempre me apoiarem e incentivarem a buscar educação. E também pelas horas de trabalho debaixo do sol que me tornaram quem sou hoje e me ensinaram a valorizar o trabalho e o esforço. À minha irmã Isabella, pelos momentos que passamos brincando, jogando e conversando. Também sou grato a todo o suporte e incentivo de toda a família Cimolin e da Família da Silva.

À minha namorada, sempre me apoiando, nos momentos bons e ruins, e que torna meu dia mais feliz.

Aos meus amigos da universidade Vini, Jorge, Guilherme, Raul, Neckel, Isabella, Erich, e muitos outros que, por todos esses anos de amizade e parceria.

Aos meus professores, por proporcionarem uma educação de qualidade e grande conhecimento. Em especial, ao professor Eduardo Batista, por toda a dedicação em me ajudar a realizar este trabalho e pelas boas conversas no laboratório.

Agradeço à empresa Horus Aeronaves, pela confiança ao executar este trabalho e por todo o conhecimento fornecido. Aos sócios e funcionários, pelas horas de bom humor e companheirismo que tivemos.

Agradeço também à meu primeiro chefe e mentor, Guilherme Tondello, por todo o conhecimento e desenvolvimento pessoal e profissional que me proporcionou, em conjunto com o pessoal da Creative Solutions: Vitor, Leonardo, Rodrigo e a Fabíola.

À Universidade Federal de Santa Catarina, pela estrutura e pela oportunidade de formação que possibilitam um excelente curso.

À todos que de alguma forma contribuíram para minha formação profissional.

“ Acredite em si próprio e chegará
um dia em que os outros não terão
outra escolha senão acreditar com você.
(Cynthia Kersey)

Resumo

Este trabalho tem como objetivo utilizar técnicas de aprendizado de máquina e visão computacional para uma aplicação em agricultura, mais especificamente na silvicultura, para detecção e contagem de indivíduos em uma plantação de eucaliptos. O objetivo de precisão deste trabalho é atingir uma contagem de 95% de todos os eucaliptos com uma taxa de erro de 5% em relação ao total de detecções. Foram utilizados os modelos mais precisos de redes neurais convolucionais atualmente existentes através da plataforma TensorFlow. O treinamento da rede foi realizado com um *dataset* obtido através do sobrevoo de uma região de plantação de eucaliptos com um VANT. Todos os dados foram catalogados e armazenados para posterior avaliação das redes encontradas sendo que na área de interesse, existem 7866 plantas de eucalipto em estado detectável pelo sistema. As redes foram treinadas em três cenários diferentes, um utilizando imagens RGB originais como entrada e nos outros dois cenários, foram utilizados como entrada das redes imagens com a aplicação dos índices de vegetação IFV e IVV. Os modelos escolhidos para o treinamento foram a SSD Inception V2, R-CNN Resnet 101 e a R-CNN Resnet Inception V2. Todos os melhores resultados foram obtidos utilizando as imagens originais em RGB, sendo a R-CNN Resnet 101 que obteve os melhores resultados em termos de precisão, contando 7471 dos 7866 indivíduos totais, resultando em uma taxa de contagem de 95% com apenas 452 resultados falsos (5%) com tempo de execução de 578 milissegundos por imagem. Em termos de velocidade, a SSD Inception V2 foi a que obteve melhores resultados, com a detecção de 7370 plantas (93, 7%) e 430 resultados falsos utilizando apenas 81 milissegundos por imagem. A rede R-CNN Resnet Inception V2, obteve 7468 detecções (95%) com 397 erros, porém com um tempo de execução de 1102 milissegundos por imagem. Todas as redes utilizando índices de vegetação forneceram resultados com menos de 80% de precisão e foram desconsideradas da análise de resultados, devido ao seu desempenho inferior. Foi utilizada a rede com os melhores resultados, a R-CNN Resnet 101, para criação de um software automatizado de detecção e contagem de plantas baseado em uma entrada de imagem. Por fim, este trabalho valida a utilização de redes neurais artificiais para extração de informações de imagens aéreas para inventoriamento de silvicultura e abre novas possibilidades para manter a inovação contante e rápida neste campo de conhecimento.

Palavras-chaves: inteligência artificial, visão computacional, TensorFlow, silvicultura, contagem de indivíduos, inventário ambiental.

Abstract

This work aims to use machine learning and computer vision techniques for an application in agriculture, more specifically in forestry, for the detection and counting of individuals in a eucalyptus plantation. The objective of this work is to reach a count of 95% of all eucalyptus trees with a 5% error rate in relation to the total number of detections. The most accurate models of convolutional neural networks currently available through the TensorFlow platform were used. Training of the network was performed with a data from an overflight of a eucalyptus plantation area with a UAV. All the data were cataloged for the evaluation of the networks found. In the area of interest, there are 7866 eucalyptus plants in a state detectable by the naked eye. The networks were trained in three different scenarios, one using original RGB images as input and in the other two scenarios, images using the GVI and VVI vegetation indices were used as inputs to the networks. The models chosen for the training were a SSD Inception V2, R-CNN Resnet 101 and R-CNN Resnet Inception V2. All the best results were obtained using the original RGB images, with R-CNN Resnet101 obtaining the best results in terms of accuracy, counting 7471 of the 7866 total individuals, resulting in a 95% counting rate with only 452 false results (5%) with a runtime of 578 milliseconds per image. In terms of speed, SSD Inception V2 was the one that obtained the best results, with the detection of 7370 plants (93.7%) and 430 false results using only 81 milliseconds per image. The Resnet Inception V2 R-CNN network obtained 7468 detections (95%) with 397 errors, but with an execution time of 1102 milliseconds per image. All results with vegetation indices provided results with less than 80% accuracy and were disregarded from the analysis due to their inferior performance. The network with the best results was utilized, R-CNN Resnet 101, for the creation of an automated plant detection and counting software based on an image input. Finally, this work validates the use of artificial neural networks for the extraction of information from aerial images for forestry inventories and opens new possibilities for keeping the innovation constant and fast in this field of knowledge.

Key-words: artificial intelligence, computer vision, TensorFlow, forestry, counting individuals, environmental inventory.

Lista de ilustrações

Figura 1 – Espectro eletromagnético. 28
Figura 2 – Resposta espectral de uma folha verde. 29
Figura 3 – Imagem original RGB. 31
Figura 4 – Índice IFV aplicado. 31
Figura 5 – Imagem original RGB. 32
Figura 6 – Índice IVV aplicado. 32
Figura 7 – Plantação de *Eucalyptus*. 33
Figura 8 – Estrutura de uma Rede Neural Convolucional. 35
Figura 9 – Estrutura básica da rede *Single Shot Detector*. 36
Figura 10 – Rede *Single Shot Detector*. 37
Figura 11 – Estrutura básica da rede *Regions with Convolutional Neural Networks*. 38
Figura 12 – Contraste entre duas idades diferentes em uma mesma cultura. 41
Figura 13 – Drone utilizado no mapeamento. 42
Figura 14 – Ortomosaico da área de voo. 43
Figura 15 – Interface do software de categorização das amostras 44
Figura 16 – Amostra inadequada 45
Figura 17 – Amostra adequada 45
Figura 18 – Imagem indicada na Tabela 1. 46
Figura 19 – Imagem original RGB. 48
Figura 20 – Índice IFV aplicado. 48
Figura 21 – Imagem original RGB. 49
Figura 22 – Índice IFV aplicado. 49
Figura 23 – Comparação entre precisão e velocidade de redes de detecção. 50
Figura 24 – *Interface* de monitoramento fornecida pelo TensorBoard. 53
Figura 25 – Imagem processada por uma rede. 55
Figura 26 – Imagem exemplo para avaliação. 57
Figura 27 – Evolução do erro total - SSD Inception V2 - RGB 59
Figura 28 – Evolução do *mAP* - SSD Inception V2 - RGB 59
Figura 29 – Detecção exemplo 1 - SSD Inception V2 - RGB 60
Figura 30 – Detecção exemplo 2 - SSD Inception V2 - RGB 60
Figura 31 – Evolução do erro total - R-CNN ResNet 101 - RGB 60
Figura 32 – Evolução do *mAP* - R-CNN ResNet 101 - RGB 60
Figura 33 – Detecção exemplo 1 - R-CNN ResNet 101 - RGB 61
Figura 34 – Detecção exemplo 2 - R-CNN ResNet 101 - RGB 61
Figura 35 – Evolução do erro total - R-CNN Inception ResNet v2 - RGB 61
Figura 36 – Evolução do *mAP* - R-CNN Inception ResNet v2 - RGB 61
Figura 37 – Detecção exemplo 1 - R-CNN Inception ResNet v2 - RGB 62

Figura 38 – Detecção exemplo 2 - R-CNN Inception ResNet v2 - RGB 62
Figura 39 – Evolução do erro total - SSD Inception v2 - IFV 63
Figura 40 – Evolução do *mAP* - SSD Inception v2 - IFV 63
Figura 41 – Detecção exemplo 1 - SSD Inception v2 - IFV. 63
Figura 42 – Detecção exemplo 2 - SSD Inception v2 - IFV. 63
Figura 43 – Evolução do erro total - R-CNN ResNet 101 - IFV 64
Figura 44 – Evolução do *mAP* - R-CNN ResNet 101 - IFV 64
Figura 45 – Detecção exemplo 1 - R-CNN ResNet 101 - IFV. 64
Figura 46 – Detecção exemplo 2 - R-CNN ResNet 101 - IFV. 64
Figura 47 – Evolução do erro total - R-CNN ResNet Inception V2 - IFV 65
Figura 48 – Evolução do *mAP* - R-CNN ResNet Inception V2 - IFV 65
Figura 49 – Detecção exemplo 1 - R-CNN ResNet Inception V2 - IFV. 66
Figura 50 – Detecção exemplo 2 - R-CNN ResNet Inception V2 - IFV. 66
Figura 51 – Evolução do erro total - SSD Inception v2 - IVV 66
Figura 52 – Evolução do *mAP* - SSD Inception v2 - IVV 66
Figura 53 – Detecção exemplo 1 - SSD Inception v2 - IVV. 67
Figura 54 – Detecção exemplo 2 - SSD Inception v2 - IVV. 67
Figura 55 – Evolução do erro total - R-CNN ResNet 101 - IVV 68
Figura 56 – Evolução do *mAP* - R-CNN ResNet 101 - IVV 68
Figura 57 – Detecção exemplo 1 - R-CNN ResNet 101 - IVV. 68
Figura 58 – Detecção exemplo 2 - R-CNN ResNet 101 - IVV. 68
Figura 59 – Evolução do erro total - R-CNN ResNet Inception V2 - IVV 69
Figura 60 – Evolução do *mAP* - R-CNN ResNet Inception V2 - IVV 69
Figura 61 – Detecção exemplo 1 - R-CNN ResNet Inception V2 - IVV. 69
Figura 62 – Detecção exemplo 2 - R-CNN ResNet Inception V2 - IVV. 69

Lista de tabelas

Tabela 1 – Trecho do dataset 46
Tabela 2 – Comparação de desempenho das redes escolhidas no dataset COCO . . 51
Tabela 3 – Valores exemplo de avaliação. 57
Tabela 4 – Desempenho da rede SSD Inception V2 utilizando imagens RGB 60
Tabela 5 – Desempenho da rede R-CNN Resnet 101 utilizando imagens RGB . . . 61
Tabela 6 – Desempenho da rede R-CNN Inception ResNet v2 utilizando imagens RGB 62
Tabela 7 – Desempenho da rede SSD Inception v2 utilizando imagens com índice IFV. 63
Tabela 8 – Desempenho da rede R-CNN ResNet 101 utilizando imagens com índice IFV. 65
Tabela 9 – Desempenho da rede R-CNN ResNet Inception V2 utilizando imagens com índice IFV. 66
Tabela 10 – Desempenho da rede SSD Inception v2 utilizando imagens com índice IVV. 67
Tabela 11 – Desempenho da rede R-CNN ResNet 101 utilizando imagens com índice IVV. 68
Tabela 12 – Desempenho da rede R-CNN ResNet Inception V2 utilizando imagens com índice IVV. 70
Tabela 13 – Comparação de taxas de erro e acerto entre as redes 70

Sumário

1 INTRODUÇÃO 25
1.1 Objetivos gerais 25
1.2 Objetivos específicos 26
1.3 Organização do trabalho 26

2 REVISÃO TEÓRICA 27
2.1 Sensoriamento remoto no estudo da vegetação 27
2.2 A vegetação e sua interação com o espectro eletromagnético 28
2.2.1 Índices espectrais 29
2.2.1.1 Índice de folha verde (IFV) 30
2.2.1.2 Índice de vegetação visível (IVV) 31
2.2.2 Silvicultura para produção de madeira com *Eucaliptus* 32
2.3 Visão computacional 33
2.4 Inteligência Artificial 34
2.4.1 Aprendizado de máquina 34
2.4.2 Redes neurais convolucionais 35
2.4.2.1 *Single Shot Detectors* 36
2.4.2.2 *Regions with Convolutional Neural Networks* 37

3 METODOLOGIA DE PESQUISA 41
3.1 Obtenção das imagens 42
3.2 Criação de *dataset* 43
3.2.1 Redução do *dataset* de processamento 45
3.2.2 Aplicação de índices de vegetação 47
3.3 *Software* de processamento 49
3.4 Escolha das redes 50
3.5 Configuração das redes 51
3.6 Treinamento das redes 51
3.7 Avaliação de desempenho 53
3.7.1 *Mean Average Precision* (*mAP*) 53
3.8 Contagem de indivíduos 56

4 AVALIAÇÃO DAS REDES E ANÁLISE DE RESULTADOS 59
4.1 Resultados 59
4.1.1 Imagens RGB 59
4.1.1.1 SSD Inception V2 59
4.1.1.2 R-CNN ResNet 101 60

4.1.1.3 R-CNN Inception ResNet v2 . . . 61
4.1.2 Índice IFV . . . 62
4.1.2.1 SSD Inception v2 . . . 62
4.1.2.2 R-CNN ResNet 101 . . . 64
4.1.2.3 R-CNN Inception ResNet v2 . . . 65
4.1.3 Com índice VVI . . . 66
4.1.3.1 SSD Inception v2 . . . 66
4.1.3.2 R-CNN ResNet 101 . . . 67
4.1.3.3 R-CNN Inception ResNet v2 . . . 69
4.2 Comparação . . . 70

5 CONCLUSÃO . . . 71

6 PONTOS À MELHORAR E FUTUROS TRABALHOS . . . 73
6.1 Melhorar pré e pós-processamento dos resultados . . . 73
6.2 Realizar estudo com câmeras multi-espectrais . . . 73
6.3 Diversificar as áreas de estudo com culturas diversas . . . 73
6.4 Construir um modelo de detecção específico . . . 73

REFERÊNCIAS . . . 75

ANEXOS 79

ANEXO A – SSD INCEPTION V2 . . . 81

ANEXO B – FASTER R-CNN RESNET 101 . . . 87

ANEXO C – FASTER R-CNN INCEPTION RESNET V2 . . . 91

1 Introdução

Com o desenvolvimento de novas tecnologias, surgem a cada dia novas oportunidades de aumentar a produtividade do trabalho e gestão inteligente de recursos. Nos últimos anos, um dos setores que obteve grandes vantagens com inovações tecnológicas foi o setor agrícola, principalmente com novas tecnologias de monitoramento e análise. Um exemplo disso é o uso de Veículos Aéreos Não Tripulados (VANTs) para o monitoramento e análise do desenvolvimento das culturas e pragas em um determinado espaço.

Essa inovação proporciona diversas vantagens em relação a antigos métodos de monitoramento de culturas, que eram baseados em amostragem através de caminhadas no campo, que toma um tempo considerável, ou imagens de satélite, que têm uma cadência relativamente grande, dependendo da cultura. Esses métodos implicam em um intervalo de tempo entre a coleta de dados e a tomada de decisão que podem comprometer os resultados do cultivo. O monitoramento com VANTs fornece um grande volume de dados, e com baixa cadência, possibilitando agir rapidamente em possíveis problemas que venham a surgir em uma plantação.

Com esta modernização e consequente aumento no volume de dados, tornou-se necessário o uso de uma inteligência de gestão capaz de processar esses dados e transformá-los em informações úteis e fáceis de lidar. Existem diversas formas de realizar processamento de grandes volumes de dados atualmente, porém, umas das áreas que mais tem avançado nos últimos anos é a inteligência artificial, principalmente na parte de aprendizado de máquina.

Os avanços nessa áreas, principalmente devido à criação de novos algoritmos de treinamento de redes profundas e *hardware* mais potente, possibilitaram o estudo desenvolvimento de redes inteligentes extremamente complexas, capazes de realizar tarefas cognitivas de ato nível. Tal evolução abriu um leque de possibilidades de aplicações que hoje já estão ao nosso alcance: carros autônomos, assistentes virtuais e classificadores automáticos de imagens.

Visando conectar essas vertentes de tecnologia atual, este trabalho é focado na utilização de redes de inteligência artificial complexas para classificar os dados obtidos através de *VANT's* em aplicações na agricultura e transformá-los em poderosas informações de análise.

1.1 Objetivos gerais

O presente trabalho tem como objetivo a avaliação e aplicação de modelos modernos de inteligência artificial para analisar dados de sensoriamento remoto de uma cultura agrícola e fornecer informações para uma análise agrícola de forma automatizada.

1.2 Objetivos específicos

- Criar um dataset de treinamento para detecção de árvores de uma cultura de eucaliptos;
- Escolher, treinar e avaliar três diferentes modelos de detecção para a aplicação detecção em condições normais e com aplicação de índices de vegetação;
- Analisar a viabilidade de utilização de redes neurais complexas para detecção e contagem de indivíduos em plantações de silvicultura;

1.3 Organização do trabalho

Este trabalho está dividido em quatro partes, começando por uma revisão teórica da literatura de forma a analisar o problema, verificar a viabilidade de execução fundamentar o desenvolvimento do mesmo. Depois, são explicadas as metodologias de desenvolvimento do trabalho, considerando todos os aspectos, desde a criação de uma base de dados até as metodologias de avaliação utilizadas. Em seguida, são analisados os resultados obtidos e comparados em diversos cenários, de forma a chegar a uma conclusão sobre os métodos utilizados. Por fim, são realizadas conclusões sobre os resultados obtidos, discutidas possíveis melhorias na metodologia utilizada e trabalhos futuros que serão realizados sobre o tema.

2 Revisão teórica

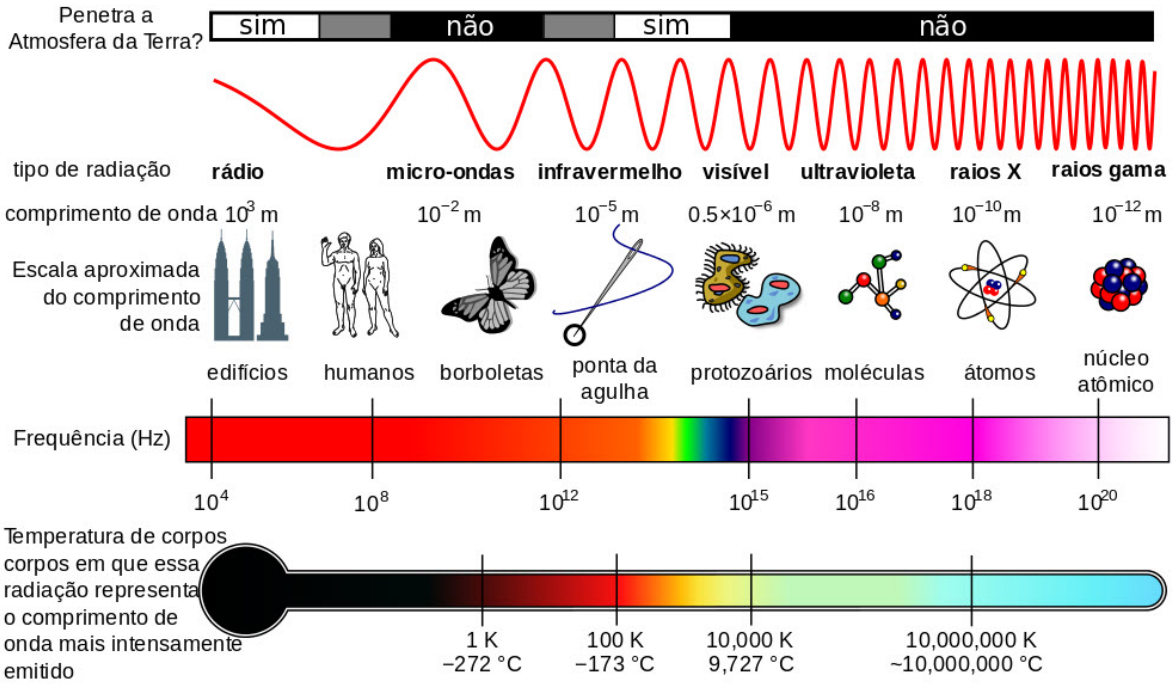
Neste trabalho foram utilizados conhecimentos das áreas de visão computacional e inteligencia artificial com aplicação na extração de informações obtidas através de sensoriamento remoto. Nas seções seguintes, serão descritas e exemplificadas todas as tecnologias e conhecimentos utilizados. Começando pelos métodos de obtenção de dados e análises através do sensoriamento remoto, seguindo para técnicas utilizadas para extrair informação. Depois são explicadas as tecnologias utilizadas para a extração automática de informação e processamento de dados, que é a visão computacional. Por fim, são explicadas as técnicas e métodos de inteligência artificial utilizados para extração das informações.

2.1 Sensoriamento remoto no estudo da vegetação

De acordo com Ponzoni, Shimabukuro e Kuplich (2007), o sensoriamento remoto é o estudo das propriedades físicas e químicas de alvos localizados na superfície terrestre, sem a necessidade de contato físico, baseando-se somente na interação desses alvos com a radiação eletromagnética. Pode-se considerar que a detecção das interações entre a radiação eletromagnética e os alvos da superfície terrestre é o princípio básico de toda tecnologia do sensoriamento remoto (PONZONI; SHIMABUKURO; KUPLICH, 2007).

Essa interação é medida através de sensores espectrais, que são equipamentos capazes de medir o espectro eletromagnético refletido. O espectro eletromagnético é uma divisão da distribuição da radiação pelas distintas regiões, organizado de acordo com o comprimento de onda e frequência de onda (ABREU, 2014). A Figura 1 mostra a representação do espectro eletromagnético, seus comprimentos de onda e temperatura de emissão de luz.

Figura 1 – Espectro eletromagnético.



Fonte: Wikipédia (2017).

O fluxo de radiação eletromagnética (REM) que atinge os objetos sofrem alterações que variam de acordo com a relação verificada entre o tipo de REM e a capacidade dos alvos para absorver, transmitir ou refletir a energia. Esses processos funcionam em diferentes proporções para cada tipo de material-alvo (MOREIRA, 2005) e podem ser definidos como:

a) **Absortância**: característica que indica o nível de absorção da REM pelo alvo.
b) **Transmitância**: propriedade do alvo que permite que a energia passe pelo mesmo sem ser absorvida.
c) **Reflectância**: comportamento característico da reflexão e espalhamento da REM.

A medida geralmente utilizada por esses sensores é a reflectância, ou indicação da quantidade de luz refletida pelos alvos. As imagens captadas podem ser interpretadas à partir de seu comportamento espectral em cada faixa medida do espectro. As faixas espectrais mais comuns são as visíveis, seguidas pelas infravermelho e termais.

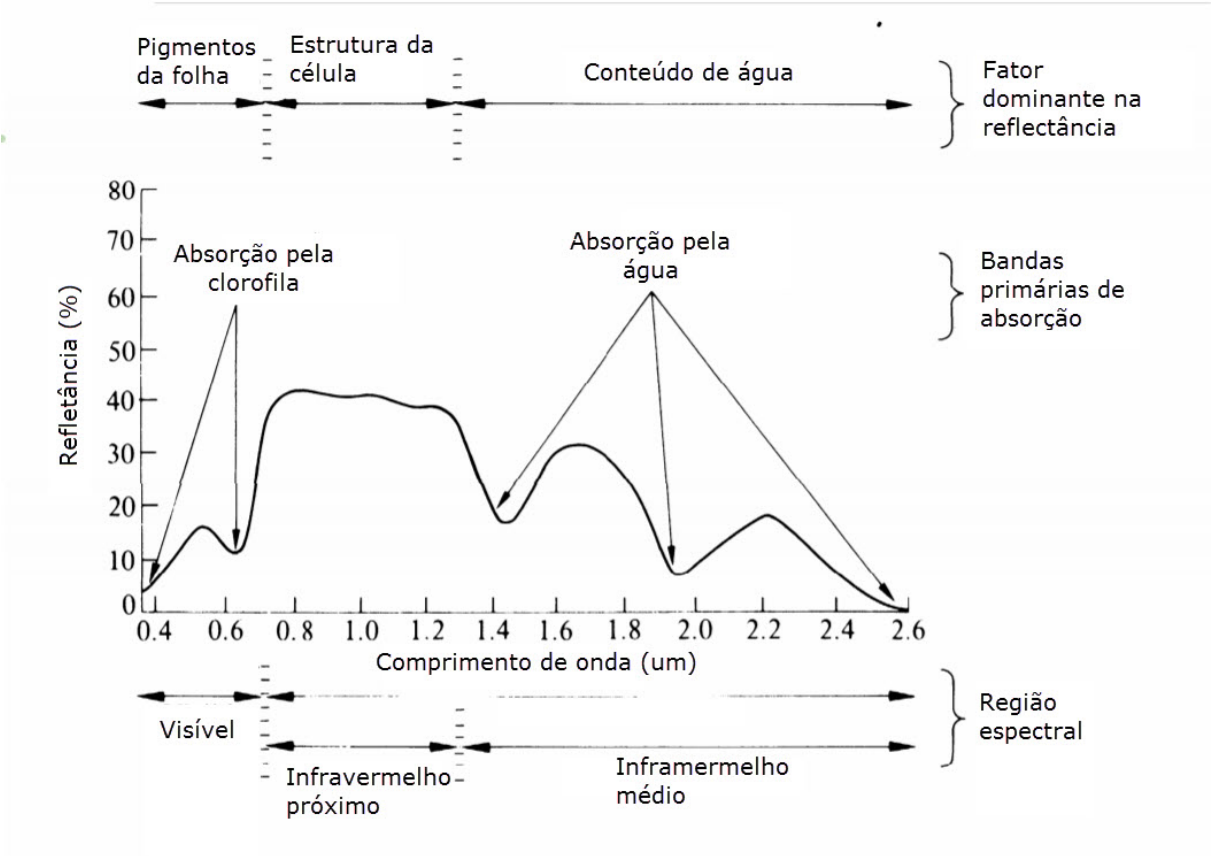
2.2 A vegetação e sua interação com o espectro eletromagnético

Todas as plantas com folhas verdes possuem um comportamento espectral característico, que é resultado de sua forma e estrutura interna. Em 1918 foi desenvolvida uma teoria que explica o comportamento espectral da radiação eletromagnética no interior da

folha verde e sadia, válida até o presente momento (MENESES; NETTO, 2001).

Uma análise da curva de reflectância de uma folha verde sadia apresenta suas principais características nas faixas visível (400nm a 750nm), infravermelho próximo (750nm a 900nm) e infravermelho médio (1550nm a 1750nm) (ANTUNES, 2001). A Figura 2 ilustra a resposta espectral de uma folha verde e seus principais fatores influenciadores.

Figura 2 – Resposta espectral de uma folha verde.



Fonte: Adaptado de Figueiredo (2005).

Porém, imagens de sensoriamento remoto raramente são de folhas individuais, e sim dos dosséis das árvores, que, devido ao ângulo de incidência e a transmitância das folhas, alteram seus valores de resposta espectral. Embora a resposta relativa entre as bandas seja mantida, outros fatores, como o ângulo solar e o tipo do solo, podem afetar os resultados finais de processamento de forma complexa.

2.2.1 Índices espectrais

Índices espectrais são combinações de níveis de reflectância de uma ou mais bandas de radiação que indicam relativamente a abundância de determinadas características de interesse. Índices de vegetação são o tipo mais popular pra aplicações agrícolas, mas também existem índices para áreas de queimadas, detecção de água e caracterização geológica.

Segundo Jackson e Huete (1991), em aplicações focadas para agricultura, índices espectrais são métodos importantes de extrair informação de dados de sensoriamento remoto, pois reduzem, mas não eliminam efeitos dos solos, topografia e ângulo de visão da análise realizada. A maior parte dos índices hoje são calculados utilizando razões ou diferenças normalizadas entre duas ou três bandas de cor (JACKSON; HUETE, 1991).

Existem dezenas de tipos de índices de vegetação disponíveis, e eles utilizam desde duas bandas de cor visível até bandas infra-vermelhas que requerem a utilização de sensores multi-espectrais para captação das imagens.

Para este trabalho, foram escolhidos para avaliação além da imagem RGB, dois índices de vegetação, sendo eles o Índice de Folha Verde (IFV) e o Índice de Vegetação Visível (IVV). O IFV foi escolhido pois é um índice muito utilizado na agricultura, principalmente para indicar saúde da vegetação utilizando sensores comuns e o IVV é um índice em desenvolvimento criado com a finalidade de ressaltar vegetação visível em imagens RGB comuns.

2.2.1.1 Índice de folha verde (IFV)

O Índice de Folha Verde foi desenvolvido por Louhaichi, Borman e Johnson (2001) para servir inicialmente como uma fórmula simples para diagnóstico e verificação do nível e tipo de queimadura do trigo utilizando apenas técnicas de sensoriamento remoto com sensores RGB (de espectro visível).

> Observamos que os *pixels* nas imagens RGB digitais das folhas e hastes da planta tinham valores verdes mais altos que o vermelho ou o azul enquanto o solo, as rochas, o lixo e as folhas mortas tendem a ter valores mais baixos para o verde do que para o vermelho ou o azul. Isto é esperado uma vez que a clorofila absorve a luz vermelha (centrada em cerca de 0,67 um) e a azul (centrada em cerca de 0,45 um) e reflete verde (centrada em cerca de 0,55 um). Portanto, classificamos as imagens determinando se a média dos números digitais vermelho e azul era maior ou menor que o número digital verde. (LOUHAICHI; BORMAN; JOHNSON, 2001)

Com o estudo da resposta espectral da vegetação, foi desenvolvido o Índice de Folha Verde (IFV), o qual é calculado da seguinte maneira:

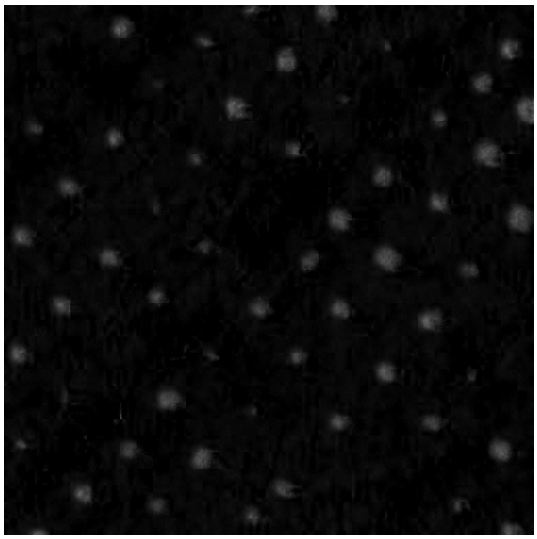
$$IFV = \frac{2 \times G - R - B}{2 \times G + R + B} \tag{2.1}$$

com R, G e B representando os valores de cores de cada pixel (vermelho, verde e azul, respectivamente). O IFV tem como resultado um valor que varia de -1 a 1 e serve para indicar qual a cobertura relativa de folhas verdes. Valores negativos tendem a indicar presença de solo ou indivíduos não vivos enquanto valores positivos indicam folhas e caules. Uma representação visual do IFV é mostrada na Figura 4 ao lado da imagem RGB original mostrada na Figura 3.

Figura 3 – Imagem original RGB.

Fonte: Produzido pelo autor.

Figura 4 – Índice IFV aplicado.



Fonte: Produzido pelo autor.

O IFV foi um índice também estudado por Gobron et al. (2000), o qual comprovou a efetividade em correlacionar a reflectância do espectro visível com a quantidade de clorofila presente na folha.

2.2.1.2 Índice de vegetação visível (IVV)

O Índice de Vegetação Visível fornece uma medida da quantidade de vegetação ou verde de uma imagem usando apenas informações do espectro visível. Sabe-se que é necessária a informação no infravermelho próximo para distinguir e separar a vegetação do solo ou superfícies de gelo em imagens de satélites. Porém, o IVV é um procedimento alternativo que tenta utilizar apenas informações no espectro visível, medindo a quantidade de verde em uma região usando índices de similaridade (Planetary Habitability Laboratory - University of Puerto Rico at Arecibo, 2017).

O IVV é como

$$IVV = \left[\left(1 - \left|\frac{R - R_0}{R - R_0}\right|\right) \times \left(1 - \left|\frac{G - G_0}{G + G_0}\right|\right) \times \left(1 - \left|\frac{B - B_0}{B - B_0}\right|\right)\right]^{1/w} \tag{2.2}$$

onde R, G e B são os componentes vermelho, verde e azul da imagem, respectivamente, RGB_0 é vetor da cor verde de referência e w é um expoente de peso usado para ajustar a sensibilidade da escala.

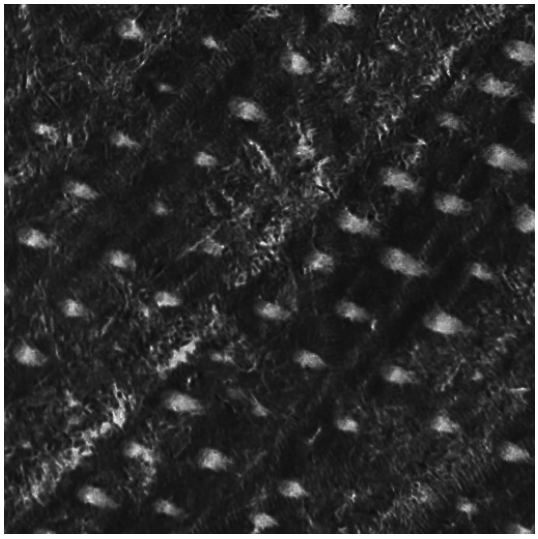
De acordo com testes realizados pelo Laboratório Planetário de Habitabilidade da Universidade de Porto Rico, os melhores resultados para imagens de 24 bits, ou 256 cores por canal, foram encontrados utilizando $RGB_0 = [30, 50, 0]$ e $w = 1$. Para evitar possíveis divisões por zero na equação, é necessário adicionar uma pequena quantidade para ambos os canais de RGB e de RGB_0, na qual o valor 10 funcionou melhor. A aplicação de tais valores no índice IVV é ilustrada na Figura 6 com uma representação da imagem de origem

em RGB na Figura 5.

Figura 5 – Imagem original RGB.

Fonte: Produzido pelo autor.

Figura 6 – Índice IVV aplicado.



Fonte: Produzido pelo autor.

2.2.2 Silvicultura para produção de madeira com *Eucaliptus*

Segundo Ford-Robertson (1971), silvicultura é a ciência de cultivar os bosques e seus possíveis produtos, com base no conhecimento da história de vida e as características gerais das árvores e especialmente do local. O sistema silvicultural é um processo que segue os princípios silviculturais aceitos, durante o qual se cultivam, extraem e renovem os produtos florestais de um bosque (FORD-ROBERTSON, 1971).

Com os avanços tecnológicos das últimas décadas, a agricultura de precisão, que baseia-se na coleta e análise de dados geoespaciais localizadas nas florestas, com a exatidão e a precisão (VETTORAZZI; FERRAZ, 2000), têm sido cada vez mais aplicada à silvicultura. A incorporação cada vez mais forte de tecnologias como sistemas de informações geográficas, sensoriamento remoto e sistemas de posicionamento global auxiliam e permitem intervenções cada vez mais precisas no povoamento das culturas (BRANDELERO; ANTUNES; GIOTTO, 2007).

Segundo Lima et al. (2010), o gênero *Eucalyptus* é a essência florestal que mais atende às necessidades de reposição de matéria-prima para a fabricação de papel no Brasil, com crescimento médio de $45m^3$ por hectare por ano de madeira. Dentre as diferentes espécies promissoras na formação de povoamentos plantados, esse gênero compreende um grande número de espécies e, consequentemente, uma alta variabilidade no que diz respeito às características físicas, mecânicas e estéticas, possibilitando a substituição de várias espécies nativas (MARCHIÜRI; SOBRAL, 1997). Um exemplo de Silvicultura Intensiva Clonal de *Eucalyptus* é mostrado na Figura 7.

Figura 7 – Plantação de *Eucalyptus*.

Fonte: CIFlorestas (2011).

2.3 Visão computacional

Visão computacional (CV, do inglês) é um campo multidisciplinar que tem como objetivo pesquisar a capacidade de computadores de ganhar um nível humano de entendimento de imagens digitais ou vídeo (BALLARD; BROWN, 1982).

Do ponto de vista das ciências biológicas, a visão computacional tem como objetivo apresentar modelos computacionais semelhantes ao do sistema visual humano. Já do ponto de vista da engenharia, procura construir sistemas autônomos que possam desempenhar algumas das tarefas que o sistema visual humano pode executar (e até mesmo superá-lo em muitos casos) (HUANG, 1996).

Aplicações desta tecnologia variam desde inspeções de qualidade em linhas de produção até a compreensão de situações humanas complexas em imagens e vídeo (BALLARD; BROWN, 1982). Existem quatro grandes áreas de visão computacional, são elas:

a) **Reconhecimento**: é a detecção, classificação e identificação de objetos em uma imagem através de algoritmos.

b) **Análise de movimento**: é o rastreamento e cálculo da velocidade de um ou mais objetos com relação à câmera da imagem de entrada.
c) **Reconstrução 3D**: utilizando algoritmos de reconstrução e modelagem, são gerados modelos tridimensionais de objetos físicos a partir de imagens de múltiplos ângulos e direções.
d) **Restauração de imagens**: tem como objetivo melhorar a qualidade de imagens através de técnicas de remoção de ruído e processamento avançado de padrões.

A principal área de estudo utilizada neste trabalho é a de reconhecimento, que consiste na identificação de padrões, objetos e cores a fim de extrair informação inteligível de uma imagem. Atualmente, os algoritmos de reconhecimento mais eficientes são as redes neurais convolucionais, um subcampo da área de inteligência artificial.

2.4 Inteligência Artificial

A definição básica de inteligência é a capacidade de um indivíduo de compreender o mundo objetivo e aplicar conhecimento para resolver problemas. Consiste em suas capacidades de perceber, entender e analisar informações para uma tomada de decisão, além de aprender com decisões passadas.

Segundo Akerkar (2014), a Inteligência Artificial (IA) é um campo de estudo que procura analisar e desenvolver sistemas capazes de desempenhar funções que, caso um ser humano fosse executar, seriam consideradas inteligentes. De acordo com Ertel (2011), a IA é um conceito amplo, do qual podemos descrever algumas características básicas que a compõem, como:

a) **Capacidade de raciocínio:** aplicar regras lógicas a um conjunto de dados disponíveis para chegar à uma conclusão.
b) **Aprendizagem:** aprender com os erros e acertos para maximizar os futuros resultados.
c) **Reconhecer padrões:** detectar e classificar objetos, texturas e padrões de comportamento.
d) **Inferência:** capacidade de aplicar raciocínio em situações cotidianas.

2.4.1 Aprendizado de máquina

É o subcampo de estudo da Inteligência Artificial que pesquisa e desenvolve algoritmos que possibilitam o computador aprender a resolver problemas sem ser explicitamente programado para isso.

Segundo Russell, Norvig e Intelligence (1995), as tarefas de treinamento de aprendizado de máquina são divididos em três campos, de acordo com o tipo de feedback utilizado no treinamento. São eles:

a) **Aprendizado supervisionado:** são fornecidas entradas e saídas exemplos para

o computador através de um "professor"humano, com o objetivo de ensiná-lo a mapear uma regra geral. Esta é a metodologia utilizada neste trabalho.

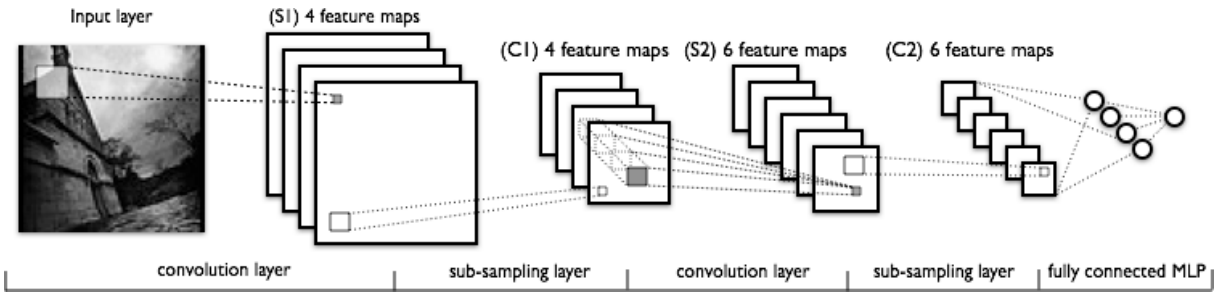
b) **Aprendizado não supervisionado:** não são fornecidas saídas mapeadas e o objetivo da rede é procurar e identificar padrões ou grupos, sendo deixada para aprender por si só.

c) **Aprendizado por reforço:** o programa do computador interage com o ambiente de forma dinâmica e recebe um *feedback* baseado em uma função de avaliação para ajustar seus pesos e ir melhorando a cada interação. Um exemplo deste tipo de aprendizado é o treinamento de uma rede para jogar um jogo contra um adversário, no qual sua pontuação final é um *feedback* para ajuste e avaliação dos pesos.

2.4.2 Redes neurais convolucionais

Uma rede neural convolucional (CNN, do inglês *Convolutional Neural Network*) é uma classe de rede neurais artificiais do tipo *feed-forward* que vem sendo aplicada com sucesso no processamento e análise de imagens digitais. De acordo com LeCun et al. (2015), as CNN's são redes multicamadas projetadas para o reconhecimento de padrões diretamente dos *pixels* sem a necessidade de um pré-processamento complexo. Isso significa que a rede "aprende"os filtros que em um algoritmo tradicional precisariam ser implementados manualmente. Elas conseguem reconhecer padrões extremamente complexos e são adaptáveis a distorções e transformações geométricas. É exatamente essa independência de um conhecimento específico e do esforço humano no desenvolvimento de suas funcionalidades básicas a maior vantagem de sua aplicação. A Figura 8 ilustra um modelo de alto nível de uma CNN.

Figura 8 – Estrutura de uma Rede Neural Convolucional.



Fonte: Adaptado de LISA Lab. (2017).

Existem muitos tipos de CNN's, cada uma com características que a deixam mais específicas para determinadas aplicações. Neste trabalho, foram utilizadas duas estruturas de rede, as *Single Shot Detectors* e as *Regions with Convolutional Neural Networks*, descritas em maiores detalhes nas seções à seguir.

2.4.2.1 *Single Shot Detectors*

A estrutura de rede *Single Shot Detector* (SSD) foi proposta por Liu et al. (2016) com o objetivo de produzir uma rede capaz de realizar detecção de objetos com precisão e em tempo real, mesmo utilizando *hardware* limitado.

A abordagem SSD é baseada em uma CNN que produz como saída um número fixo de caixas e *scores* de detecção de um determinado objeto em uma imagem, seguido de uma etapa de supressão não máxima para produzir as detecções finais. As primeiras camadas de rede são baseadas em uma arquitetura padrão usada para classificação de imagens de alta qualidade chamada de rede base e, em seguida, são adicionadas estruturas auxiliares na rede para fornecer as seguintes características:

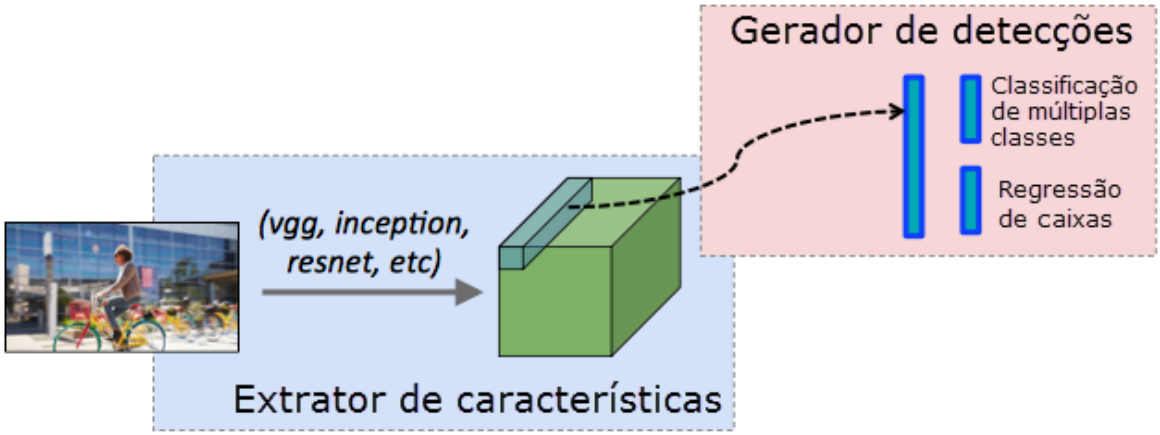
a) **Mapas de características de múltiplas escalas:** essas camadas permitem a detecção de objetos em múltiplas escalas sem a necessidade de uma reiteração da imagem em diferentes tamanhos. Tais camadas são cada vez menores em tamanho e possibilitam a rápida execução da rede.

b) **Caixas de detecção padrão:** essa camada cria milhares de caixas padrão de detecção e as liga a cada célula de mapa de características. As caixas padrão dividem a imagem em partes de forma convolucional, de forma que a posição das caixas com relação a célula de detecção é sempre fixa.

c) **Previsores convolucionais:** cada camada de detecção adicionada pode produzir uma saída fixa de previsões de detecção usando filtros convolucionais.

A Figura 9 ilustra o princípio de funcionamento da rede SSD.

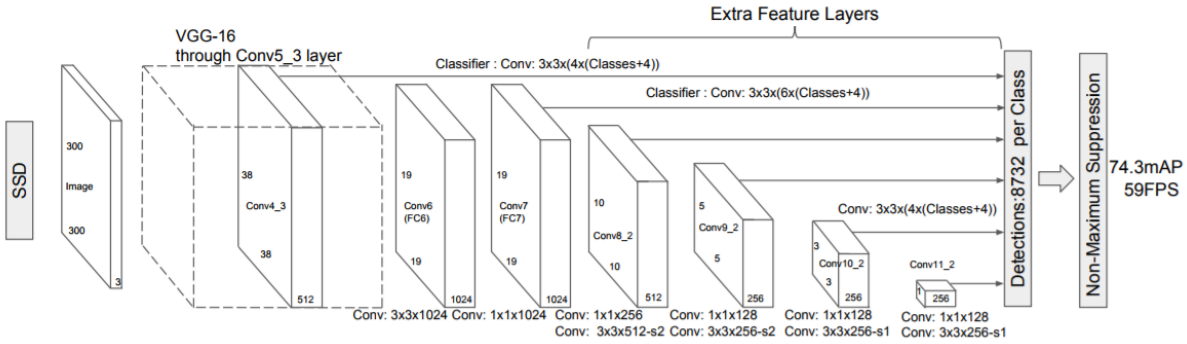
Figura 9 – Estrutura básica da rede *Single Shot Detector*.



Fonte: Adaptado de Lau (2017).

Já a Figura 10 ilustra a estrutura da rede SSD utilizada no artigo de Liu et al. (2016), com todas as camadas e seu respectivo *score* de detecção e a velocidade de processamento em *frames* por segundo (FPS). Em tal figura podemos notar que todas as camadas de detecção geram resultados finais, que são processados por um filtro de supressão. Para cada classe, em uma entrada de tamanho 300 x 300 *pixels*, podem ocorrer até 8732 detecções.

Figura 10 – Rede *Single Shot Detector*.



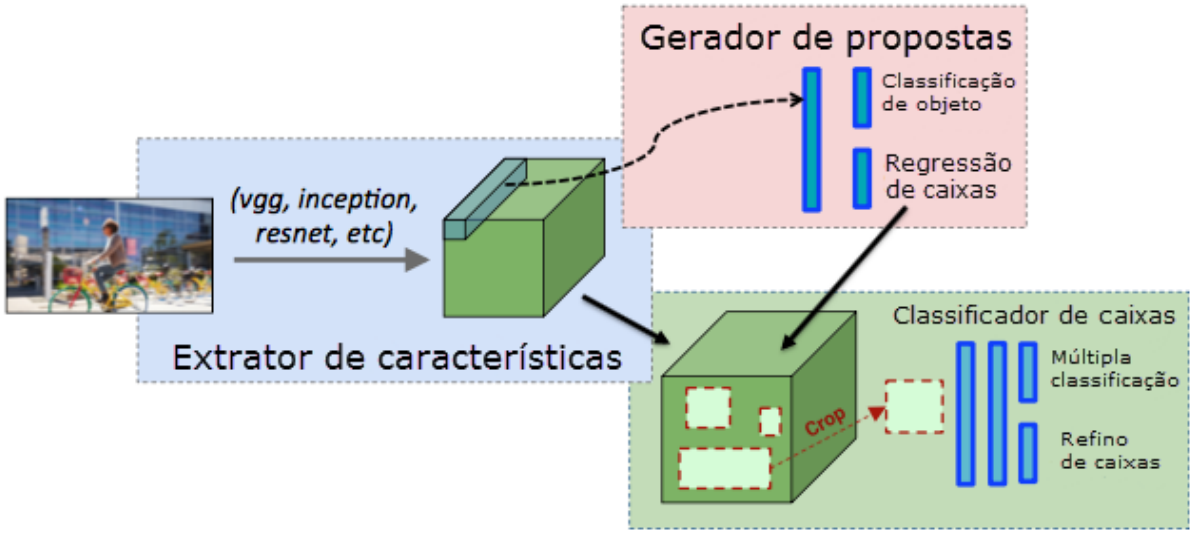
Fonte: Adaptado de Liu et al. (2016).

Resultados encontrados experimentalmente por Liu et al. (2016) nos conjuntos de dados PASCAL VOC, COCO e ILSVRC confirmam que as redes SSD possuem precisão competitiva com métodos que utilizam um passo extra de proposta de região e são muito mais rápidas, ao mesmo tempo que fornecem uma estrutura unificada para formação e inferência. Para entrada de 300 x 300, SSD atinge 74,3% de mAP no teste VOC2007 a 59 FPS em uma Nvidia Titan X; e para entrada 512 x 512, o SSD atinge 76,8% de mAP, superando um modelo R-CNN mais avançado de 2016. Comparado com outros métodos de estágio único, o SSD tem uma precisão muito melhor, mesmo com um tamanho de imagem de entrada menor (LIU et al., 2016).

2.4.2.2 *Regions with Convolutional Neural Networks*

As redes *Regions with Convolutional Neural Networks* (R-CNN) são modelos de redes neurais artificiais complexos para detecção e classificação de objetos com alta performance. A Figura 11 ilustra o princípio de funcionamento da rede RCNN.

Figura 11 – Estrutura básica da rede *Regions with Convolutional Neural Networks*.



Fonte: Adaptado de Lau (2017).

Diferentemente das redes SSD, que realizam a detecção em uma única etapa, as R-CNN's possuem um funcionamento em três etapas, descritas a seguir.

a) **Adaptação de cor e profundidade:** através de filtros de cor, forma e detecção de borda, a imagem é transformada, para servir como entrada para as próximas camadas da rede.
b) **Gerador de propostas de regiões:** é uma rede específica para detectar e demarcar a existência ou não dos objetos a serem detectados, esta rede não classifica ou gera probabilidades de detecção dos objetos desejados, apenas propõe possíveis regiões ondem podem haver objetos de interesse com base nas características para qual foi treinada.
c) **Classificador:** o classificador recebe informações das duas primeiras etapas e tem como objetivo classificar as regiões propostas em objetos e fornecer um índice de certeza para tal predição.

A arquitetura R-CNN foi inicialmente proposta por Girshick et al. (2014) como uma rede capaz de realizar a detecção precisa de classes que atingiu desempenho superior à 30% comparado às redes mais avançadas da época no teste VOC2012. Seu principal problema era que quantidade de convoluções e detecções necessárias para execução da rede era muito grande e, embora a rede fosse precisa, era também extremamente lenta.

Com o avanços das pesquisas, foi proposta a rede *Fast R-CNN*, também por Girshick (2015), que tinha como objetivo reduzir o impacto das camadas de classificação dos objetos através da união das camadas de extração de características iniciais com as camadas de classificação, resultando em uma redução do tempo de processamento por um fator de

aproximadamente 10 mantendo a mesma precisão de resultados.

O último problema da rede era a camada de geração de propostas, que era o gargalo da rede *Fast R-CNN*. Esse problema foi solucionado quando, em 2016, foi apresentada a rede *Faster R-CNN*, por Ren et al. (2015). Tais autores propuseram uma nova camada de geração de propostas chamada de *Region Proposal Network* (RPN), que compartilha convoluções entre as outras camadas e regiões similares da rede, de forma a reduzir o tempo de proposta a níveis marginais enquanto aumenta a qualidade das propostas. Com tais modificações, foi possível obter uma melhoria de precisão de $2,8\%$ em relação à rede anterior e também uma redução no tempo de execução em 5%.

Com essas melhorias, as redes R-CNN conseguem ser executadas em quase tempo real. Utilizando o modelo proposto, a rede *Faster R-CNN* atingiu $5FPS$ enquanto mantia o estado da arte em detecções nos *datasets* PASCAL VOC 2007 ($73,2\%$), 2012 ($70,4\%$) e MS COCO ($41,5\%$) utilizando apenas 300 propostas de região.

3 Metodologia de pesquisa

Como este trabalho tem objetivo de avaliar o desempenho de sistemas de inteligência artificial em diversas circunstâncias, é necessária uma boa metodologia de criação, treinamento e teste, para que os resultados apresentados sejam os melhores possíveis. Em boa parte dos trabalhos que utilizam inteligência artificial, existe uma quantidade complexa de variáveis e etapas necessárias para obtenção de resultados com validade real. Portanto, o presente trabalho foi dividido em três etapas: criação do *dataset*, treinamento das redes e avaliação dos resultados.

A primeira etapa consiste na criação de um *dataset* com um número de imagens mínimo para garantir convergência dos modelos de detecção. A etapa seguinte consiste na escolha e treinamento das redes utilizando um *cluster* computacional. Depois é necessário um novo conjunto de dados semelhante para teste dos resultados e análise de performance do sistema. As imagens utilizadas tanto no treinamento quanto no teste da rede para avaliação serão com as árvores em estágio de crescimento inicial, que possibilita fácil extração dos indivíduos, como mostrado na parte direita da Figura 12.

Nas próximas seções, são detalhadas e justificadas as metodologias utilizadas no trabalho.

Figura 12 – Contraste entre duas idades diferentes em uma mesma cultura.

Fonte: Araujo (2012)

3.1 Obtenção das imagens

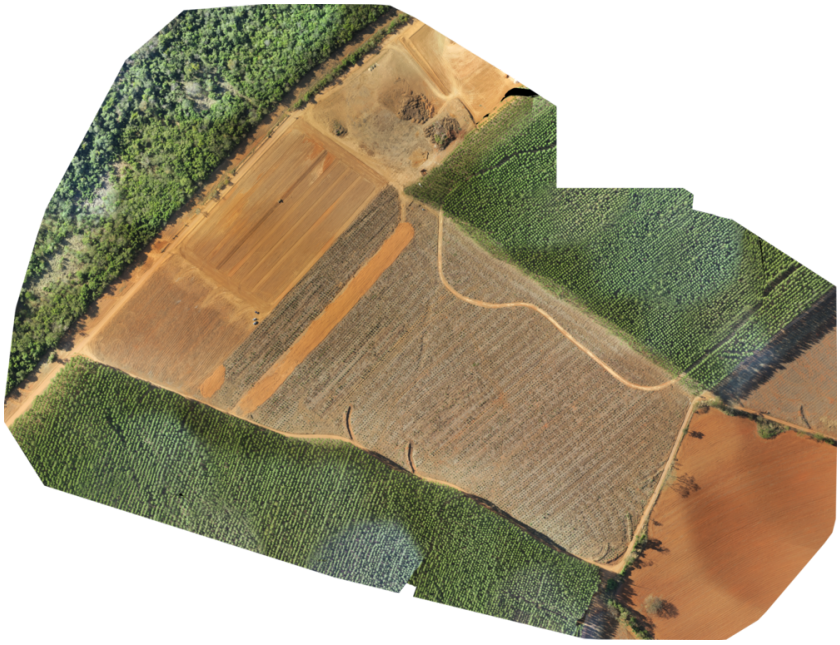
As imagens foram obtidas pela empresa Hórus Aeronaves utilizando um Veículo Aéreo Não Tripulado (ou VANT) de mapeamento. Este método de mapeamento possibilita imagens de maior resolução que os métodos tradicionais de sensoriamento remoto, como imagens de satélite, e estão cada vez mais acessíveis. O VANT utilizado foi o modelo Maptor da empresa Hórus Aeronaves, o qual é mostrado na Figura 13.

Figura 13 – Drone utilizado no mapeamento.

Fonte: Horus Aeronaves.

As fotos foram tiradas com uma câmera embarcada de 20MP e de uma altura de $130m$. As imagens foram então processadas no *software Pix4D* de forma a obter um ortomosaico da área do voo, mostrado na Figura 14. A área de interesse na imagem é a área central com uma coloração marrom acinzentada, e contém a plantação escolhida para estudo.

Figura 14 – Ortomosaico da área de voo.



Fonte: Horus Aeronaves.

3.2 Criação de *dataset*

Em toda implementação e análise de sistemas com inteligência artificial, são necessários dois conjuntos de dados (*datasets*), um para treinamento, e um para avaliação do desempenho. Esses conjuntos devem ser cuidadosamente selecionados de forma a evitar problemas com relação ao treinamento da rede. Um bom *dataset* de treinamento contém amostras suficientes para representar boa parte dos casos presentes em uma análise e em quantidade suficiente para fazer com que o sistema chegue em uma boa convergência.

Para criação dos *datasets* utilizados neste trabalho, foi utilizada a ferramenta de código livre *LabelImg*, disponível em repositório no *GitHub* (SILVA, 2017), cuja interface está mostrada na Figura 15, em conjunto com a ferramenta online de serviços de rotulação de dados da *Amazon*, a Amazon Mturk (2017). Como regra escolhida para separação das imagens, em qualquer um dos *datasets*, apenas o dossel das plantas foi recortado sem selecionar sua sombra e sem que apareça parte de uma planta adjacente na amostra.

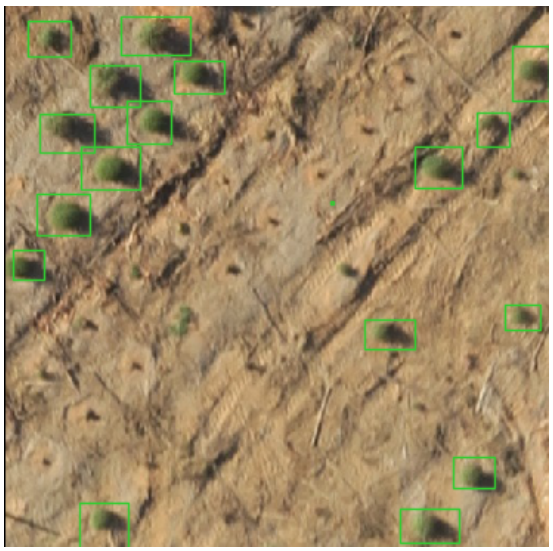
Figura 15 – Interface do software de categorização das amostras

Fonte: Produzido pelo autor.

Foram treinadas e avaliadas três redes neurais de modelos complexos com três tipos diferentes de imagens para verificação da acuracidade de detecção e rotulação de indivíduos em uma plantação.

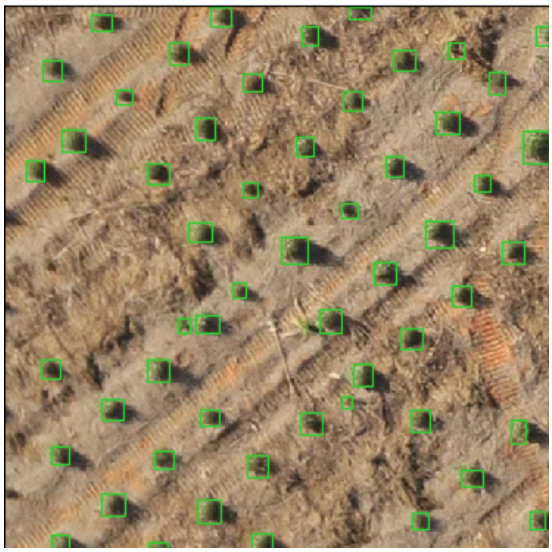
Na Figura 17 podemos ver o recorte de uma boa amostra para treinamento do sistema, enquanto que na Figura 16 observamos uma amostra ruim, com as marcações sofrendo sobreposição e caixa de demarcação grande, englobando a sombra da árvore, que pode levar a *overtraining* da rede para este *dataset* com a sombra neste ângulo.

Figura 16 – Amostra inadequada



Fonte: Produzido pelo autor.

Figura 17 – Amostra adequada



Fonte: Produzido pelo autor.

3.2.1 Redução do *dataset* de processamento

Como os ortomosaicos geralmente possuem dezenas de quilômetros quadrados de informações, é necessário processar esses dados de forma fracionada, pois redes neurais computacionais exigem um alto custo computacional mesmo utilizando pequenas imagens, os quais estão associados à complexidade dos modelos utilizados. Por este motivo, foi escolhido recortar a imagem em partes de 512 x 512 *pixels* de resolução. As implicações dessa escolha serão discutidas no Capítulo 6.

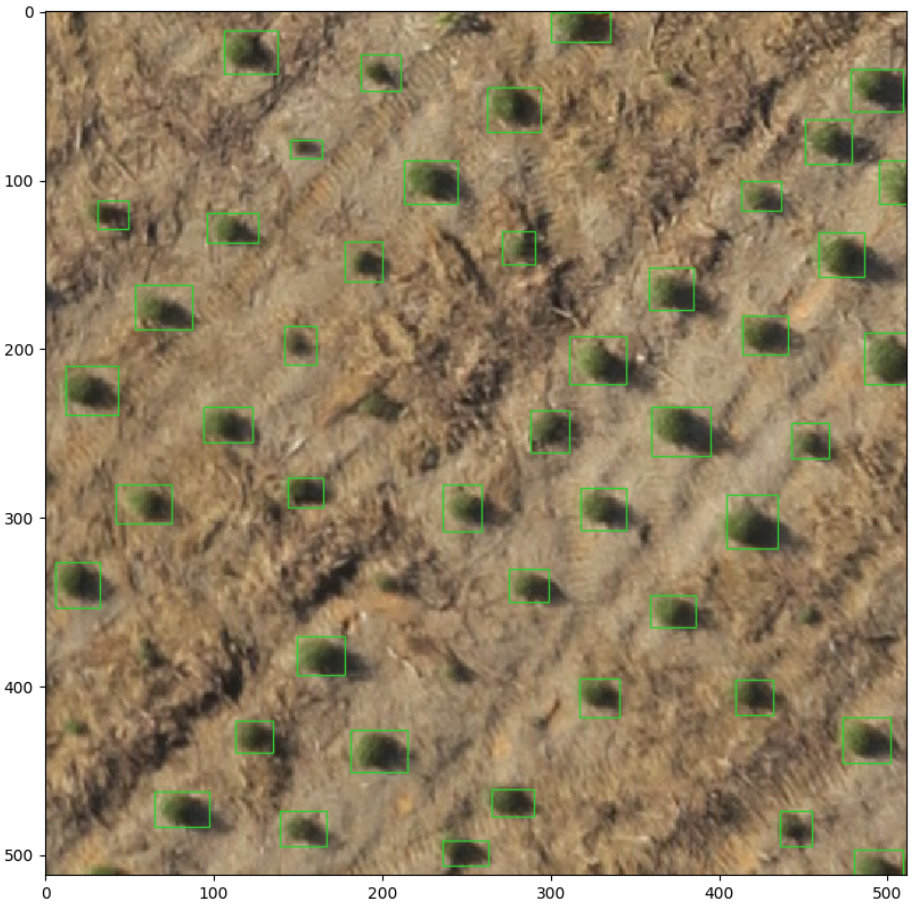
Após o recorte da imagem, todas os recortes que fazem parte da área de interesse são demarcados de acordo com os procedimentos indicados da metodologia deste trabalho. Utilizando a ferramenta *Amazon Mechanical Turk* foi possível rotular toda a área de interesse com ótima qualidade e de maneira rápida. Após a conferência do *dataset* e extração das informações, conseguimos obter a posição e quantidade total de indivíduos da área plantada. O total de indivíduos demarcados na plantação é de 7866 e os dados de posição da árvore são armazenados em um arquivo de texto com as informações de nome do arquivo e coordenadas máximas e mínimas do contorno de cada árvore, como mostrado na Tabela 1 e na Figura 18.

Tabela 1 – Trecho do dataset

filename	xmin	ymin	xmax	ymax
tile_15_21.jpg	145	76	164	87
tile_15_21.jpg	187	25	211	47
tile_15_21.jpg	31	112	49	129
tile_15_21.jpg	96	119	126	137
tile_15_21.jpg	53	162	87	188
tile_15_21.jpg	12	210	43	239
tile_15_21.jpg	94	234	123	255
tile_15_21.jpg	42	280	75	303
tile_15_21.jpg	6	326	32	353
tile_15_21.jpg	213	88	245	114
tile_15_21.jpg	262	45	294	71

Fonte: Produzido pelo autor.

Figura 18 – Imagem indicada na Tabela 1.



Fonte: Produzido pelo autor.

Com as informações de posicionamento, o *dataset* é então separado aleatoriamente

em porções de treinamento e de avaliação, na proporção de 70% para treinamento e 30% para teste dos resultados. Então, das 228 imagens, foram selecionadas 163 para o treinamento e 65 para teste da rede. O *dataset* foi separado utilizando um algoritmo de separação aleatória disponível no repositório deste trabalho. Com a finalização dessa etapa, possuímos todas as árvores demarcadas e com imagens de um tamanho aceitável para o processamento pela rede, além de estarem separadas em dois grupos: um para treinamento e outro para teste.

3.2.2 Aplicação de índices de vegetação

Com o *dataset* já separado, conseguimos aplicar os índices de vegetação para comparar o desempenho das diversas redes com diferentes tipos de imagem. Para aplicação do índice, foi desenvolvida uma ferramenta em *Python* para processar todas as imagens e aplicar os índices escolhidos a cada *pixel* individualmente.

Considerando que o *dataset* reduzido já é em RGB, serão aplicadas as seguintes transformações:

- **IFV**: Índice de Folha Verde.
- **IVV**: Índice de Vegetação Visível.

  O índice IFV é definido pela equação 3.1,

$$IFV = \frac{2 \times G - R - B}{2 \times G + R + B} \tag{3.1}$$

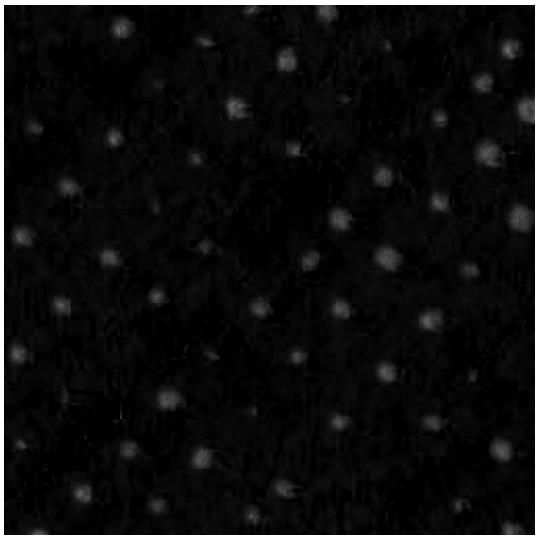
onde R, G e B são os canais vermelho, verde e azul da imagem, respectivamente, e IFV é o valor calculado do índice.

Com a aplicação de 3.1, podemos obter na saída uma visualização gráfica em escala de cinza apresentada na Figura 20, ao lado da imagem RGB original, ilustrada pela Figura 19.

Figura 19 – Imagem original RGB.

Fonte: Produzido pelo autor.

Figura 20 – Índice IFV aplicado.



Fonte: Produzido pelo autor.

O índice IVV é definido pela equação 3.2,

$$IVV = \left[\left(1 - \left|\frac{R - R_0}{R - R_0}\right|\right) \times \left(1 - \left|\frac{G - G_0}{G + G_0}\right|\right) \times \left(1 - \left|\frac{B - B_0}{B - B_0}\right|\right)\right]^{1/w} \tag{3.2}$$

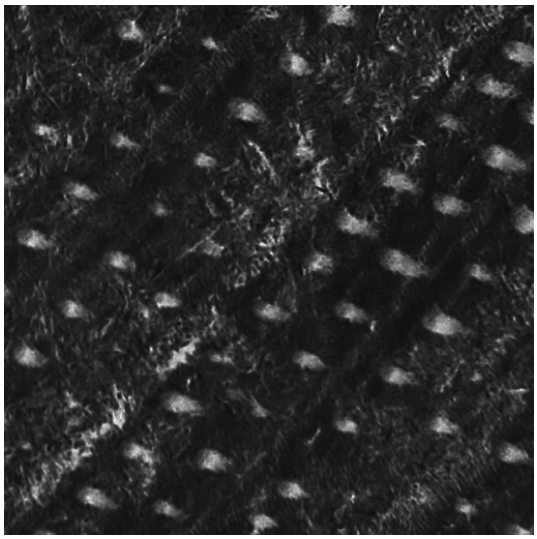
onde as variáveis R, G e B como as entradas dos canais de cor vermelho, verde e azul, respectivamente. Além disso, nessa equação, é necessário um valor de referencia de cor (R_0, G_0 e B_0) e um fator de amortecimento w, para os quais foram utilizados os valores recomendados pelos autores do índice ($R_0 = 30$, $G_0 = 50$, $B_0 = 0$ e $w = 1$). Além disso, para evitarmos possíveis divisões por zero, foi somado um pequeno valor de 10 em todos os canais da imagem e na referência de cor.

Após o processamento da imagem obtemos uma saída como ilustrada na Figura 22, ao lado da imagem RGB original, ilustrada pela Figura 21.

Figura 21 – Imagem original RGB.

Fonte: Produzido pelo autor.

Figura 22 – Índice IFV aplicado.



Fonte: Produzido pelo autor.

Comparando os índices das Figuras 20 e 22, podemos verificar que a imagem do Índice de Folha Verde possui menos artefatos que o Índice de Vegetação Visível. Porém, no segundo índice as árvores são mais ressaltadas e tomam uma forma mais característica.

Com os *datasets* separados, geramos arquivos do tipo *TFRecord* utilizados pela plataforma de inteligência artificial *TensorFlow*. Esse formato coloca todas as informações necessárias ao treinamento da rede em um único arquivo, incluindo imagens e a localização das árvores.

3.3 *Software* de processamento

Para o treinamento das redes, foi utilizada a linguagem de programação *Python* e biblioteca de aprendizado de máquina *TensorFlow*, da Google, devido à facilidade de aplicação das tecnologias e grande comunidade para suporte e discussões sobre o tema. O *software* de processamento utilizado neste trabalho foi desenvolvido por Huang et al. (2016a) para comparação de desempenho de diferentes modelos de redes neurais para detecção de objetos utilizando os *datasets* Common Objects on Context (COCO), da Microsoft, e PETS, da Universidade de Oxford. Ele é distribuído sob a licença de código aberto *Apache License 2.0*, que permite a utilização livre e comercial, desde que mantidos o *Copyright* e avisos da licença.

Nesta ferramenta estão disponíveis as seguintes arquiteturas de redes de detecção:

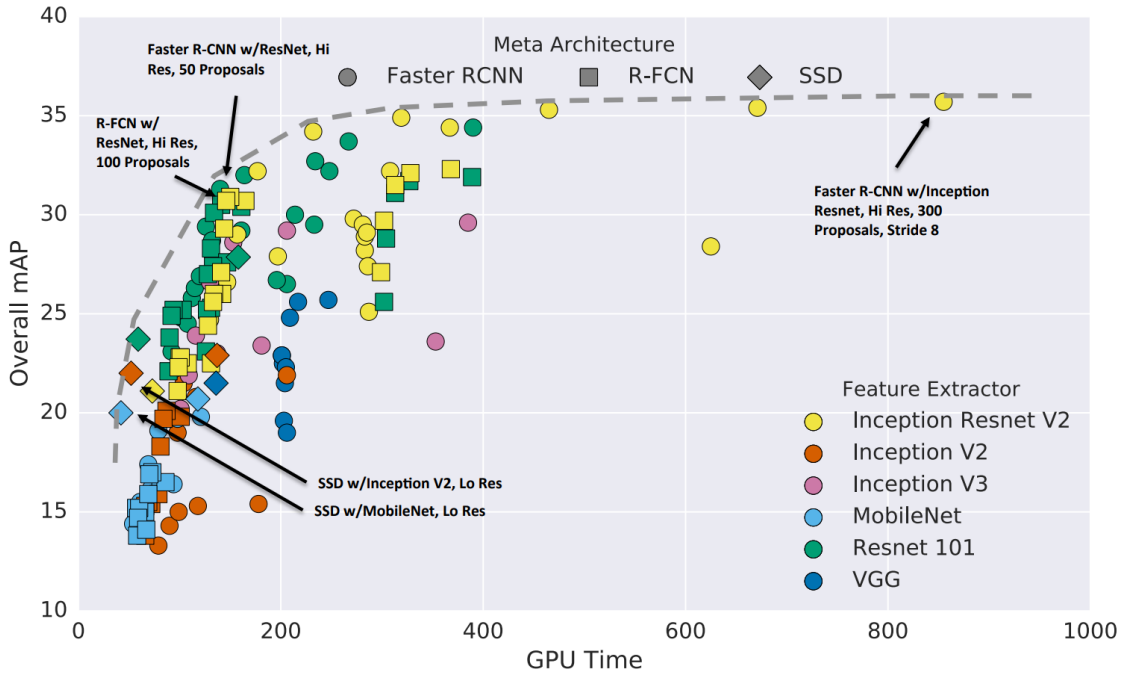
- ***Single Shot Detectors*** **(SSD)**:
  - Mobilenet v1
  - Inception v2
- ***Faster Region-proposal Convolutional Neural Network*** **(Faster R-CNN)**:
  - ResNet 50

- ResNet 101
- Inception ResNet v2

3.4 Escolha das redes

Como existem diversas redes com diferentes características disponíveis na ferramenta e o objetivo do trabalho é comparar as redes em precisão e velocidade, foram escolhidas as redes de forma a obter um bom espectro em termos de resultados. A Figura 23 mostra uma curva de comparação entre os principais tipos de redes disponíveis no software utilizando o *dataset* COCO. No eixo horizontal está representada a velocidade de processamento e no eixo vertical está representada a precisão das redes.

Figura 23 – Comparação entre precisão e velocidade de redes de detecção.



Fonte: Huang et al. (2016a).

De forma a obter um bom espectro de comparação, foram utilizadas três redes com diferentes características: uma rede *Single Shot Detector* (SSD) e duas redes *Regions with Convolutional Neural Networks* (R-CNN).

A rede SSD escolhida utiliza o extrator de características *Inception* V2, que definiram o estado da arte nos desafios ILSVRC 2014. De acordo com o trabalho de Huang et al. (2016a), as redes SSD são extremamente rápidas e consomem poucos recursos, podendo ser utilizadas em sistemas portáteis com latências muito baixas. Porém, esta rede sofre com uma menor taxa de acerto e precisão que as outras.

A primeira rede R-CNN escolhida utiliza o extrator de características *ResNet* 101

e está localizada no cento do espectro de detecção da Figura 23. Ela possui resultados superiores em precisão com relação à rede SSD porém sofre um pouco com tempo de execução e uso de recursos, tornando inviável para utilização em pequenos sistemas embarcados atuais com baixo consumo de potência.

A segunda rede R-CNN escolhida utiliza o extrator de características *Inception ResNet V2* e está localizada no lado extremo direito da curva. Ela possui altíssima precisão, porém seu uso de recursos é extremo, o que a torna inviável para dispositivos móveis e necessita de máquinas com alta performance para que possa ser utilizada de forma aceitável com relação à velocidade.

A Tabela 2 apresenta uma comparação entre as redes escolhidas treinadas no *dataset* COCO, rodando em uma máquina com placa de vídeo *Nvidia GeForce GTX TITAN X* e com imagens de tamanho $600x600$ *pixels*. A métrica *mAP* é uma medida de precisão com relação ao dataset COCO e deve ser interpretada de forma que o maior número indica um melhor desempenho, tal métrica é exemplificada detalhadamente na seção 3.7.1.

Tabela 2 – Comparação de desempenho das redes escolhidas no dataset COCO

Modelo	Tempo (ms)	COCO mAP
SSD Inception v2	42	24
Faster R-CNN ResNet 101	106	32
Faster R-CNN Inception ResNet v2	620	37

Fonte: Adaptado de Huang et al. (2016b).

3.5 Configuração das redes

Em sua maior parte, as configurações da rede foram deixadas com o padrão fornecidos pelos pesquisadores. Os arquivos de configuração das redes utilizadas podem ser encontrados nos Anexos A, B e C, com seus respectivos nomes de rede.

Os principais valores alterados foram relacionados à duração do treinamento, pois como foram utilizados modelos pré-treinados, não foi necessária uma grande quantidade de passos para finalizar o treinamento e chegarmos à uma convergência. O treinamento foi limitado à 3000 passos com uma redução da taxa de aprendizado para 1/10 do valor inicial aos 1500 passos. Essa técnica é conhecida como otimização de gradiente de treinamento, e permite que a rede chegue cada vez mais próxima do estado ótimo de treinamento.

3.6 Treinamento das redes

O treinamento de redes neurais complexas necessita de hardware potente devido à grande complexidade dos modelos utilizados. Atualmente, o método mais eficiente de se treinar redes neurais é utilizando placas de vídeo, que possuem centenas de unidades de

processamento e aceleram o processo de aprendizado de redes com diversas camadas e coeficientes. Uma das principais demandas de sistemas de *deep learning* é a necessidade de uma grande quantidade de memória para realizar o treinamento e avaliação das redes.

Devido ao alto custo de se obter *hardware* para treinamento de redes neurais, foi optado por realizar o treinamento e avaliação das redes em máquinas remotas na nuvem da *Amazon*. A máquina utilizada para o treinamento e avaliação de todas as redes possui quatro processadores, 61 GB de memória RAM e uma instância de GPU NVIDIA K80 com 12 GB de memória de vídeo, além de possuir armazenamento por SSD.

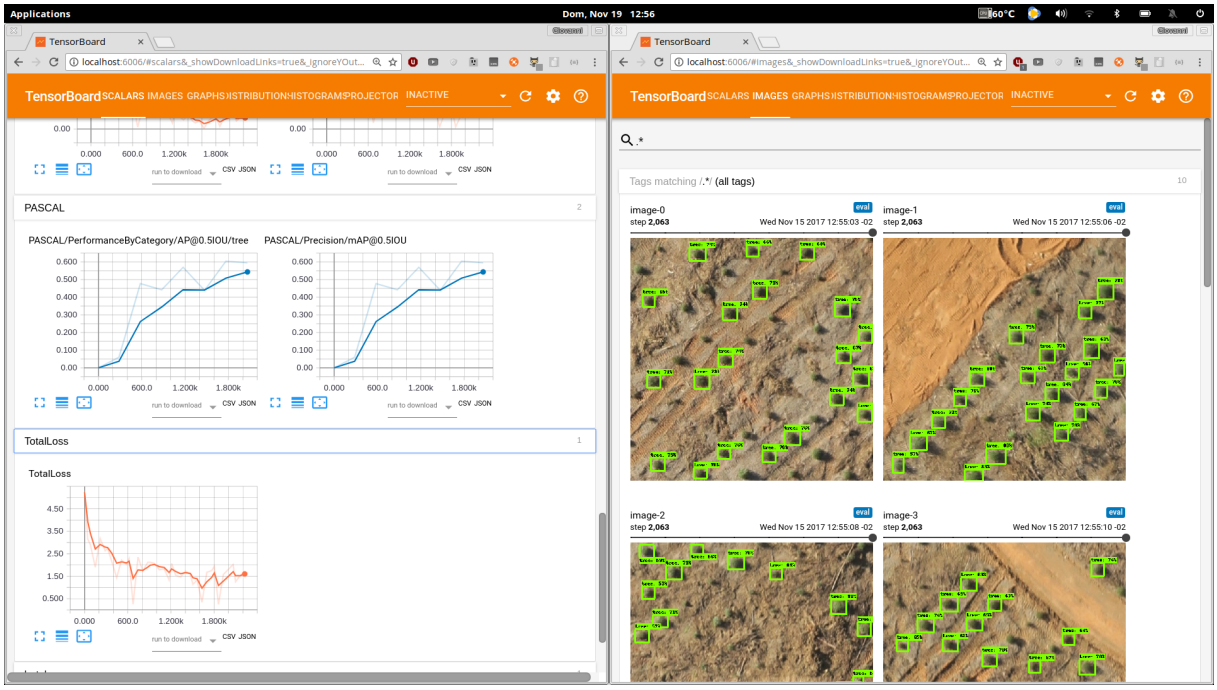
Foi utilizado um sistema Linux baseado em Ubuntu com as seguintes ferramentas instaladas:

- Python 3.6.2
- TensorFlow Models
- TensorFlow 1.4.0
- NVIDIA CUDA v8.0
- NVIDIA cuDNN v6

O treinamento e avaliação das redes foi realizado simultaneamente utilizando os programas de treinamento e avaliação do módulo *object_detection* da biblioteca *TensorFlow Models*.

O programa de treinamento inicia o treinamento e salva os passos do treinamento a cada 120 segundos, por outro lado, o programa de avaliação fica ativamente procurando passos salvos e rodando eles sobre o dataset de avaliação. Como os softwares rodam em linha de comando, a única saída de estado do treinamento, além dos arquivos de *checkpoint*, é uma informação de perda total da rede. Para resolver o problema do monitoramento do progresso do aprendizado e obter estatísticas mais detalhadas, foi utilizada a ferramenta *Tensorboard*, que fornece uma *interface* via navegador com todos os gráficos de andamento do treinamento, conforme mostrado na Figura 24.

Figura 24 – *Interface* de monitoramento fornecida pelo TensorBoard.



Fonte: Produzido pelo autor.

Como critério para parar o treinamento, foi observada a variável *TotalLoss* e, após a variável entrar em equilibro por um determinado tempo, o treinamento foi parado.

De forma a facilitar o treinamento e avaliação das redes, foi utilizado um *script* para automatizar a inicialização do sistema e treinamento das redes, que está disponível no repositório do trabalho. A parada do treinamento é realizada manualmente através da linha de comando.

3.7 Avaliação de desempenho

Após o treinamento da rede, é extraído o gráfico de inferência congelado, que é um arquivo com os pesos e a estrutura do modelo pronto e otimizado para ser utilizado. É com este arquivo que foram processados o *dataset* de teste e obtidas as métricas de avaliação.

A avaliação de desempenho das redes será foi realizada de duas maneiras: a métrica de desempenho fornecida pela ferramenta de treinamento e a desempenho de contagem de plantas com nosso *dataset*. As métricas utilizadas estão descritas à seguir.

3.7.1 *Mean Average Precision* (*mAP*)

De acordo com Christopher, Prabhakar e Hinrich (2008), a métrica fornecida pelo sistema de treinamento é a *Mean Average Precision* (*mAP*), que indica a média de precisão entre todas as imagens processadas no *dataset* de avaliação. Esta métrica é útil para

comparar diretamente as diversas redes e seus níveis de certeza, porém não oferece dados confiáveis quanto à contagem e detecção de indivíduos.

O mAP é calculado utilizando a equação 3.3

$$mAP(Q) = \frac{1}{Q} \sum_{j=1}^{Q} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{3.3}$$

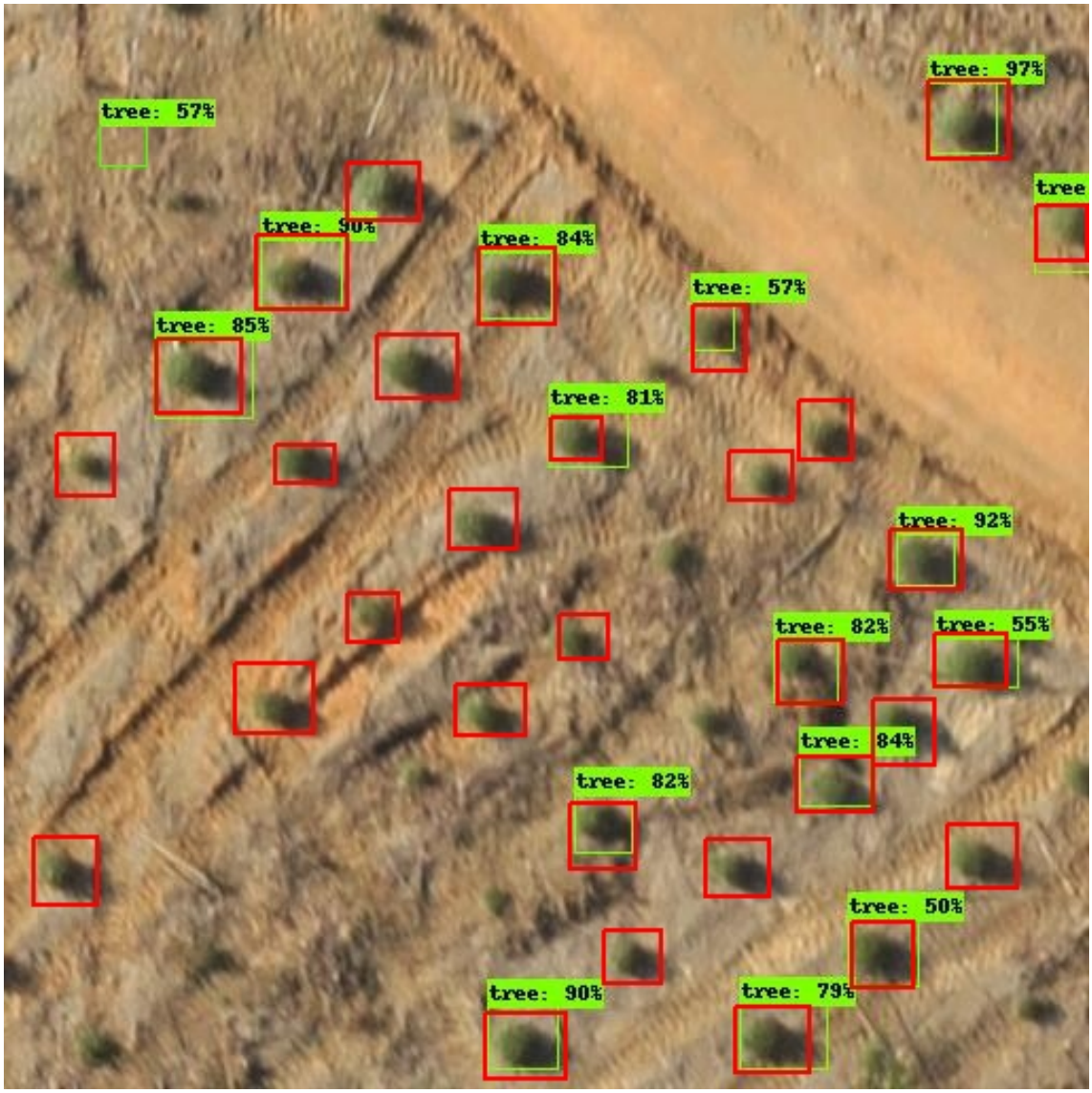
onde Q é o tamanho total de nosso *dataset* de avaliação, m_j é o número total de indivíduos **verdadeiros** em uma determinada imagem e $P(R_{jk})$ é a precisão do indivíduo na imagem. É importante dizer que esta métrica não leva em conta falsos positivos, pois indivíduos detectados que não existem no quadro de verdade não são contabilizados na equação.

De forma a exemplificar o cálculo, será feita uma avaliação manual de uma imagem resultado. Para calcular o mAP total, precisamos primeiro calcular o AP de cada imagem, ou seja, o somatório interno de 3.3, qual é dado por 3.4.

$$AP(m) = \frac{1}{m} \sum_{k=1}^{m_j} P(R_{jk}) \tag{3.4}$$

Na Figura 25, as caixas sem valores representam a verdade, ou seja, as árvores demarcadas no *dataset* de avaliação. Já as caixas com pequenos valores de probabilidade representam as árvores detectadas por uma rede, com seu respectivo valor de certeza.

Figura 25 – Imagem processada por uma rede.



Fonte: Produzido pelo autor.

Utilizando a equação 3.4 e os dados de verdade presente na imagem, obtemos o valor mostrado na equação 3.5.

$$AP = \frac{(0.85 + 0.57 + ... + 0.79 + 0.5)}{31} = 0.403 \tag{3.5}$$

Vale a pena lembrar que o falso positivo no canto superior direito da imagem não está sendo considerado no somatório. Finalmente, para calcularmos o valor mAP, precisamos realizar essa avaliação em todas as imagens e então realizar uma média aritmética dos resultados.

3.8 Contagem de indivíduos

De forma a avaliarmos melhor as redes em relação aos objetivos do trabalho podemos utilizar métricas que não utilizem diretamente os valores de certeza da rede e tamanho e formato das caixas.

O primeiro passo é definir um limiar de corte da probabilidade de detecção fornecida pela rede. O critério escolhido foi utilizar o maior número possível de amostras de forma a manter o erro total abaixo de 5%.

Como a informação de formato da caixa de detecção também não é relevante para nossa análise, foram calculadas e utilizadas apenas as coordenadas centrais. O raio de detecção deste ponto central é calculado à partir do tamanho médio das caixas de detecção encontradas.

Depois de calculados estes valores, podemos fazer uma avaliação estatística utilizando os seguintes valores para indivíduos:

- **Limiar de detecção (L)**: Limite para o qual todas as detecções com probabilidade abaixo deste são desconsideradas.
- **Verdadeiro Positivo (VP)**: Quando uma árvore é detectada corretamente.
- **Dupla Detecção (DD)**: Quando uma árvore é detectada duas vezes.
- **Verdadeiro Positivo Real (VPR)**: O valor de verdadeiros positivos subtraído da quantidade de duplas detecções, que nos fornece a quantidade real de indivíduos detectados.
- **Falso Negativo (FN)**: Quando uma árvore existe no quadro de verdade mas não é detectada.
- **Falso Positivo (FP)**: Quando uma árvore é detectada porém ela não existe em nosso quadro de verdade.

Além disso, trabalharemos com duas taxas percentuais para comparar as redes de forma mais simples.

A taxa de detecção (TD) será a relação entre o número total de indivíduos e o valor Verdadeiro Positivo Real, conforme mostrado 3.6,

$$TD_{\%} = \frac{VPR}{T} \tag{3.6}$$

onde T é o número total de indivíduos e VPR é o número de verdadeiros positivos reais.

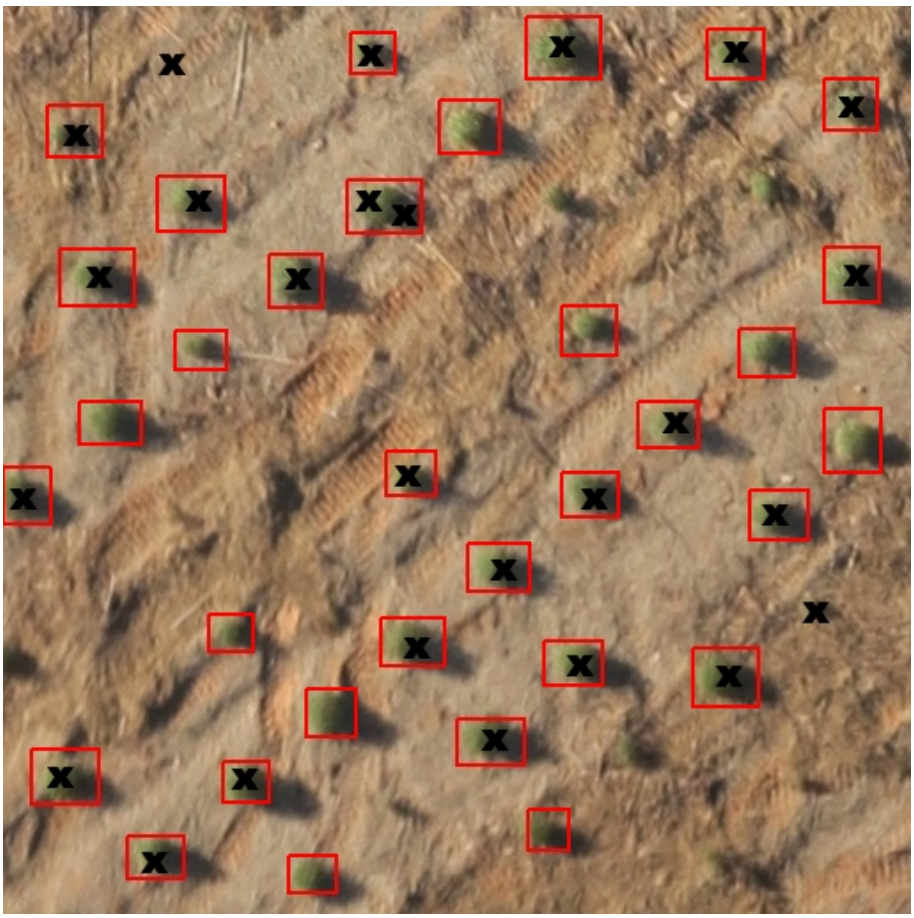
Já a taxa de erro total (TE) leva em consideração todos os valores incorretos de detecção que não podem ser corrigidos por pós-processamento, ou seja, sem incluir o número de duplas detecções, em relação ao valor total de detecções. Assim, tem-se que 3.7

$$TE_{\%} = \frac{FP}{DT} \tag{3.7}$$

onde FN é o número de falsos negativos, FP o número de falsos positivos e DT é a quantidade total de detecções.

A Figura 26 e a Tabela 3 mostram os valores de contagem extraídos de uma imagem processada. Na imagem, os quadrados representam o quadro de verdades e as marcações 'X' representam as saídas da rede já processadas.

Figura 26 – Imagem exemplo para avaliação.



Fonte: Produzido pelo autor.

Tabela 3 – Valores exemplo de avaliação.

Total verdadeiro	**Verdadeiro Positivo**	**Dupla Detecção**	**Verdadeiro Positivo Real**	**Falso Negativo**	**Falso Positivo**
33	24	1	23	10	2

Com os valores verificados conseguimos calcular percentuais referentes à taxa de detecção e taxa de erro das redes para a imagem. Assim, obtemos os valores de $TD_{\%} = 69,3\%$ e $TE_{\%} = 34,2\%$. Vale a pena lembrar que estes valores foram utilzados apenas a fim de demonstrar as formas de avaliação e não correspondem à nenhuma rede em específico.

4 Avaliação das redes e análise de resultados

O treinamento das redes levou uma média de 3 horas por rede para atingir níveis estáveis de detecção e atingir os objetivos de treinamento. Com as redes treinadas, foi utilizado mesmo cluster computacional para realizarmos as avaliações desempenho e precisão.

4.1 Resultados

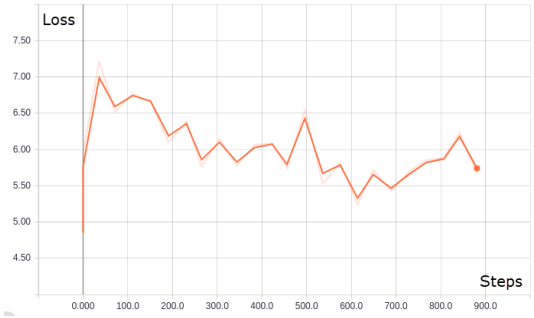
Os resultados foram divididos com relação ao tipo de imagem e subdivididos em categorias de rede para comentários em relação aos dados encontrados.

4.1.1 Imagens RGB

4.1.1.1 SSD Inception V2

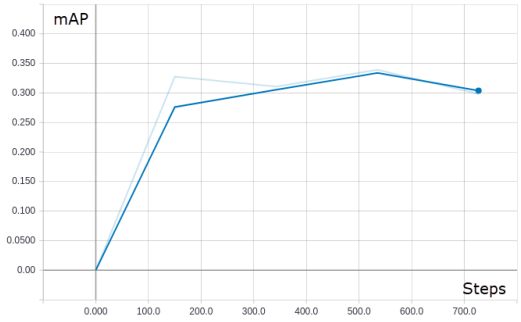
Iniciando com a estrutura de rede mais simples e rápida, a rede atingiu convergência com 900 passos, e a evolução do desempenho da rede durante o treinamento pode ser visualizado nas Figuras 27 e 28.

Figura 27 – Evolução do erro total - SSD Inception V2 - RGB



Fonte: Produzido pelo autor.

Figura 28 – Evolução do mAP - SSD Inception V2 - RGB

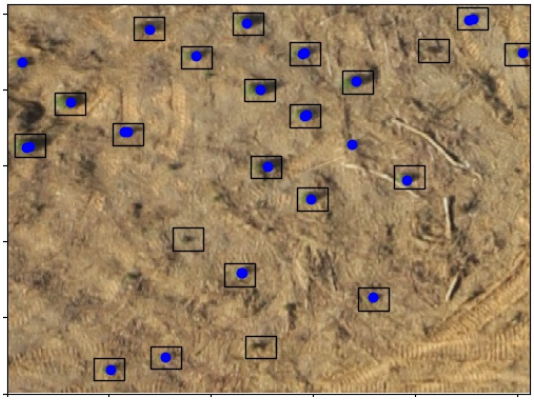


Fonte: Produzido pelo autor.

Na evolução do erro total, podemos ver que há uma leve descida nos valores. O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,2983$.

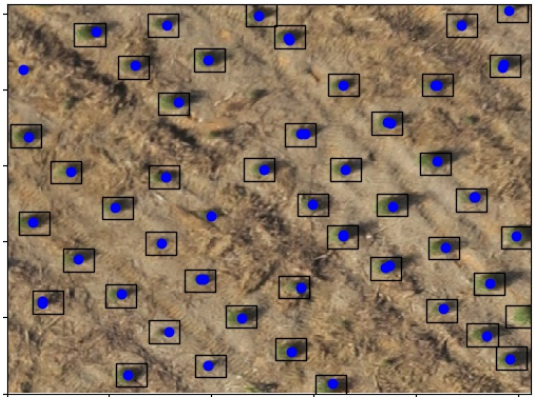
A Figura 29 e a Figura 30 ilustram a saída da rede SSD Inception V2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 29 – Detecção exemplo 1 - SSD Inception V2 - RGB



Fonte: Produzido pelo autor.

Figura 30 – Detecção exemplo 2 - SSD Inception V2 - RGB



Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 4.

Tabela 4 – Desempenho da rede SSD Inception V2 utilizando imagens RGB

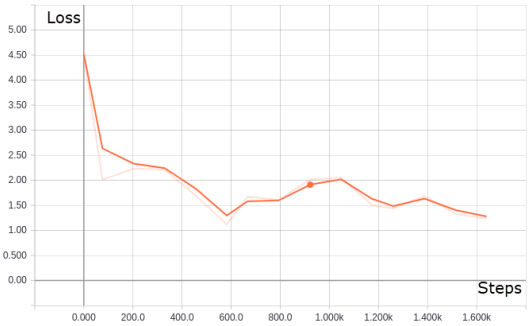
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
25,5	8670	7915	7381	439	484	93,7 %	5,0 %	81 ms

Fonte: Produzido pelo autor.

4.1.1.2 R-CNN ResNet 101

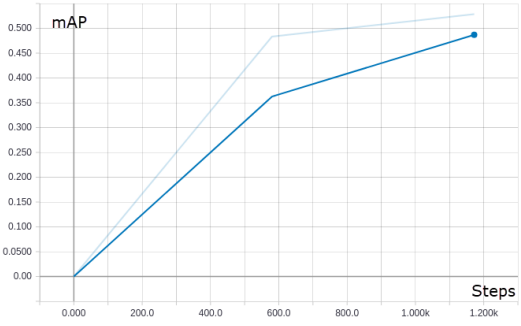
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 31 e 32.

Figura 31 – Evolução do erro total - R-CNN ResNet 101 - RGB



Fonte: Produzido pelo autor.

Figura 32 – Evolução do *mAP* - R-CNN ResNet 101 - RGB



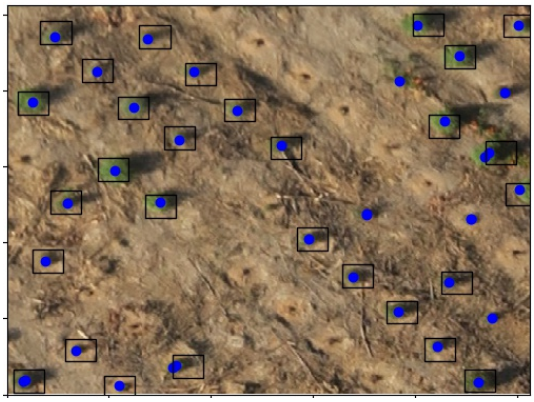
Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,5284$.

A Figura 33 e a Figura 34 ilustram a saída da rede R-CNN Resnet 101 sobreposta

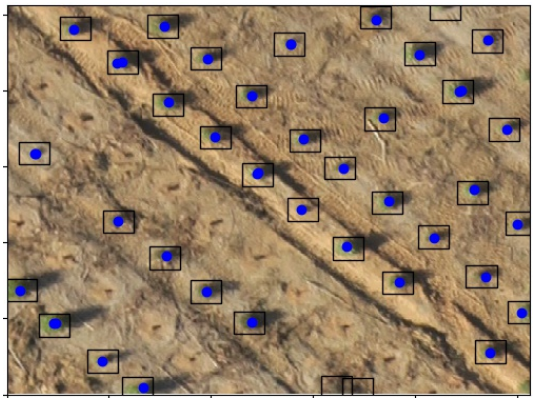
na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 33 – Detecção exemplo 1 - R-CNN ResNet 101 - RGB



Fonte: Produzido pelo autor.

Figura 34 – Detecção exemplo 2 - R-CNN ResNet 101 - RGB



Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 5.

Tabela 5 – Desempenho da rede R-CNN Resnet 101 utilizando imagens RGB

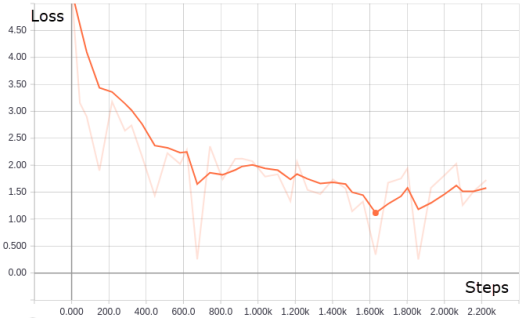
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
31,0	9028	8249	7471	452	394	95,0 %	5,0 %	578 ms

Fonte: Produzido pelo autor.

4.1.1.3 R-CNN Inception ResNet v2

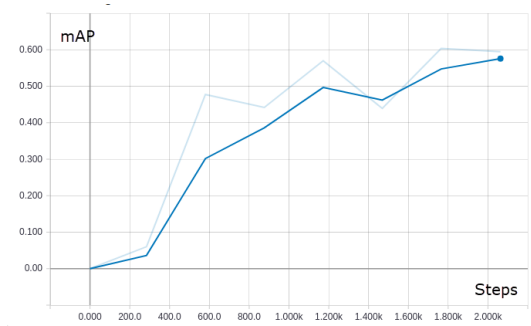
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 35 e 36.

Figura 35 – Evolução do erro total - R-CNN Inception ResNet v2 - RGB



Fonte: Produzido pelo autor.

Figura 36 – Evolução do *mAP* - R-CNN Inception ResNet v2 - RGB

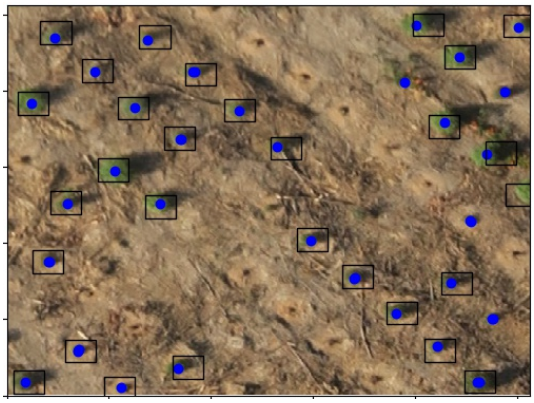


Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,5945$.

A Figura 37 e a Figura 38 ilustram a saída da rede R-CNN Inception ResNet v2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 37 – Detecção exemplo 1 - R-CNN Inception ResNet v2 - RGB



Fonte: Produzido pelo autor.

Figura 38 – Detecção exemplo 2 - R-CNN Inception ResNet v2 - RGB

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 6.

Tabela 6 – Desempenho da rede R-CNN Inception ResNet v2 utilizando imagens RGB

L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
18,9	9831	8991	7468	496	397	95,0 %	5,0 %	1102 ms

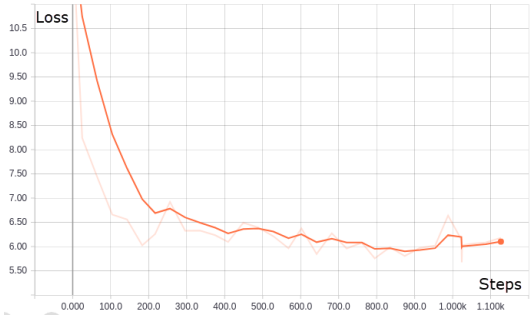
Fonte: Produzido pelo autor.

4.1.2 Índice IFV

4.1.2.1 SSD Inception v2

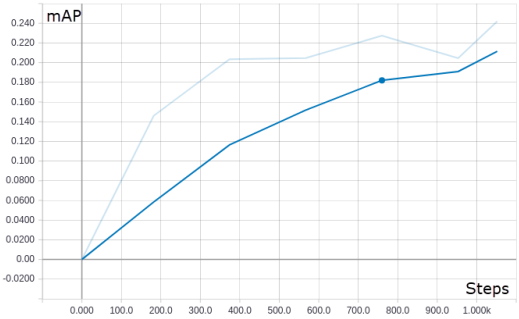
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 39 e 40.

Figura 39 – Evolução do erro total - SSD Inception v2 - IFV



Fonte: Produzido pelo autor.

Figura 40 – Evolução do *mAP* - SSD Inception v2 - IFV



Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,0430$.

A Figura 41 e a Figura 42 ilustram a saída da rede SSD Inception v2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 41 – Detecção exemplo 1 - SSD Inception v2 - IFV.

Fonte: Produzido pelo autor.

Figura 42 – Detecção exemplo 2 - SSD Inception v2 - IFV.

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 7.

Tabela 7 – Desempenho da rede SSD Inception v2 utilizando imagens com índice IFV.

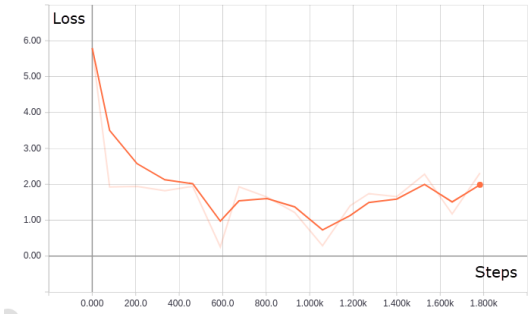
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
27,3	24	9	9	1	7856	0,1 %	4,2 %	6 ms

Fonte: Produzido pelo autor.

4.1.2.2 R-CNN ResNet 101

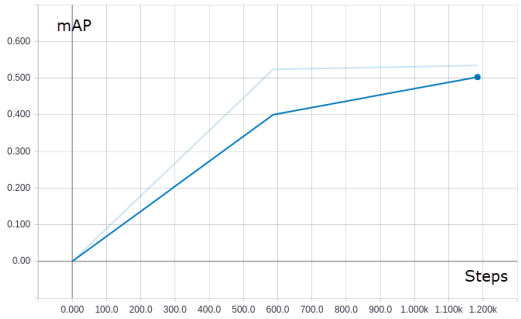
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 43 e 44.

Figura 43 – Evolução do erro total - R-CNN ResNet 101 - IFV



Fonte: Produzido pelo autor.

Figura 44 – Evolução do mAP - R-CNN ResNet 101 - IFV



Fonte: Produzido pelo autor.

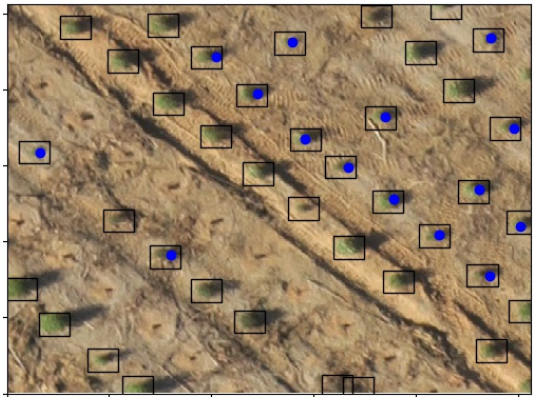
O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,5343$.

A Figura 45 e a Figura 45 ilustram a saída da rede R-CNN ResNet 101 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 45 – Detecção exemplo 1 - R-CNN ResNet 101 - IFV.

Fonte: Produzido pelo autor.

Figura 46 – Detecção exemplo 2 - R-CNN ResNet 101 - IFV.



Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 11.

Tabela 8 – Desempenho da rede R-CNN ResNet 101 utilizando imagens com índice IFV.

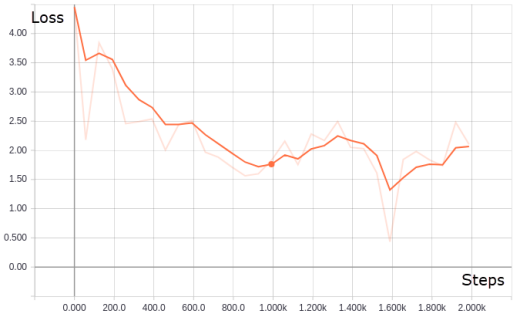
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
24,0	3143	2810	2748	157	5117	34,9 %	5,0 %	570 ms

Fonte: Produzido pelo autor.

4.1.2.3 R-CNN Inception ResNet v2

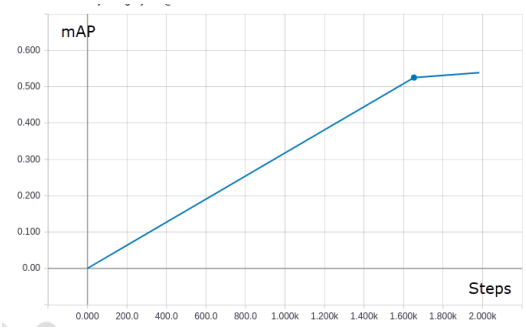
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 59 e 60.

Figura 47 – Evolução do erro total - R-CNN ResNet Inception V2 - IFV



Fonte: Produzido pelo autor.

Figura 48 – Evolução do mAP - R-CNN ResNet Inception V2 - IFV



Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,5386$.

A Figura 61 e a Figura 61 ilustram a saída da rede R-CNN ResNet Inception V2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 49 – Detecção exemplo 1 - R-CNN ResNet Inception V2 - IFV.

Fonte: Produzido pelo autor.

Figura 50 – Detecção exemplo 2 - R-CNN ResNet Inception V2 - IFV.

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 12.

Tabela 9 – Desempenho da rede R-CNN ResNet Inception V2 utilizando imagens com índice IFV.

L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
1,5	7859	7214	4474	368	3391	56,9 %	4,7 %	1101 ms

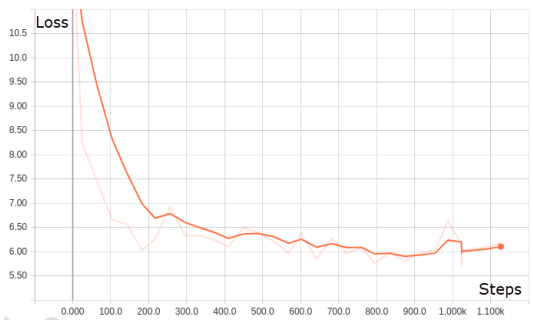
Fonte: Produzido pelo autor.

4.1.3 Com índice VVI

4.1.3.1 SSD Inception v2

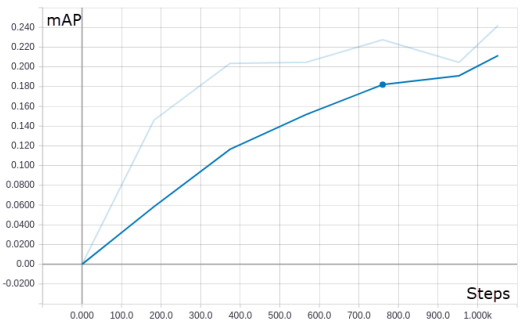
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 51 e 52.

Figura 51 – Evolução do erro total - SSD Inception v2 - IVV



Fonte: Produzido pelo autor.

Figura 52 – Evolução do *mAP* - SSD Inception v2 - IVV



Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,2420$.

A Figura 54 e a Figura 54 ilustram a saída da rede SSD Inception v2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 53 – Detecção exemplo 1 - SSD Inception v2 - IVV.

Fonte: Produzido pelo autor.

Figura 54 – Detecção exemplo 2 - SSD Inception v2 - IVV.

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 10.

Tabela 10 – Desempenho da rede SSD Inception v2 utilizando imagens com índice IVV.

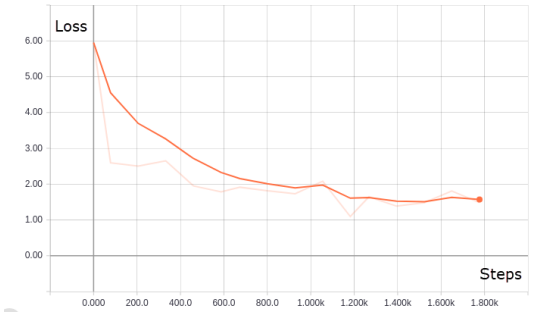
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
33,0	29	1	1	1	7864	0,0 %	3,4 %	9 ms

Fonte: Produzido pelo autor.

4.1.3.2 R-CNN ResNet 101

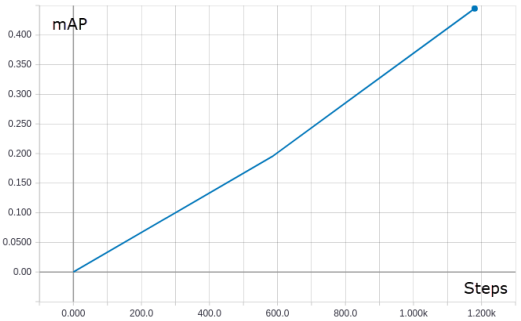
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 55 e 56.

Figura 55 – Evolução do erro total - R-CNN ResNet 101 - IVV



Fonte: Produzido pelo autor.

Figura 56 – Evolução do *mAP* - R-CNN ResNet 101 - IVV

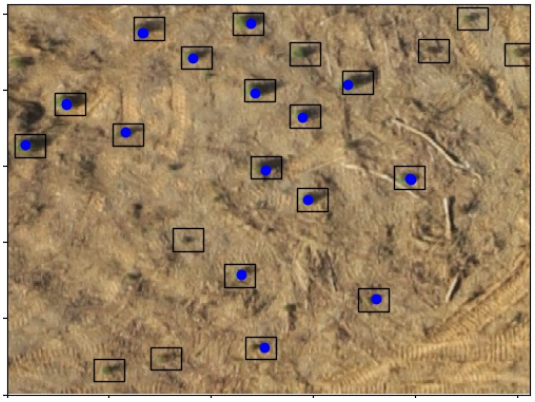


Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,4444$.

A Figura 57 e a Figura 58 ilustram a saída da rede R-CNN ResNet 101 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 57 – Detecção exemplo 1 - R-CNN ResNet 101 - IVV.



Fonte: Produzido pelo autor.

Figura 58 – Detecção exemplo 2 - R-CNN ResNet 101 - IVV.

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 11.

Tabela 11 – Desempenho da rede R-CNN ResNet 101 utilizando imagens com índice IVV.

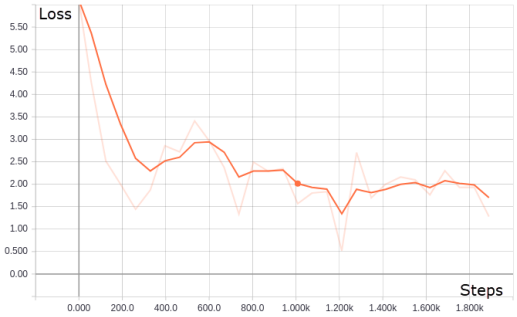
L (%)	**TD**	**VP**	**VPR**	**FN**	**FP**	**TD (%)**	**TE (%)**	**Tempo (ms)**
12,0	6866	6249	5708	342	2157	72,6 %	5,0 %	573 ms

Fonte: Produzido pelo autor.

4.1.3.3 R-CNN Inception ResNet v2

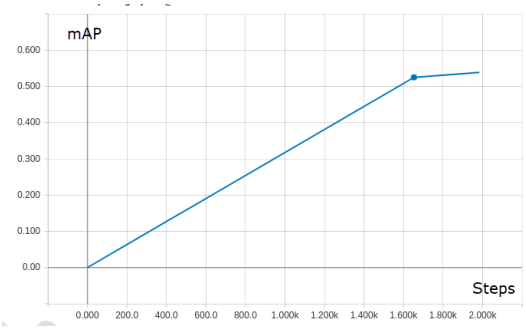
A evolução da rede durante o treinamento pode ser visualizada nas Figuras 59 e 60.

Figura 59 – Evolução do erro total - R-CNN ResNet Inception V2 - IVV



Fonte: Produzido pelo autor.

Figura 60 – Evolução do mAP - R-CNN ResNet Inception V2 - IVV

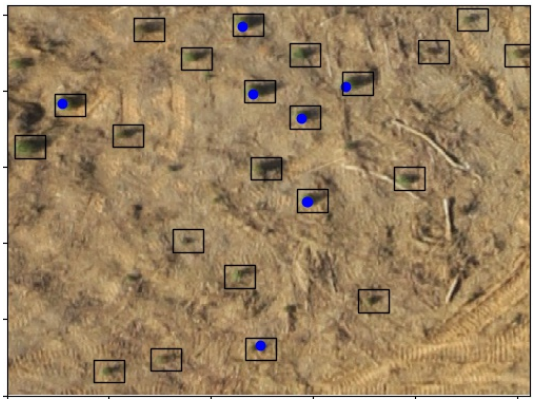


Fonte: Produzido pelo autor.

O modelo obtido no final do treinamento obteve uma precisão média de $mAP = 0,4782$.

A Figura 61 e a Figura 61 ilustram a saída da rede R-CNN ResNet Inception V2 sobreposta na imagem RGB original. Nas figuras, os pontos azuis são os indivíduos detectados pela rede e o quadrado é a marcação de onde realmente existem árvores plantadas.

Figura 61 – Detecção exemplo 1 - R-CNN ResNet Inception V2 - IVV.



Fonte: Produzido pelo autor.

Figura 62 – Detecção exemplo 2 - R-CNN ResNet Inception V2 - IVV.

Fonte: Produzido pelo autor.

Utilizando nossa metodologia de avaliação através de um *script* rodando em *Python*, obtivemos os valores mostrados na Tabela 12.

Tabela 12 – Desempenho da rede R-CNN ResNet Inception V2 utilizando imagens com índice IVV.

L (%)	TD	VP	VPR	FN	FP	TD (%)	TE (%)	Tempo (ms)
3,8	12114	11149	6068	574	1797	77,2 %	4,7 %	1095 ms

Fonte: Produzido pelo autor.

4.2 Comparação

A Tabela 13 mostra as principais métricas de cada rede de forma a possibilitar uma comparação direta de desempenho.

Tabela 13 – Comparação de taxas de erro e acerto entre as redes

Tipo	Rede	L (%)	VP	VPR	FP	FN	TD (%)	TE (%)	Tempo (ms)
RGB	SSD Inception	25,5	7887	7370	430	495	93,7	5,0	81
	R-CNN Resnet	31,0	8249	7471	452	394	95,0	5,0	578
	R-CNN Inception	18,9	8991	7468	496	397	95,0	5,0	1102
IFV	SSD Inception	27,3	9	9	1	7856	0,1	4,3	6
	R-CNN Resnet	24,0	2810	2748	157	5117	34,9	5,0	570
	R-CNN Inception	1,5	7214	4474	368	3391	56,9	4,7	1101
IVV	SSD Inception	27,3	9	9	1	1	0,0	3,4	9
	R-CNN Resnet	12,0	6249	5708	342	2157	72,6	5,0	573
	R-CNN Inception	3,8	11149	6068	574	1797	77,2	4,7	1095

Podemos verificar que as redes com melhor desempenho utilizaram imagens RGB sem aplicação de nenhum filtro, isso ocorre pois as redes utilizadas neste trabalho possuem camadas iniciais de filtragem que se ajustam melhor aos dados do treinamento para filtrar apenas informação útil.

As redes treinadas com os índices IFV e IVV não obtiveram resultados satisfatórios de desempenho e não foram consideradas para o restante da análise de resultados.

A melhor rede no quesito de precisão foi a R-CNN Resnet 101, que obteve uma taxa de detecção de 95% com taxa de erro de 5%, com um tempo de execução de 578 milissegundos por imagem.

No quesito velocidade, a melhor rede foi a SSD Inception V2, que obteve uma taxa de detecção de 93,7% com taxa de erro de 5% e o tempo de execução de 81 milissegundos. Este resultado já era esperado, pois tal rede foi projetada com objetivo de ser extremamente rápida e leve.

A rede R-CNN Resnet Inception V2 obteve resultados semelhantes à rede R-CNN Resnet 101, porém, com um limiar de detecção inferior (18, 9%), que pode elevar a taxa de erros em ouos datasets, e com tempo 90% maior.

5 Conclusão

Com os resultados obtidos, foi possível validar a possibilidade de utilização de tecnologias de visão computacional e *machine learning* para a aplicação de contagem e detecção de plantas. Os modelos treinados ofereceram uma ótima precisão e taxas de erros próximas ao padrão de trabalho da indústria (97% de precisão e erro menor que 3%), mas ainda não utilizáveis comercialmente. Através de melhorias no processo de treinamento e melhores técnicas de pré-processamento e análise, acredita-se que é possível obter os resultados desejados.

O modelo com maior precisão, o R-CNN Resnet 101, pode ser utilizado para fins comerciais com pequenas melhorias de treinamento e processamento. Porém ele apenas pode ser utilizado através de pós-processamento de images, visto que é um modelo que demanda *hardware* de alto desempenho e tem alto consumo de recursos.

A segunda melhor precisão foi obtida pela rede SSD Inception V2, que obteve resultados apenas $1,3\%$ piores, porém em sete vezes menos tempo. Esta rede consegue rodar com poucos recursos e de maneira rápida, que possibilita embarcar o processamento e a detecção de árvores na própria aeronave, desde que seus resultados sejam otimizados.

Por fim, este trabalho demonstrou a viabilidade de aplicação de técnicas de *machine learning* para fornecer produtos e análises no setor de silvicultura. Esta prova de conceito é importante, pois as tecnologias utilizadas auxiliam no desenvolvimento acelerado da indústria e ajudam a manter um nível de inovação à um passo rápido.

6 Pontos à melhorar e futuros trabalhos

Nessa seção são delineadas possíveis melhorias a serem desenvolvidas e estudadas para aumentar as taxas de detecção e contagem e reduzir o tempo de processamento. Com estas melhorias, pretende-se obter uma taxa de precisão de $TD = 98\%$ enquanto reduzida a taxa de erro para $TD < 3\%$.

6.1 Melhorar pré e pós-processamento dos resultados

Adicionar camadas mais sofisticadas de processamento nas entradas e saídas da rede.

Com os dados de entrada, pode-se usar um algorítimo determinístico para detectar a probabilidade de haver vegetação em um determinado local e utilizar essa informação como uma entrada da rede. Também pode-se usar uma combinação entre filtros de cor e redução da qualidade da imagem para gerar camadas adicionais de informação que podem levar à uma melhor detecção.

Com os dados de saída, deve ser projetado um algoritmo para aglomerar pontos próximos com base na distância de forma a eliminar duplas detecções pelo sistema. Outro problema existente que pode ser resolvido através de pós-processamento é a dupla detecção de indivíduos em duas bordas diferentes das imagens.

6.2 Realizar estudo com câmeras multi-espectrais

Realizar o mesmo estudo deste trabalho, porém utilizando câmeras com mais bandas espectrais, de forma a captar características da folha não presentes na radiação visível. Comparar e avaliar os impactos das bandas de infravermelho próximo e infravermelho médio na detecção e contagem de indivíduos.

6.3 Diversificar as áreas de estudo com culturas diversas

Treinar e avaliar as redes utilizadas para contagem e detecção de múltiplas culturas, possibilitando uma mesma rede de contar e detectar mais de um tipo de plantação por ortomosaico, ou até mesmo calcular a área de plantio e demarcar os talhões da plantação.

6.4 Construir um modelo de detecção específico

Utilizando como base os modelos SSD e R-CNN, projetar uma rede específica para a detecção e contagem de indivíduos em plantações de diferentes culturas.

O trabalho pode ser iniciado à partir do treinamento completo (de todas as camadas) de modelos já existentes e então modificar camadas para otimizar o funcionamento da rede. Outra maneira é iniciar à partir de um modelo vazio e ir adicionando camadas de forma à chegar no resultado desejado.

Referências

ABREU, L. M. C. Karla Maria Pedra de. Sensoriamento remoto aplicado ao estudo da vegetação com ênfase em índice de vegetação e métricas da paisagem. *VÉRTICES*, p. 173–198, jan 2014. Citado na página 27.

AKERKAR, R. *Introduction to artificial intelligence*. [S.l.]: PHI Learning Pvt. Ltd., 2014. Citado na página 34.

Amazon Mturk. *Amazon Mechanical Turk*. 2017. <https://www.mturk.com/>. Acesso em: 17 de novembro de 2017. Citado na página 43.

ANTUNES, A. F. B. Fundamentos de sensoriamento remoto em ambiente de geoprocessamento. *Apostila. CIEG. UFPR*, 2001. Citado na página 29.

ARAUJO, L. C. *Configuração*: uma perspectiva de Arquitetura da Informação da Escola de Brasília. Dissertação (Mestrado) — Universidade de Brasília, Brasília, mar. 2012. Citado na página 41.

BALLARD, D. H.; BROWN, C. M. Computer vision, 1982. *Prenice-Hall, Englewood Cliffs, NJ*, 1982. Citado na página 33.

BRANDELERO, C.; ANTUNES, M.; GIOTTO, E. Silvicultura de precisão: mapeamento, inventário e geoestatística. *Revista Geomática*, v. 2, n. 1, p. 15–25, 2007. Citado na página 32.

CHRISTOPHER, D. M.; PRABHAKAR, R.; HINRICH, S. Introduction to information retrieval. *An Introduction To Information Retrieval*, v. 151, p. 177, 2008. Citado na página 53.

CIFLORESTAS. *Mitos e verdades sobre o plantio de eucalipto*. 2011. <http://www.ciflorestas.com.br/conteudo.php?id=6107>. Acesso em: 12 de novembro de 2017. Citado na página 33.

ERTEL, W. *Introduction to artificial intelligence*. [S.l.]: Springer Science & Business Media, 2011. Citado na página 34.

FIGUEIREDO, D. Conceitos básicos de sensoriamento remoto. *Companhia Nacional de Abastecimento-CONAB. Brasília-DF*, 2005. Citado na página 29.

FORD-ROBERTSON, F. *Terminology of forest science, technology, practice*. [S.l.]: Washington, DC: Society of American Foresters, 1971. Citado na página 32.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 38.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587. Citado na página 38.

GOBRON, N. et al. Advanced vegetation indices optimized for up-coming sensors: Design, performance, and applications. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 38, n. 6, p. 2489–2505, 2000. Citado na página 31.

HUANG, J. et al. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*, 2016. Citado 2 vezes nas páginas 49 e 50.

HUANG, J. et al. *Tensorflow detection model zoo*. 2016. <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>. Acesso em: 12 de novembro de 2017. Citado na página 51.

HUANG, T. Computer vision: Evolution and promise. Cern, 1996. Citado na página 33.

JACKSON, R. D.; HUETE, A. R. Interpreting vegetation indices. *Preventive veterinary medicine*, Elsevier, v. 11, n. 3-4, p. 185–200, 1991. Citado na página 30.

LAU, F. *Speed/accuracy trade-offs for modern convolutional object detectors*. 2017. <https://medium.com/@phelixlau/speed-accuracy-trade-offs-for-modern-convolutional-object-detectors-bbad4e4e0718>. Acesso em: 17 de novembro de 2017. Citado 2 vezes nas páginas 36 e 38.

LECUN, Y. et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015. Citado na página 35.

LIMA, C. G. d. R. et al. Atributos físico-químicos de um latossolo do cerrado brasileiro e sua relação com características dendrométricas do eucalipto. *Revista Brasileira de Ciência do Solo*, Sociedade Brasileira de Ciência do Solo, v. 34, n. 1, 2010. Citado na página 32.

LISA Lab. *Convolutional Neural Networks (LeNet)*. 2017. <http://deeplearning.net/tutorial/lenet.html>. Acesso em: 17 de novembro de 2017. Citado na página 35.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37. Citado 2 vezes nas páginas 36 e 37.

LOUHAICHI, M.; BORMAN, M. M.; JOHNSON, D. E. Spatially located platform and aerial photography for documentation of grazing impacts on wheat. *Geocarto International*, Taylor & Francis, v. 16, n. 1, p. 65–70, 2001. Citado na página 30.

MARCHIÜRI, J.; SOBRAL, M. *Dendrologia das Angiospermas*. [S.l.]: Myrtales. Santa Maria: Editoraufsm, 1997. Citado na página 32.

MENESES, P. R.; NETTO, J. d. S. M. *Sensoriamento remoto: reflectância dos alvos naturais*. [S.l.]: Editora Universidade de Brasilia; Planaltina: Embrapa Cerrados, 2001. Citado na página 29.

MOREIRA, M. A. *Fundamentos do sensoriamento remoto e metodologias de aplicação*. [S.l.]: UFV, 2005. Citado na página 28.

Planetary Habitability Laboratory - University of Puerto Rico at Arecibo. *Visible Vegetation Index (VVI)*. 2017. <http://phl.upr.edu/projects/visible-vegetation-index-vvi>. Acesso em: 17 de novembro de 2017. Citado na página 31.

PONZONI, F. J.; SHIMABUKURO, Y. E.; KUPLICH, T. M. *Sensoriamento remoto no estudo da vegetação*. [S.l.]: Parêntese São José dos Campos, 2007. Citado na página 27.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 91–99. Citado na página 39.

RUSSELL, S.; NORVIG, P.; INTELLIGENCE, A. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, v. 25, p. 27, 1995. Citado na página 34.

SILVA, G. C. da. *Repoitório oficial do TCC*. 2017. <https://github.com/giovannicimolin/TCC>. Acesso em: 17 de novembro de 2017. Citado na página 43.

VETTORAZZI, C. A.; FERRAZ, S. F. d. B. Silvicultura de precisão: uma nova perspectiva para o gerenciamento de atividades florestais. *Agricultura de precisão. Viçosa: Ed. Da UFV*, p. 65–75, 2000. Citado na página 32.

Wikipédia. *Espectro eletromagnético*. 2017. <https://pt.wikipedia.org/wiki/Espectro_eletromagn%C3%A9tico>. Acesso em: 17 de novembro de 2017. Citado na página 28.

Anexos

ANEXO A – SSD Inception v2

```
# SSD with Inception v2 configuration for MSCOCO Dataset.
model {
  ssd {
    num_classes: 1
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
        reduce_boxes_in_lowest_layer: true
      }
```

```
}
image_resizer {
  fixed_shape_resizer {
    height: 512
    width: 512
  }
}
box_predictor {
  convolutional_box_predictor {
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    use_dropout: false
    dropout_keep_probability: 0.7
    kernel_size: 3
    box_code_size: 4
    apply_sigmoid_to_scores: false
    conv_hyperparams {
      activation: RELU_6,
      regularizer {
        l2_regularizer {
          weight: 0.00004
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
    }
  }
}
feature_extractor {
  type: 'ssd_inception_v2'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
```

```
      regularizer {
        l2_regularizer {
          weight: 0.00004
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
      batch_norm {
        train: true,
        scale: true,
        center: true,
        decay: 0.9997,
        epsilon: 0.001,
      }
    }
  }
  loss {
    classification_loss {
      weighted_sigmoid {
        anchorwise_output: true
      }
    }
    localization_loss {
      weighted_smooth_l1 {
        anchorwise_output: true
      }
    }
    hard_example_miner {
      num_hard_examples: 3000
      iou_threshold: 0.98
      loss_type: CLASSIFICATION
      max_negatives_per_positive: 3
      min_negatives_per_image: 0
    }
    classification_weight: 1.0
```

```
      localization_weight: 1.0
    }
    normalize_loss_by_num_matches: true
    post_processing {
      batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.5
        max_detections_per_class: 200
        max_total_detections: 200
      }
      score_converter: SIGMOID
    }
  }
}

train_config: {
  batch_size: 24
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 2000
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
  fine_tune_checkpoint: "model/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 2000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
```

```
    ssd_random_crop {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/objects.pbtxt"
}

eval_config: {
  num_examples: 8000
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/test.record"
  }
  label_map_path: "data/objects.pbtxt"
  shuffle: false
  num_readers: 1
  num_epochs: 1
}
```

ANEXO B – Faster R-CNN Resnet 101

```
# Faster R-CNN with Resnet-101 (v1) configuration for MSCOCO Dataset.
model {
  faster_rcnn {
    num_classes: 1
    image_resizer {
      fixed_shape_resizer {
        height: 512
        width: 512
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.5
```

```
  first_stage_max_proposals: 300
  first_stage_localization_loss_weight: 2.0
  first_stage_objectness_loss_weight: 1.0
  initial_crop_size: 22
  maxpool_kernel_size: 2
  maxpool_stride: 2
  second_stage_box_predictor {
    mask_rcnn_box_predictor {
      use_dropout: false
      dropout_keep_probability: 1.0
      fc_hyperparams {
        op: FC
        regularizer {
          l2_regularizer {
            weight: 0.0
          }
        }
        initializer {
          variance_scaling_initializer {
            factor: 1.0
            uniform: true
            mode: FAN_AVG
          }
        }
      }
    }
  }
  second_stage_post_processing {
    batch_non_max_suppression {
      score_threshold: 0.0
      iou_threshold: 0.5
      max_detections_per_class: 300
      max_total_detections: 300
    }
    score_converter: SOFTMAX
  }
  second_stage_localization_loss_weight: 2.0
  second_stage_classification_loss_weight: 1.0
}
```

```
}

train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 2000
            learning_rate: .00003
          }
          schedule {
            step: 4000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "model/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 2000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
```

```
    input_path: "data/train.record"
  }
  label_map_path: "data/objects.pbtxt"
}

eval_config: {
  num_examples: 8000
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/test.record"
  }
  label_map_path: "data/objects.pbtxt"
  shuffle: false
  num_readers: 1
  num_epochs: 1
}
```

ANEXO C – Faster R-CNN Inception Resnet v2

```
# Faster R-CNN with Inception Resnet v2, Atrous version
model {
  faster_rcnn {
    num_classes: 1
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 512
        max_dimension: 512
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_resnet_v2'
      first_stage_features_stride: 8
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 8
        width_stride: 8
      }
    }
    first_stage_atrous_rate: 2
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
```

```
}
first_stage_nms_score_threshold: 0.0
first_stage_nms_iou_threshold: 0.7
first_stage_max_proposals: 300
first_stage_localization_loss_weight: 2.0
first_stage_objectness_loss_weight: 1.0
initial_crop_size: 17
maxpool_kernel_size: 1
maxpool_stride: 1
second_stage_box_predictor {
  mask_rcnn_box_predictor {
    use_dropout: false
    dropout_keep_probability: 1.0
    fc_hyperparams {
      op: FC
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        variance_scaling_initializer {
          factor: 1.0
          uniform: true
          mode: FAN_AVG
        }
      }
    }
  }
}
second_stage_post_processing {
  batch_non_max_suppression {
    score_threshold: 0.0
    iou_threshold: 0.5
    max_detections_per_class: 150
    max_total_detections: 150
  }
  score_converter: SOFTMAX
}
```

```
    second_stage_localization_loss_weight: 2.0
    second_stage_classification_loss_weight: 1.0
  }
}

train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 1000
            learning_rate: .00003
          }
          schedule {
            step: 2000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "faster_rcnn_inception_resnet_v2_atrous_coco_2017_
  from_detection_checkpoint: true
  num_steps: 2000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
```

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/objects.pbtxt"
}

eval_config: {
  num_examples: 8000
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/test.record"
  }
  label_map_path: "data/objects.pbtxt"
  shuffle: false
  num_readers: 1
  num_epochs: 1
}
```