UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

WILLIAN OLIVEIRA PIRES

RECONHECIMENTO DE ESPÉCIES FLORESTAIS PELA FOLHA, UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2018

WILLIAN OLIVEIRA PIRES

RECONHECIMENTO DE ESPÉCIES FLORESTAIS PELA FOLHA, UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Computação".

Orientador: Prof. Dr. Pedro Luiz de Paula Filho

Co-orientador: Prof. Dr. Arnaldo Candido Junior

MEDIANEIRA

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

TERMO DE APROVAÇÃO

RECONHECIMENTO DE ESPÉCIES FLORESTAIS PELA FOLHA, UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Por

WILLIAN OLIVEIRA PIRES

Este Trabalho de Conclusão de Curso foi apresentado às 08:30h do dia 21 de novembro de 2018 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Msc. Jorge Aikes Junior
UTFPR - Câmpus Medianeira

Prof. Dr. Paulo Lopes de Menezes
UTFPR - Câmpus Medianeira

Prof. Dr. Pedro Luiz de Paula Filho
UTFPR - Câmpus Medianeira

Prof. Dr. Arnaldo Candido Junior
UTFPR - Câmpus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

PIRES, Willian Oliveira. RECONHECIMENTO DE ESPÉCIES FLORESTAIS PELA FOLHA, UTILIZANDO REDES NEURAIS CONVOLUCIONAIS. 96 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2018.

Buscando o controle e a preservação de espécies de plantas, a etapa de identificação se torna uma atividade primordial. O processo de identificação é realizado principalmente por botânicos, nele a identificação ocorre com a comparação de espécimes já conhecidas ou com o auxílio de livros, manuais ou chaves de identificação. As redes neurais artificiais têm se mostrado uma boa acurácia para problemas de classificação, por isso o projeto utilizou redes neurais convolucionais para classificar espécies, por imagem da folha. No projeto foram coletadas 29 espécies na Universidade Tecnológica Federal do Paraná no campus Medianeira. Foram utilizadas dois modelos de rede, YOLO e Googlenet, devido a diferença na manipulação da imagem e visando realizar uma comparação entre elas. Os modelos de YOLO e Googlenet atingiram o reconhecimento de 86,2% e 90,3%, respectivamente.

Palavras-chave: deep learning, folha, reconhecimento

ABSTRACT

PIRES, Willian Oliveira. RECOGNITION OF FOREST SPECIES BY LEAF, USING CONVOLUTIONAL NEURAL NETWORKS. 96 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2018.

Seeking the control and preservation of plant species, the identification stage becomes a primordial activity. The identification process is carried out mainly by botanists, where the identification occurs with the comparison of already known specimens or with the aid of books, manuals or identification keys. Artificial neural networks have been shown to be a good accuracy for classification problems, so the project used convolutional neural networks to classify species by leaf image. In the project, 29 species were collected at the Universidade Tecnológica Federal do Paraná at the Medianeira campus. Two network models, YOLO and Googlenet, were used due to the difference in image manipulation and a comparison between them. The YOLO and Googlenet models achieved recognition of 86.2 % and 90.3 %, respectively.

Keywords: deep learning, sheet, recognition

AGRADECIMENTOS

Agradeço a minha família pelo apoio e o incentivo, mesmo não gostando muito da idéia de estudar longe de São Paulo nunca deixaram de me apoiar. Em especial agradeço ao meu pai Gonçalo Coelho por seus ensinamentos e valores que me ajudaram muito durante essa fase da minha vida. Gostaria de agradecer a todos os professores que tive na universidade, todas as aulas e conversas me ajudaram a não só compreender o conteúdo, como também ter a certeza que escolhi o curso certo. Entre os professores gostaria de agradecer o professor Agostinho Zanini que tornou esse trabalho possível, já que teve a gentiliza de me apresentar as plantas que ele cuida na universidade, e permitir a coleta de alguns exemplares. Espero que algum dia encontre algo que me motive igual o professor Agostinho em relação a suas plantas, nunca vou esquecer o prazer que ele tinha de me apresentar cada espécie, muito obrigado professor. Agradeço ao meu orientador Pedro de Paula que me ajudou muito durante todo curso, graças a ele tive muitas oportunidades na universidade e durante o projeto agradeço a suas dicas e orientações, muito obrigado professor. Agradeço a meu co-orientador Arnaldo Candido que graças a suas aulas em fundamentos de sistemas inteligentes, tornaram possível a conclusão do trabalho. Gostaria de agradecer aos amigos que fiz nessa fase da minha vida, agradeço por suas conversas e trabalhos feitos juntos. Entre eles estão alguns da minha turma (alias a melhor): Paulo Vitor, Leandro Salles, Tarlon Gomes, Lucas Menegazzi, Alexandre Jasper, Marco Yuri e Otavio Nascimento muito obrigado aprendi muito com vocês galera. Para finalizar gostaria de agradecer o "bonde" LGEJ (Lucas, Gabriel, Eduardo e Julio) que são os principais responsáveis de eu ter chegado até aqui, estiveram comigo nos momentos bons (pedrada) e ruins (pedrada) durante esses quatro anos, aprendi muito com cada um, sou muito grato de ter conhecido vocês.

“Se eu tivesse uma hora pra resolver um problema e minha vida dependesse dessa solução, eu passaria 55 minutos definindo a pergunta certa a se fazer” **(Albert Einstein)**.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

FP	Falso positivo
FN	Falso negativo
GB	GigaByte
IA	Inteligência Artificial
IoU	intersecção entre a predição da caixa e *ground truth*
Pc	Probabilidade da classe
PDI	Processamento Digital de Imagem
RAM	Random Access Memory
ReLU	Rectified Linear Unit
TB	Rectified Linear Unit
Vc	Valor de confiança
Vcc	Valor da caixa delimitadora
VN	Verdadeiro negativo
VP	Verdadeiro positivo

SUMÁRIO

1 INTRODUÇÃO

O Brasil é um pais que possui uma grande riqueza natural, isso devido à variedade de formações vegetais e ecossistemas, estudos apontam que existem cerca de 7.880 espécies arbóreas catalogadas (FOOD; NATIONS, 2005). Para que se tenha um controle sobre essas espécies, a etapa de identificação se torna uma atividade primordial para o manejo florestal. O processo de identificação é realizado principalmente pelos botânicos, nele a identificação ocorre com a comparação de espécimes já conhecidas ou com o auxílio de livros, manuais ou chaves de identificação. Esse processo é uma etapa importante em que o profissional envolvido deve saber lidar com os mais variados tipos de espécies, desde casos simples como identificar plantas que possuem órgão reprodutivos (flores e frutos) a mais complexos, em que se deve saber quais atributos morfológicos observar pode evitar equívocos. Para não profissionais esse processo pode se tornar demorado e propenso a erro, sendo que pode ser muito benéfico tornar essa atividade de identificação mais acessível a eles. Já que conhecer o nome popular ou cientifico, acaba trazendo familiaridade, valorização e estimula a preservação das espécies, um ponto muito importante nos dias atuais. Com o auxílio de outras áreas, como a computação, com técnicas de processamento de imagem e reconhecimento de padrões, surgiram novas formas de identificação de espécies, que podem auxiliar tanto profissionais como não profissionais.

1.1 JUSTIFICATIVA

Para que uma pessoa se torne um profissional habilitado para fazer a identificação de plantas, é necessário estudo das características morfológicas das espécies. Muitos botânicos fazem uso das chaves de identificação para classificar e identificar espécies, em que basicamente o profissional compara a espécie com um conjunto de características. A escolha dessas características dependendo do caso pode não ser tão trivial. Para auxiliar nesse problema surgiram estudos fazendo uso de técnicas de processamento de imagem e reconhecimento de

padrões, usando diferentes elementos das espécies para o reconhecimento. Nesta proposta busca-se fazer uso da folha para realizar o reconhecimento, um método menos invasivo, considerando também que a folha contém uma grande variedade de característica como cor, textura, formato e nervura. A coleta das folhas para a base de imagens será visando a menor danificação das espécies, utilizando a frente e o verso da folha, aumentando a variedade de características sem danificar a espécie.

1.2 OBJETIVOS GERAL E ESPECÍFICOS

Esse trabalho tem como objetivo avaliar o desempenho de redes neurais convolucionais para reconhecimento de espécies florestais. Esse objetivo principal pode ser dividido nos seguintes objetivos específicos:

- Coletar imagens de folhas para o projeto;
- Realizar um comparativo entre as Redes YOLO (*You Only Look Once*) e Googlenet perante a seus resultados;
- Desenvolver algoritmo para aumentar base de imagens;
- Desenvolver algoritmo para automatizar os recortes do YOLO.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Esse documento será organizado da seguinte forma. O Capítulo 2 apresentará inicialmente um estudo sobre chaves de identificação utilizadas pelos botânicos, como forma de verificação de atributos. Em seguida será apresentado uma seção sobre técnicas de processamento de imagem importantes para o desenvolvimento do projeto.

Após a seção de processamento de imagem será apresentado o *deep learning* passando por inteligência artificial, aprendizado de máquina até redes neurais. Por fim são apresentados trabalhos correlatos. A metodologia utilizada se encontra no Capítulo 3, nele são descritas as ferramentas e a metodologia empregada. No Capítulo 4 são apresentados os resultados e discursão sobre o trabalho. No Capítulo 5 encontra-se a conclusão da proposta do trabalho de conclusão de curso.

2 LEVANTAMENTO BIBLIOGRÁFICO

Nesta seção será descrito o estado da arte do tema escolhido. Primeiramente será realizada uma breve descrição das chaves dicotômicas utilizadas por botânicos, em seguida serão apresentadas técnicas de processamento de imagem, importantes para entender o desenvolvimento do projeto. A seção de processamento de imagem será dívida em segmentação, limiarização e segmentação por região. Após isso será apresentado um breve histórico sobre inteligência artificial passando pelo aprendizado de máquina, redes neurais até chegar ao *deep learning*. Continuando a seção *deep learning* serão apresentados dois exemplos de redes neurais convolucionais YOLO e Googlenet, em seguida, serão apresentados os trabalhos correlatos recentes.

2.1 BOTÂNICA

A botânica é a ciência que estuda a vida de plantas e algas, analisando crescimento, reprodução e evolução da vida das plantas (LORENZI, 2013). Sendo feita observando desde unidades muito pequenas, em níveis nanoscópicos ou unidades maiores como o corpo da planta. A utilização de conhecimentos sobre as plantas vem desde a Grécia antiga, tanto Aristóteles quanto Teofrasto se propuseram a identificar plantas e descrevê-las. Por causa de suas contribuições, Teofrasto ficou conhecido como o "pai da botânica" por causa de seus dois trabalhos sobreviventes em estudos de plantas, Historia plantarum e De causis plantarum (FERRI, 1970). No final do século XVI foi inventado o microscópio, o que permitiu um estudo mais preciso das estruturas morfológicas das plantas. Desde então houve um maior interesse em estudos nessa área, em 1893 foi fundada a BSA (Sociedade Botânica da América) uma das principais organizações botânicas com membros em todo mundo (LORENZI, 2013).

Atualmente a botânica se dividir em diferentes disciplinas, sendo as principais:

- Fitopatologia ou patologia vegetal: É a disciplina que estuda as doenças, os danos e as

alterações do funcionamento normal das plantas, qualquer que seja a sua origem (animal, vegetal, parasitária ou infecciosa) abrangendo todas as suas etapas, desde o diagnóstico até o tratamento e controle. A historia da Fitopatologia é dividido em cinco períodos: Período Místico, Período da Predisposição, Período Etiológico, Período Ecológico e Período Fisiológico (LORENZI, 2013).

- Ecologia vegetal: Essa disciplina estuda o modo como as plantas interagem com o meio ambiente, como reage a mudanças ecológicas, como as mudanças climáticas. Como criar ou alterar condições para as plantas sobreviver ao meio (FERRI, 1970). São base de conhecimento ecofisiologia vegetal, a ecologia populacional vegetal, a ecologia comunitária, a ecologia ecossistemática, a ecologia da paisagem e a ecologia global. Ela tem origem tanto na geografia das plantas, como em estudos sobre o relacionamento entre plantas individuais e o seu meio envolvente (FERRI, 1970).
- Paleobotânica: Estuda plantas extintas ou plantas fossilizadas recuperadas, além disso é estudado algas, bactérias, fungos e líquens. Essa disciplina tem sido muito utilizada para entender as mudanças climáticas do passado. A paleobotânica tem como foco principal o estudo de fósseis do Reino Plantae, mas também estão inclusos fósseis dos Reinos Monera (bactérias e cianobactérias), Fungi (fungos e afins) e alguns do Reino Protista (algas), pois estes organismos eram classificados como pertencentes ao Reino Plantae (LORENZI, 2013).
- Arqueobotânica: Nessa disciplina estuda como era utilizada plantas por pessoas no passado, casos como: disseminação da agricultura, drenagem de terras úmidas, irrigação e outras formas de engenharia ecológica. Além do uso das plantas pelo antepassados (LORENZI, 2013);
- Botânica forense: Nessa disciplina utiliza-se plantas, sementes ou qualquer vestígios botânicos visando obter provas para um crime, comprovar o testemunho ou acusação. Os pólens se encontram em grande quantidade no ambiente sendo muito resistentes à deterioração química, mecânica e biológica. Esse fato permite encontrar com precisão a localidade de deposição do corpo, e identificar suspeitos e objetos no local do crime. O que facilita e dá comprovação científica ao trabalho do perito criminal (LORENZI, 2013).
- Dendrologia: É uma disciplina da botânica que estuda plantas lenhosas, principalmente árvores e arbustos e suas madeiras. Entre os materiais utilizados para o estudo das espécies estão as chaves dicotômicas (MARCHIORI, 1995).

2.2 CHAVES DICOTÔMICAS

Visando o melhor entendimento será apresentado uma tabela com algumas características e elementos presentes nas folhas (MARCHIORI, 1995):

Tabela 1 – Tabela com características e elementos da folha

Nome Cientifico	Descrição
Nervação	Estruturas de espessamento das folhas, tecido vascular
Pilosidade	Estruturas como pelos e escamas na superficie da folha
Filotaxia	Diz respeito ao arranjo das folhas ao longo do eixo caulinar
Eixo Caulinar	Estrutura de sustentação da folhas, flores e frutos
Estipulas	Estrutura com a forma de escama localizada no caule de folhas
Mudança da Base	Estrutura localizada próximo ao pecıolo da folha, se dividindo em nove tipos
Mudança do Ápice	A ponta da folha que varia entre nove tipos
Folha Simples	Folhas que tem um só um folıolo
Folha Composta	Folhas com mais de um folıolo
Folıolo	São subdivisões da folha, ligadas pelo pecıolo à raque
Pecıolo	Parte estreita localizado entre o folıolo e o caule
Raque	Eixo central que liga o Pecıolo
Julga	A folha composta em si, contêm mais de um folıolo
Gema Auxiliar	Diz respeito estrutura de junção dos ramos com o pecıolo de uma planta

Fonte: Autoria própria

Para realizar o processo de identificação e classificação de espécies, os botânicos se baseiam na taxonomia do vegetal, com essas características determinadas procura-se agrupar as espécies por sua semelhança morfológica e por ligações de parentesco genético (PINHEIRO, 2000). Todo esse processo acabou gerando uma disciplina dentro da botânica denominada dendrologia, que estuda a identificação, distribuição e classificação das plantas lenhosas (MARCHIORI, 1995). Ela abrange muitas características, como os tipos de raízes, porte da árvore, pilosidade e o fuste, além de elementos de diagnóstico (cor, textura e estrutura). O fuste é a parte do tronco livre de ramificações que pode ter diferentes tipos, formas e bases (Figura 1). A Figura 1 mostra a variedade de característica que a madeira apresenta fora cor e textura, entre elas tem-se os tipos de troncos que podem ser cilíndricos, acanalados, elípticos e irregulares.

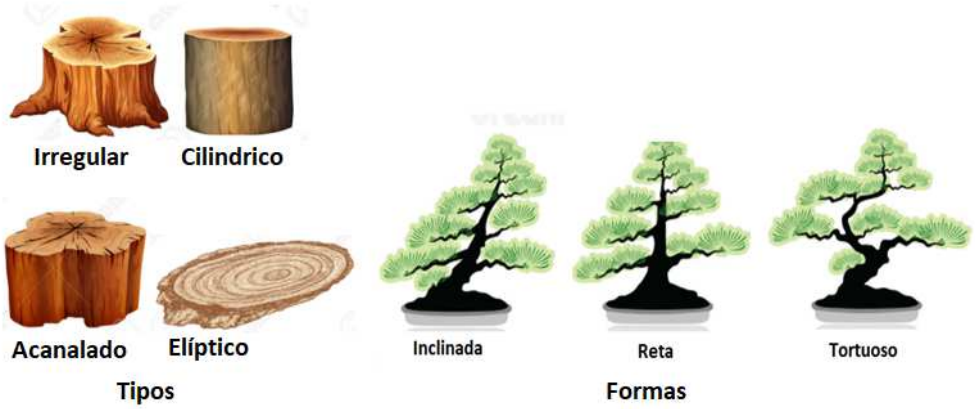

Figura 1 – Tipos, formas e bases de troncos
Fonte: Marchiori (1995)

Como o projeto visa realizar a identificação de espécies pela folha, serão objeto de estudo suas características. A variedade das folhas começa por sua divisão já que se tem folhas simples que apresentam apenas uma lâmina foliar após a gema axilar como mostrado na Figura 2.

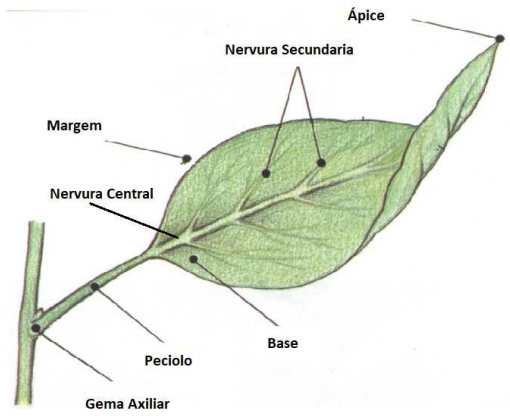

Figura 2 – Folha Simples
Fonte: Marchiori (1995)

Tem-se também folhas compostas as quais se apresentam mais de um folíolo como mostrado a Figura 3. Esse conhecimento é importante já que em alguns casos pode haver uma confusão sobre o que pode ser considerado uma folha, no caso da espécie *Tabebuia chrysotricha* (Ipê-Amarelo) é uma folha composta digitada, então a julga deve ser levada em conta para fazer a identificação e não apenas um folíolos (Figura 3).

Figura 3 – Folha Composta

Fonte: Marchiori (1995)

Tem-se ainda elementos importantes da folha apresentados na Figura 2 que geram uma grande variedade de espécies, sendo eles a forma da folha, ápice (ponta da folha), base e margem todos atributos que podem ser usados na hora de diferenciar espécies. Tem-se ainda a filotaxia que se dividem em quatro tipos: Alternadas, Oposto Cruzadas, Opostas Dísticas e Verticiladas verificado na Figura 4. Esse atributo que pode ser utilizado antes da coleta da folha já que diz respeito a como as folhas estão organizadas. Outro atributo é a nervação, que se dividem entre: Pinada, Reticulada, Paralelinérvea, Palmatinérvea e Dicotômica, Figura 5.

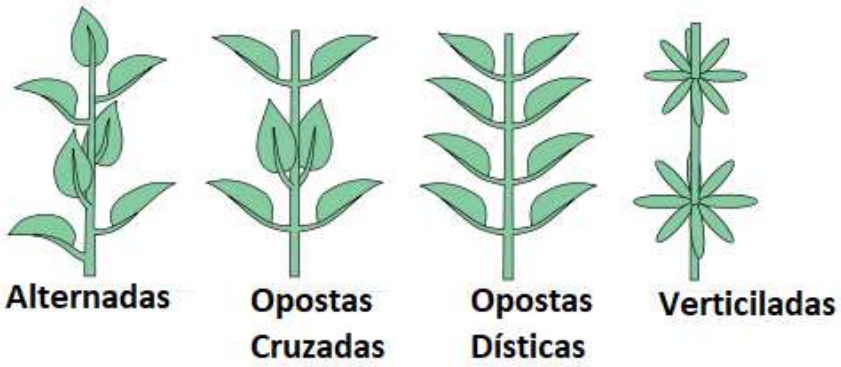

Figura 4 – Filotaxia

Fonte: Marchiori (1995)

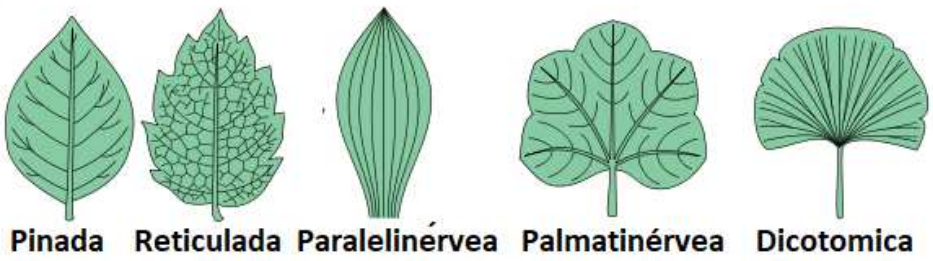

Figura 5 – Nervação
Fonte: Marchiori (1995)

Todas essas características são levadas em conta na hora da identificação das espécies, servindo de base para um método muito comum entre os botânicos que é a utilização de chaves dicotômicas. A chave dicotômica é uma técnica baseada na observação das características das plantas, em que basicamente pesquisadores comparam as características das espécies extraídas do campo com a espécie que se quer identificar, caso atenda uma condição, passar para outra característica até encontrar o nome da espécie (PINHEIRO, 2000). Um exemplo é utilizar a chave para classificar a folha em relação a nervação, em que é utilizado uma folha simples para identificar o tipo de nervação em que ela se encaixa (Figura 10).

Figura 6 – Folha exemplo
Fonte: Odemir et al. (2017)

Com a folha, o especialista fará uso da chave, que pode estar organizada da seguinte forma (versão simples):

Quadro 1 – Chave dicotômica simples

1. Folha com uma só nervura não ramificada..... Uninérvea Folhas com mais de uma nervura..... 2
2. Folhas com mais de uma nervura todas paralelas entre si..... Paralelinérveas Folhas com nervuras não paralelas..... 3
3. Com uma nervura principal, partem nervuras secundárias..... Peninérveas Folhas com várias nervuras principais partindo da base do limbo..... Palminérveas

Fonte: Marchiori (1995)

Observando a folha, o profissional percebe que trata-se de um exemplar com mais de uma nervura, então passa para o passo 2, nota-se que suas nervuras não são paralelas indo para o passo 3 e conclui ser uma folha com nervação do tipo Peninérveas, ela possui uma nervura principal onde partem nervuras secundárias (PINHEIRO, 2000). Esse é um exemplo simples e resumido do uso da chave de identificação, mas em alguns casos a escolha da característica pode não ser tão simples e a identificação envolver mais de uma característica. Um exemplo é a *Xanthosoma taioba* uma hortaliça própria para o consumo de difícil classificação, muito parecida com espécies não indicadas (*Xanthosoma violaceum*) para o consumo, Figura 7.

Figura 7 – A *Xanthosoma taioba* verdadeira esta a direita, à esquerda *Xanthosoma violaceum*

Fonte: Marchiori (1995)

Na Figura 7 fica nítida a dificuldade de classifica-la, já que características comuns para classificar espécies pela folha como formato, base, ápice e nervação acabam não sendo o suficiente. A folha da *Xanthosoma taioba* é cordiforme, lembra a forma do coração, a base é mais larga, com uma reentrância e com os lobos arrendodados (LORENZI, 2013). Por apenas essa descrição a um grande risco de confusão na hora de classifica-la, devido a semelhança de

outras espécies, como visto na Figura 8.

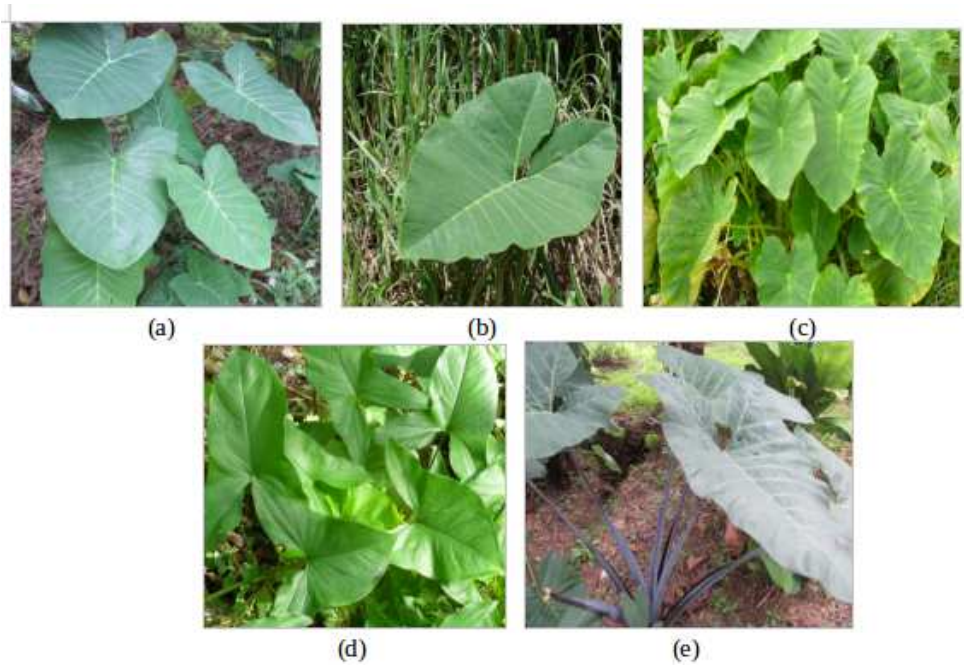

Figura 8 – a - *Xanthosoma taioba*, b - *Xanthosoma robustum*, c - *Colocasia antiquorum*, d - *Syngonium podophyllum*, e - *Xanthosoma violaceum*

Fonte: Adaptado - Marchiori (1995)

Para identificar a espécie *Xanthosoma taioba*, a base é recortada e funde no pecíolo, tem linha circundante na borda e o ápice da folha aponta para baixo Figura 9.

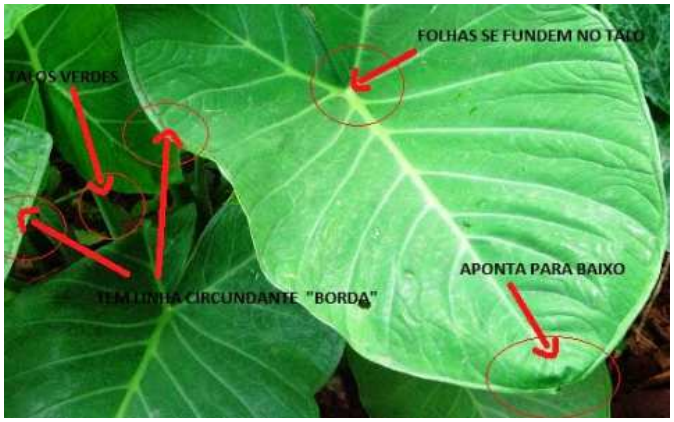

Figura 9 – *Xanthosoma taioba*

Fonte: Adaptado - Marchiori (1995)

Já as principais características da *Xanthosoma violaceum* é que as folhas se fundem antes do pecíolo e não há linha circundante, Figura 10.

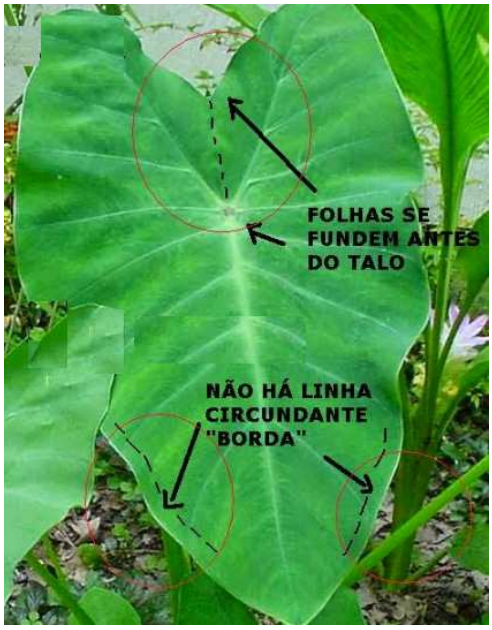

Figura 10 – ***Xanthosoma violaceum***
Fonte: Adaptado - Marchiori (1995)

As chaves dicotômicas são utilizadas por botânicos para classificar e identificar espécies, mas não é a única forma existente, com o advento de áreas como processamento de imagem, inteligência artificial e aprendizagem de máquina, problemas que só eram resolvidos por humanos passaram a ser solucionados por máquinas.

2.3 PROCESSAMENTO DE IMAGEM

O processamento digital de imagem (PDI) existe desde 1920, surgindo da necessidade de melhoramento de imagens utilizadas em jornais da época. Desde então o PDI está presente em diversas áreas como: Astronomia, Biologia, Matemática, Medicina, Física e etc.

Os interesses em técnicas de processamento digital de imagem podem ser divididos em duas atividades: melhora das informações visuais para interpretação humana e processamento de dados de imagens para armazenamento, transmissão e representação, considerando aprendizado de máquinas e reconhecimento de padrão (GONZALEZ; WOODS, 2008).

Segundo Gonzalez e Woods (2008) não é interessante limitar ambiente de atuação do processamento de imagem, sendo o campo de visão computacional um exemplo de abrangência.

O campo de visão computacional tem como meta utilizar computadores para emular a visão humana, incluindo o aprendizado e a capacidade de fazer inferências e agir com base em informações visuais (GARAGE, 2017).

Visando o uso de PDI na visão computacional são propostos três tipos de processos computacionais: processos de níveis baixos, médio e alto. O processo de nível baixo envolve operações primitivas, como pré-processamento de imagem para reduzir o ruído. Um processo de nível médio envolve atividades como segmentação (separação de uma imagem em regiões ou objetos), a descrição desses objetos para reduzi-los a uma forma adequada para o processamento computacional e a classificação de objetos individuais. No caso de nível alto envolve dar sentido a um conjunto de objetos reconhecidos, como na análise de imagem (GONZALEZ; WOODS, 2008).

Mas antes de se aprofundar em PDI é necessário estabelecer o que é uma imagem digital. Uma imagem pode ser definida como uma função bidimensional, *f(x,y)*, em que *x* e *y* são coordenadas espaciais (plano), e a amplitude de *f* em qualquer par de coordenadas *(x,y)* é chamada de intensidade ou nível de cinza da imagem nesse ponto. Quando *x*, *y* e os valores de intensidade *f* são quantidades finitas e discretas, chamamos de imagem digital (GONZALEZ; WOODS, 2008). As coordenadas fazem referência ao um elemento conhecido na literatura como pixel (PEDRINI; SCHWARTZ, 2008), serão tratados diferentes conceitos (Vizinho, Conectividade, Adjacencia e Caminho) que visam trata o relacionamento entre pixel.

O primeiro é a vizinhaça, um pixel *p* na coordenada *(x,y)* possui quatro vizinhos horizontais e verticais cujas as coordenadas são apresentadas por: *(x+1, y),(x-1,y),(x,y+1),(x,y-1)*. Esse conjunto é denominado vizinhança-4 de *p*, expresso por *N4(p)*. Cada pixel é uma unidade de distância de *(x,y)*, em que vizinhos de p ficarão para fora da imagem se *(x,y)* estiver na borda da imagem. Outro tipo de vizinhança são os diagonais de *p*, expresso em: *(x+1, y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1)* são expresas por *Nd(p)*. O *N4(p)* e o *Nd(p)* juntos formam a vinhaça-8 de *p* (GONZALEZ; WOODS, 2008).

Outro conceito é o da conectividade, sendo utilizado para fixar limites de objetos e componentes de regiões em uma imagem. Para verificar se dois elementos são conexos é preciso determinar se eles são vizinhos, o tipo de vizinhança utilizada e se os elementos satisfazem critérios de similaridade, como intensidade de cinza, cor ou textura (PEDRINI; SCHWARTZ, 2008).

A adjacência entre dois pixels ocorre quando em dois subconjuntos de pixels, *S1* e *S2*, são adjacentes se pelo menos um elemento em *S1* for adjacente a algum elemento de *S2* (PEDRINI; SCHWARTZ, 2008). Tem-se a adjacência mista que segundo Gonzalez e Woods (2008) visa remover ambiguidade da adjacência 8. Adjacência 8 é representada por $q \in$ conjunto

N8(p), em que *q* corresponde ao um pixel. A adjacência mista ocorre quando dois pixels *p* e *q* com valores pertencentes a *V* realize uma das regras:

1. *q* estiver em *N4(p)*, ou
2. *q* estiver em *Nd(p)* e o conjunto *N4(p)* intersecção *N4(q)* não contiver nenhum pixel cujos valores pertençam a *V*.

Um caminho na imagem do pixel *(X1,Y1)* a um pixel *(Xn,Yn)* é uma sequência de pixels distintos com coordenadas *(X1,Y1), (X2,Y2), ...,(Xn,Yn)* em que *n* é o comprimento do caminho, e *(Xi,Yi)* e *(Xi+1,Yi+1)* são adjacentes, tal que $i = 1,2,\ldots,n-1$ (PEDRINI; SCHWARTZ, 2008).

2.3.1 Distância

A literatura não propõe uma única forma para realizar o cálculo entre pixels, para Gonzalez e Woods (2008) inicialmente a função de distância *D* deve:

Para os pixels *p*, *q* e *z*, com coordenadas *(x,y), (s,f)* e *(v,w)*, respectivamente, *D* é uma função distancia se

1. $D(p,q) \geqslant 0$
2. $D(p,q) = D(q,p)$ e
3. $D(p,z) \leqslant D(p,q) + D(q,z)$.

entre as funções de distância tem-se a euclidiana, Equação 1:

$$D_E(f_1, f_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

Os pixels com uma distância menor ou igual a algum valor *d* formam um disco de raio *d* centrado em *f1*. A função requer mais esforço computacional e pode produzir valores fracionários (PEDRINI; SCHWARTZ, 2008).

A distância *D4* pode ser vista na Equação 2:

$$D_4(f_1, f_2) = |x_1 – x_2| + |y_1 - y_2| \tag{2}$$

Nessa os pixels com uma distância *D4* de *F1* menor ou igual a algum valor *d* formam um losango centrado em *f1*.

A distância *D8* entre *f1* e *f2*, também conhecida como *chessboard*, é definida na

Equação 3:

$$D_8(f_1, f_2) = max(|x_1 – x_2|, |y_1 - y_2|) \quad (3)$$

Os pixels com uma distância *D8* de *f1* menor ou igual a algum valor *d* formam um quadrado centrado em *f1* (GONZALEZ; WOODS, 2008).

2.4 SEGMENTAÇÃO

A segmentação é uma atividade que subdivide uma imagem em regiões ou objetos que a compõem, o nível de detalhe em que essa subdivisão ocorre depende do problema (GONZALEZ; WOODS, 2008). Após a segmentação, cada objeto é descrito por meio de suas propriedades geométricas e topológicas (PEDRINI; SCHWARTZ, 2008). Um exemplo são atributos como área, forma e textura que podem ser extraídos dos objetos e utilizados no processo de análise.

As abordagens utilizadas para segmentação de imagens são baseadas nas propriedades básicas dos níveis cinza da imagem, buscando detectar descontinuidade ou similaridade na imagem (GONZALEZ; WOODS, 2008). Na descontinuidade, os métodos visam particionar a imagem com base em mudanças abruptas nos níveis de cinzas, caracterizadas pela presença de pontos isolados, linhas ou bordas na imagem (GARAGE, 2017). Na similaridade os métodos procuram agrupar pontos da imagem que apresentam valores similares para um determinado conjunto de caracteristica (PEDRINI; SCHWARTZ, 2008). Nas próximas seções serão tratados com maior destaque a descontinuidade e a similaridades.

2.4.1 Detecção de Descontinuidades

Os tipos básicos de descontinuidades normalmente detectadas em imagens digitais são pontos, segmentos de retas, junções e bordas (PEDRINI; SCHWARTZ, 2008). Uma maneira de identificar descontinuidade é por meio da varredura da imagem por uma máscara (mapa), ou seja, uma convolução que nada mais é que uma operação com matrizes, Figura 11. A resposta da mascára é definida em relação à sua posição central (GONZALEZ; WOODS, 2008).

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 11 – Máscara 3 x 3
Fonte: Gonzalez e Woods (2008)

2.4.2 Detecção de pontos

Segundo Gonzalez e Woods (2008) é de conhecimento que a detecção de pontos deve se basear nas técnicas que utilizam as derivadas de segunda ordem, fazendo uso da combinação de expressões conhecidas como Laplaciano, Equação 4.

$$\bigtriangledown^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4(x,y) \tag{4}$$

A detecção de pontos em uma imagem pode ser realizada pela aplicação direta da máscara *h* definida como na Figura 12.

1	*1*	*1*
1	*-8*	*1*
1	*1*	*1*

Figura 12 – Máscara de detecção de ponto
Fonte: Gonzalez e Woods (2008)

Um ponto é detectado na posição central da máscara se $|R| > T$, em que T é um limiar não-negativo e R é dado pela Equação 5.

$$R = \sum_{i=1}^{\rho} w_i z_i \tag{5}$$

No caso essa formula envolve o cálculo da soma dos produtos dos coeficiente da máscara pelos níveis de cinza da região delimitada na máscara (PEDRINI; SCHWARTZ, 2008). O ponto será detectado se houver uma discrepância entre seu valor de nível de cinza e seus vizinhos (PEDRINI; SCHWARTZ, 2008).

2.4.3 Detecção de Linhas

É possível utiliza máscara laplaciana para detecção de linhas também, mas se deve levar em consideração o efeito de linha dupla gerado pela segunda derivada (GONZALEZ; WOODS, 2008). A Figura 13 (b) o resultado do filtro laplaciano sobre a Figura 13 (a), essa imagem possui valores negativos, o que segundo Gonzalez e Woods (2008) pede ajustes. A diferença pode ser vista na seção ampliada, os tons mais escuros de cinza representam valores negativos e os tons mais claros são positivos, o efeito de linha dupla. A Figura 13 (c) mostra uma abordagem incorreta, que é tomar como resposta o módulo dos valores calculados pelo filtro laplaciano, o que só dobra a espessura das linhas. A Figura 13 (d) mostra a abordagem correta que é a utilização de apenas os valores positivos do filtro laplaciano, o que acaba resultando em linhas mais finas (GONZALEZ; WOODS, 2008).

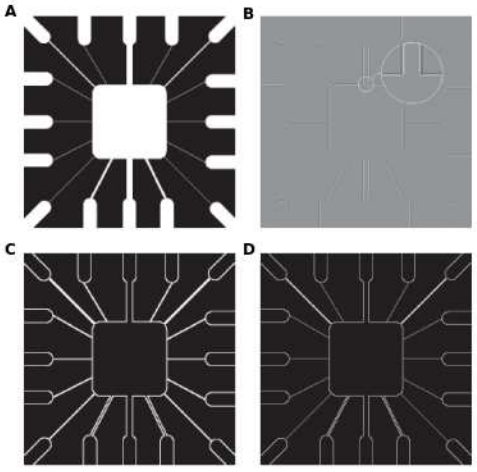

Figura 13 – Resultado do filtro laplaciano

Fonte: Gonzalez e Woods (2008)

A detecção de linhas (retas) também pode ser feita com uso de máscaras (PEDRINI; SCHWARTZ, 2008). Na Figura 14 são apresentados diferentes ângulos de retas que podem ser identificadas.

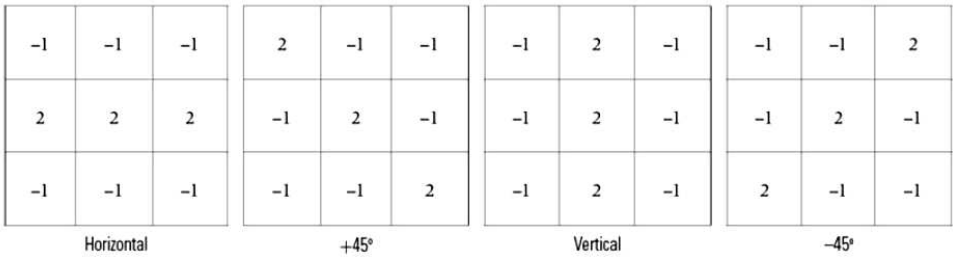

Figura 14 – Exemplo de filtros
Fonte: Gonzalez e Woods (2008)

2.4.4 Detecção de bordas

O método mais utilizado para segmentar as imagens com base nas variações abruptas de intensidade é a detecção de borda (GONZALEZ; WOODS, 2008). Uma borda é o limite ou fronteira entre duas regiões com propriedades relativamente distintas (PEDRINI; SCHWARTZ, 2008).

Os modelos de borda são classificados perante seus perfis de intensidade. A Figura 15 (a) apresenta uma borda vertical em degrau e o perfil de intensidade horizontal da borda, sendo um exemplo comum para modelagem de sólidos e animações. Na Figura 15 (b) o modelo da borda é denominado rampa, a inclinação rampa é inversamente proporcional ao grau de indefinição da borda, nesse modelo não tem a chamada borda fina. Um outro modelo de borda é denominado forma de telhado, Figura 15 (c), as bordas são modelos de linhas através de uma região com base de uma borda em forma de telhado, determinada pela espessura e a nitidez da linha. A borda telhado está presente na digitalização de desenho e imagens de satélite (GONZALEZ; WOODS, 2008).

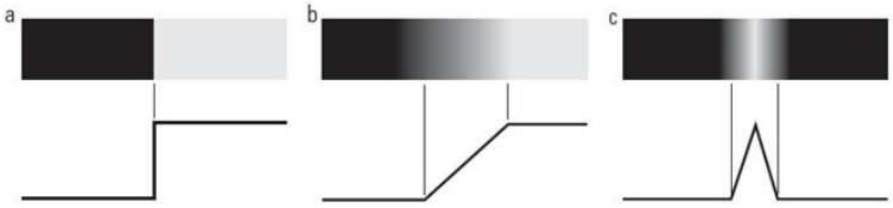

Figura 15 – Exemplo de borda
Fonte: Gonzalez e Woods (2008)

2.5 LIMIARIZAÇÃO

A limiarização é uma das técnicas mais simples de segmentação, e que consiste na classificação dos pixels de uma imagem de acordo com a especificação de um ou mais limiares (PEDRINI; SCHWARTZ, 2008). Segundo Gonzalez e Woods (2008) a limiarização tem uma posição de destaque na segmentação devido suas propriedades intuitivas, a simplicidade de implementação e a velocidade computacional. A limiarização possui variações, uma delas é a limiarização de intensidade. Suponha uma imagem *f(x,y)* composta por objetos claros sobre um fundo escuro de tal forma que os pixels do objeto e do fundo tenham valores de intensidade agrupados em dois grupos dominantes. Uma maneira óbvia de extrair os objetos do fundo é selecionar um limiar *T*, que separa estes modos (PEDRINI; SCHWARTZ, 2008). Portanto, qualquer ponto *(x,y)* na imagem em que $f(x,y) > T$ é chamado de ponto do objeto, caso contrário o ponto é denominado ponto do fundo.

2.5.1 Limiarização global e local

Quando as distribuições de intensidade dos pixels de fundo e dos objetos são suficientemente diferentes é possível um limiar global a imagem (PEDRINI; SCHWARTZ, 2008). Para isso é necessário um algoritmo capaz de calcular automaticamente o valor do

limiar para cada imagem (GONZALEZ; WOODS, 2008). O algoritmo iterativo tem as seguintes etapas:

1. Selecionar uma estimativa inicial para o limiar global T.
2. Segmentar a imagem usando T de limiarização de intensidade (na seção limiarização). O que dará origem a dois grupos de pixels: G_1, composto por todos os pixels com valores de intensidade $>T$, e G_2, composto de pixels com valores $\leqslant T$.
3. Nessa etapa é feito o cálculo dos valores de intensidade média m_1 e m_2 para pixels em G_1 e G_2 , respectivamente.
4. Calcular um novo valor de limiar, Equação 6:

$$T = \frac{1}{2}(m_1 + m_2) \tag{6}$$

5. Repita as etapas 2 a 4 até que a diferença entre valores de T em iterações sucessivas seja menor que os parâmetros predefinido ΔT.

Um exemplo do uso do algoritmo pode ser visto na Figura 16 (a) imagem original, e Figura 16 (b) é o histograma da imagem com o vale bem nítido. O algoritmo resultou no limiar T = 125,4 após três interações começando $T = m$ (média de intensidade da imagem) e usando ΔT = 0, a Figura 16 (c) mostra o resultado com T = 125 (GONZALEZ; WOODS, 2008).

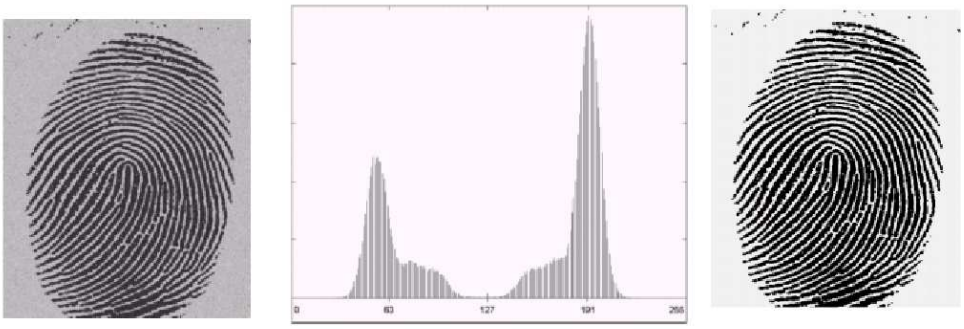

Figura 16 – Exemplo de algoritmo

Fonte: Gonzalez e Woods (2008)

Tem-se casos em que a utilização de um único limiar para segmentar toda a imagem não produza bons resultados, sendo a limiarização local uma alternativa (PEDRINI; SCHWARTZ, 2008). A forma comum de realizar a limiarização local é analisar as intensidades dos pixels em uma região da imagem para determinar limiares locais (PEDRINI; SCHWARTZ, 2008).

Para calcular um limiar local é utilizada a média dos valores de intensidade em uma vizinhança local da imagem. As duas outras equações utilizadas são: a mediana (Equação 7) e

a média dos valores mínimo e maximo (Equação 8) em que v é uma vizinhança local ao ponto p na imagem (PEDRINI; SCHWARTZ, 2008).

$$T = medianav(p) \tag{7}$$

$$T = \frac{minv(p) + maxv(p)}{2} \tag{8}$$

2.6 SEGMENTAÇÃO DE REGIÃO

Os métodos de segmentação dessa seção visam detectar regiões diretamente nas imagens, ao invés de encontrar as bordas que delimitam as regiões (PEDRINI; SCHWARTZ, 2008). Essa seção será dividida entre crescimento, divisão e união de regiões.

2.6.1 Crescimento região

Segundo Gonzalez e Woods (2008) o crescimento de região é um procedimento que agrupa os pixels ou as sub-regiões maiores com base em critérios predefinidos para o crescimento. Uma abordagem simples é iniciar com um conjuntos de pixels denominado sementes, sendo escolhidos aleatoriamente, a partir deles crescer as regiões anexando a cada ponto semente outros pixels que possuem características semelhantes (PEDRINI; SCHWARTZ, 2008). Na Figura 17 (a) é mostrado a imagem original, cada valor na imagem diz respeito ao nível de cinza de cada pixel, as sementes escolhidas foram (1,1) e (5,1). Na Figura 17 (b) é feita a segmentação utilizando uma diferença absoluta entre os níveis de cinza de cada pixel em menor que 4, separando a imagem em R_1 e R_2. Para outro efeito de exemplo tem a Figura 17 (c) tem o exemplo de valor absoluto menor que 8 (PEDRINI; SCHWARTZ, 2008).

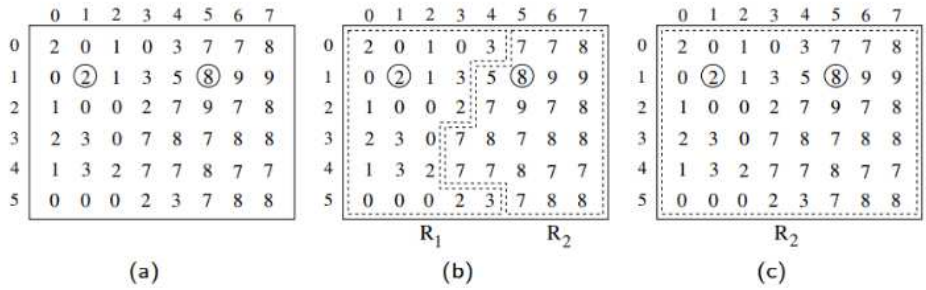

Figura 17 – Exemplo do uso da sementes

Fonte: Pedrini e Schwartz (2008)

2.6.2 Divisão de região

Um outro processo é subdividir uma imagem inicialmente em um conjunto de regiões distintas, visando satisfazer as condições de segmentação, a divisão de região (GONZALEZ; WOODS, 2008). A divisão de região inicia-se com regiões formadas por pixels da imagem e, recursivamente, subdivide as regiões não-homogêneas em áreas menores. Uma técnica comum de subdivisão utiliza a representação *quadtree*, que é uma estrutura hierárquica baseada na decomposição recursiva e regular da imagem em quadrantes, de maneira que, para qualquer região R_i, $P(R_i)$ = verdadeiro. Se $P(R)$ = falso então a imagem deve ser dividida em quadrantes (PEDRINI; SCHWARTZ, 2008). A Figura 18, mostra o *quadtree* da imagem correspondente.

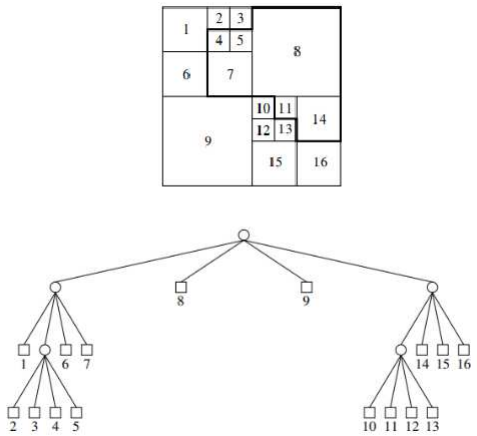

Figura 18 – Exemplo do *quadtree*

Fonte: Pedrini e Schwartz (2008)

2.6.3 União de regiões

Apenas a utilização da divisão de regiões, a partição final conterá regiões adjacentes com propriedades idênticas, o que pode ser solucionada com a união junto da divisão. A união só de regiões é permitida a regiões adjacentes cujos pixels combinados cumprem a propriedade de P. Duas regiões adjacentes R_i e R_j são unidas se $P(R_iUR_j)$ = verdadeiro (PEDRINI; SCHWARTZ, 2008). O algoritmo para divisão e união segue os seguintes passos:

1. Dividir em quatro quadrantes separados qualquer região R_i para a qual $Q(R_i)$ = falso.
2. Quando não for possível continuar dividindo, unir as regiões adjacentes R_i e R_j para as quais $Q(R_jUR_k)$ = verdade.
3. Parar quando a união não for mais possível (GONZALEZ; WOODS, 2008).

Depois de apresentado um breve resumo de técnicas de processamento de imagem, será introduzido os próximos tópicos importantes para conclusão do trabalho, Inteligência artificial e aprendizado de máquina.

2.7 INTELIGÊNCIA ARTIFICIAL E APRENDIZADO DE MÁQUINA

A Inteligência Artificial (IA) pode ser definida como um estudo de como os computadores podem fazer tarefas que hoje são melhores desempenhadas pelas pessoas (RICH et al., 1983). Essa é uma das inúmeras definições que esse campo possui. Nela se busca construir mecanismos ou dispositivos que simulem a capacidade do ser humano de realizar atividades simples como reconhecer um rosto em uma foto. Essa área de estudo é bem recente no campo da ciência e engenharia, tendo trabalhos verificados a partir da Segunda Guerra Mundial (NORVIG; RUSSELL, 2004). Durante a história da IA, existiram estudiosos que ajudaram ela a se desenvolver, entre eles o cientista Warren McCulloch e Walter Pitts (1943) que propuseram um modelo de neurônios artificiais, sendo considerado o primeiro trabalho reconhecido na área de IA. Eles mostraram que qualquer função computável podia ser calculada por uma rede de neurônios conectados. Outros importantes cientistas foram Allen Newell e Herbert Simon que no verão de 1956 criaram o Logic Theorist um programa de computador, criado para imitar as habilidades de resolver problemas. Alan Turing foi outro cientista que ajudou a desenvolver a inteligência artificial, em 1947 ele já divulgava palestra

sobre o tema na Sociedade Matemática de Londres e articulou um programa de trabalhos em 1950, “Computing Machinery and Intelligence”. Nesse artigo, ele apresentou o teste de Turing, basicamente um computador passava por um interrogatório aplicado por um humano, o teste teria sucesso caso o interrogador não conseguisse descobrir se a resposta gerada pela a máquina vinha de uma pessoa ou de um computador. Para o computador passar no teste, deveria ter as seguintes disciplinas:

- Processamento de linguagem natural;
- Representação de conhecimento para armazenar o que sabe ou ouve;
- Raciocínio automatizado para usar as informações armazenadas com a finalidade de responder a perguntas e tirar novas conclusões;
- Aprendizado de máquina para se adaptar a novas circunstâncias e para detectar e extrapolar padrões.

No teste de Turing evitou-se a interação física direta entre o interrogador e o computador deixando de fora as disciplinas:

- Visão computacional para perceber objetos;
- Robótica para manipular objetos e movimentar-se.

Juntas essas seis disciplinas compõem a maior parte da área IA (NORVIG; RUSSELL, 2004). Na construção de sistemas inteligentes, a disciplina aprendizagem de máquina, ganha um destaque, já que a capacidade de aprendizagem é essencial para o comportamento inteligente. Atividades como memorizar, observar e explorar situações melhorar habilidades cognitivas. Definir o que é aprendizagem de máquina passa pelo mesmo problema que é definir inteligência artificial, devido às várias definições que cada uma tem. Uma delas, diz que, aprendizagem de máquina é a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência (FACELI et al., 2011). Essa é uma boa definição, por colocar ênfase na experiência passada as máquinas. Um quesito muito importante para que esse processo ocorra é empregado o princípio da indução, na qual se obtêm conclusões genéricas a partir de um conjunto de exemplos. Basicamente eles aprendem a induzir uma função ou hipótese capaz de resolver um problema, a partir de um conjunto de dados. Por exemplo, tem-se um banco de dados de espécies florestais, nele uma tabela contendo atributos como: cor, tamanho da planta, tamanho da folha entre outros, para cada espécie. Pode ser feito o uso de um algoritmo de aprendizagem de máquina para aprender uma hipótese capaz de identificar espécies, através dos valores de seus atributos de entrada. Busca-se induzir uma hipótese capaz de reconhecer novas

espécies de plantas diferentes daquelas que foram utilizadas para aprender a regra de decisão. Assim, uma vez induzida uma hipótese, é desejável que ela também seja válida para outros objetos do mesmo domínio ou problema que não fazem parte do conjunto de treinamento, uma nova espécie de planta que não estava no conjunto de dados de treinamento, mas que a regra gerada tem que valer mesmo assim para essa nova espécie. A essa propriedade de uma hipótese continuar a ser válida para novos objetos dá-se o nome capacidade de generalização da hipótese. Quanto maior a capacidade de generalização uma hipótese apresenta, melhor serão os resultados ao aplicar em dados novos (FACELI et al., 2011). No processo de aprendizagem a partir de um conjunto de dados de treinamento, o algoritmo busca uma hipótese. Cada algoritmo utiliza uma forma para representar a hipótese induzida. Tem-se formas em redes neurais artificiais que são apresentados por um conjunto de valores reais, associados aos pesos de conexões da rede. Já em árvores de decisão utilizam uma estrutura de árvore em que cada nó interno é representado por uma pergunta referente ao valor de um atributo e cada nó externo está associado a uma classe e por fim tem-se as regras que tentam representar uma hipótese como visto na Figura 19 (FACELI et al., 2011). Dentre os algoritmos apresentados para representar hipóteses induzidas, serão objeto de estudo as redes neurais.

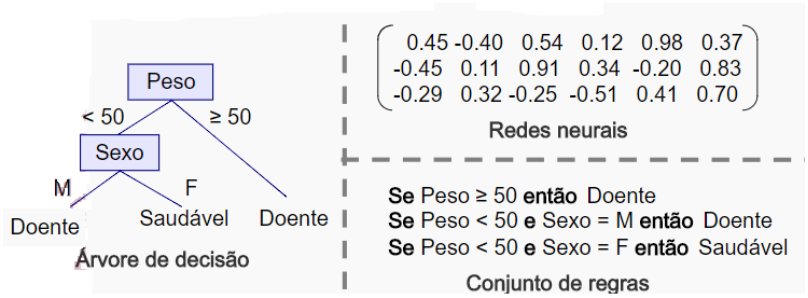

Figura 19 – Representações de hipóteses

Fonte: Faceli et al. (2011)

2.8 REDES NEURAIS

Dentro da disciplina aprendizado de máquina existem as redes neurais, que na sua forma mais geral é uma máquina projetada para modelar a maneira como o cérebro realiza uma tarefa em particular. A rede é normalmente implementada utilizando componentes eletrônicos ou é simulada por programação em um computador digital (HAYKIN, 2001). O conceito de

redes neurais tem como sua principal origem a disciplina de biologia, com foco no estudo do cérebro humano, já que o próprio cérebro é uma grande máquina de aprendizagem, capaz de processar as informações recebidas, analisá-las e gerar um resultado com base nas experiências adquiridas pelo indivíduo. O cérebro humano é constituído por grande conjunto de neurônios, que são também denominadas de células nervosas que têm a propriedade de receber e transmitir estímulos, permitindo ao organismo responder a alterações do meio (LOPES, 1995).

O neurônio biológico pode ser observado na Figura 20, é composto por um corpo celular denominado pericário, onde partem os prolongamentos e que acomoda o núcleo do neurônio, dendritos que são prolongamentos responsáveis por receber os estímulos do ambiente e o axônio um prolongamento único condutor dos impulsos nervosos à outras células. O impulso nervoso que se propaga pelo neurônio é de origem elétrica (LOPES, 1995) resultando alteração de cargas interna e externa da membrana celular.

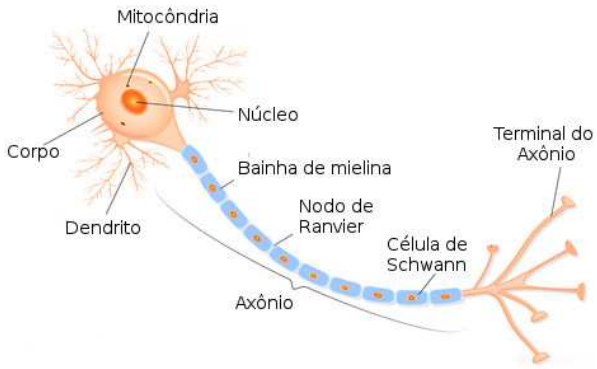

Figura 20 – Representação do Neurônio Biológico

Fonte: Lopes (1995)

Quando a membrana do neurônio está em repouso apresenta cargas positivas no lado externo e carga negativo no lado interno, quando o neurônio está nesse estado se diz que ele está polarizado. Essa diferença de cargas é mantida pela bomba de sódio e potássio, que só pode ser alterada com entrada de um estímulo químico, mecânico ou elétrico.

É importante destacar que o estímulo nervoso deve ser forte o suficiente para haver a mudança do estado polarizado, o que determina isso é o valor crítico, que pode variar em diferentes tipos de neurônio, abaixo desse valor o estimulo só provoca alterações locais na membrana não desencadeando o impulso nervoso.

Conhecendo o papel do impulso nervoso no neurônio, pode-se entrar em outro ponto importante que é sinapse, que é processo quando tem-se a transmissão de um impulso nervoso de um neurônio a outra célula de órgãos efetores (LOPES, 1995). Essa comunicação ocorre na região denominada fenda sináptica, que fica entre o terminal do axônio e os dendritos que irão receber os impulsos. A transmissão ocorre na mudança do estado despolarizado para polarizado

da membrana.

Essa mudança vai ocorrer na entrada de cálcio no axônio e com isso vai estimular a geração de neurotransmissores liberados na fenda sináptica, atingindo o dentrito do próximo neurônio o ativando, como mostrado na Figura 21. No final a membrana fica polarizada novamente. Isso é só o funcionamento de um neurônio dos inúmeros existente no cérebro, então basicamente quando uma pessoa ver um objeto como uma folha, ela recebe um estimulo visual que será convertido pelo olho em estimulo nervoso que irá passar para o cérebro e com isso gerar sinapse com neurônios famílarizado com o objeto observado (LENT, 2010).

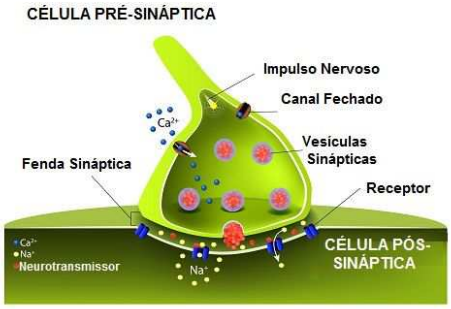

Figura 21 – Representação do Impulso Elétrico

Fonte: Lopes (1995)

Essa breve explicação sobre o funcionamento do neurônio biológico é importante para apresentar o neurônio artificial, já que ele se espelhou no neurônio biológico para ser criado, lembrando que o nível de estrutura de organização presente no cérebro humano é único (HAYKIN, 2001), não sendo encontrado em nenhum computador, sendo apenas uma representação bem primitiva de um neurônio. O modelo utilizado para representá-lo é visto na Figura 22.

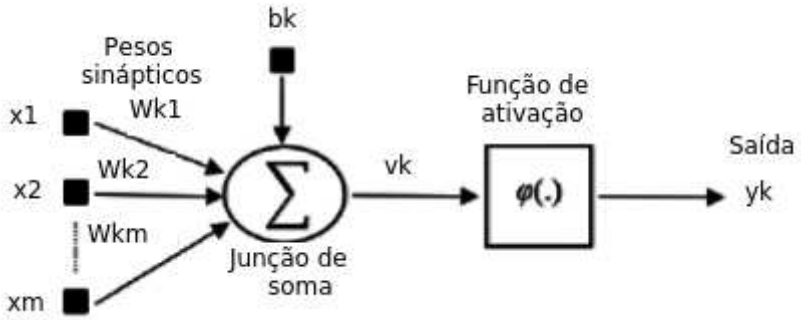

Figura 22 – Representação do Neurônio Artificial

Fonte: Haykin (2001)

A entrada do neurônio artificial é o conjunto de sinapse ou elos de conexões, cada uma caracterizada por um peso próprio. Especificamente, um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicada por peso sináptico $w_k j$. O índice desse peso é descrito da seguinte forma: o primeiro índice se refere ao neurônio em questão e o segundo se refere ao terminal de entrada da sinapse à qual o peso se refere. Lembrando que o peso sináptico de um neurônio pode ser tanto positivo quanto negativo. O segundo elemento é um somador para somar os sinais de entrada, ponderados pelas respectivas sinapses. E por último tem-se a função de ativação para restringir a amplitude da saída de um neurônio (HAYKIN, 2001), semelhante ao que acontece com valor critico dos neurônios biológico que determinam se o neurônio será ativado, ainda tem-se a *bias* representadas por b_k que tem a função de aumentar ou diminuir o que passa na função de ativação. Basicamente ao neurônio artificial é passado um estimulo, que são x_m, deve-se então utilizar cada um desses estímulos e multiplicar por um número (peso), desse produto é feito um somatório que será somado ao final com b_k (*bias*), o resultado é passado por uma função de ativação, que vai dizer se o neurônio será ativado ou não (SILVA et al., 2015). Matematicamente o neurônio é representado pela seguintes equações 10, 11 e 12. Na qual a equação 10 representa a saída do combinador linear dos sinais de entrada, X_j são os sinais de entrada e W_j os pesos sinápticos. A equação 11 representa o sinal de saída do neurônio, em que b_k são as *bias* e φ a função de ativação. Por último a equação 12 representa o campo local induzido (HAYKIN, 2001):

$$U_k = \sum_{j=1}^{m} w_{kj} * x_j \tag{9}$$

$$y_k = \varphi(U_k + b_k) \tag{10}$$

$$V_k = U_k + b_k \tag{11}$$

A escolha das funções de ativação de uma rede neural é uma consideração importante uma vez que define como devem ser a entrada dos dados na rede, na Figura 23 são apresentado alguns exemplos. Entre as funções tem-se a função linear mostrada na Figura 23 (a) geralmente utilizada nas camadas de saída em redes neurais de regressão. Já a função Sigmond, Figura 23 (b), utilizada em redes neurais com propagação positiva, onde se necessita apenas saídas de números positivos outra função muito comum é a tangente hiperbólica, Figura 23 (c), ela gera saídas no intervalo de -1 a 1, essas são só algumas das funções de ativação existente (DACOM, 2016).

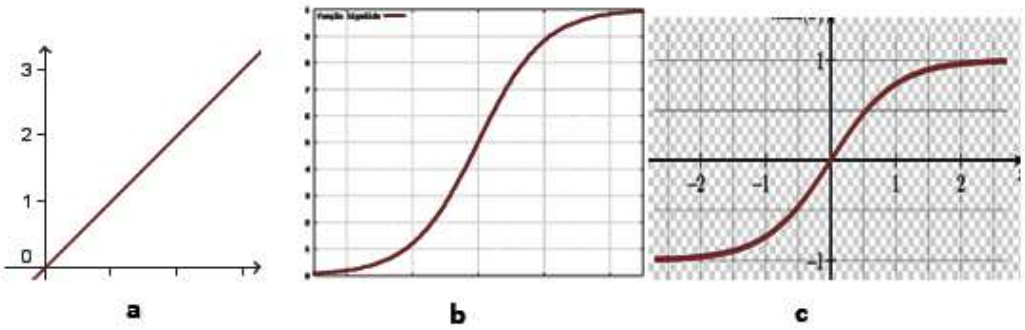

Figura 23 – Representação por gráfico das curvas de ativação

Fonte: Dacom (2016)

Com o modelo de neurônio artificial estabelecido, é importante explorar as redes em si, já que com apenas um neurônio não é possível solucionar todos os problemas. Para construir uma rede neural se faz uso de um conjunto de neurônios, sendo essa rede estruturada baseada no problema que se quer resolver. Em geral é possível identificar três classes de arquiteturas de rede fundamentalmente diferentes. As primeiras são as redes alimentadas adiante com uma camada única, Figura 24. Nela os neurônios estão organizados na forma de camada, sendo a forma mais simples tendo uma camada de entrada de nós de fonte que se projeta sobre uma camada de saída de neurônio, sendo uma rede acíclica (HAYKIN, 2001).

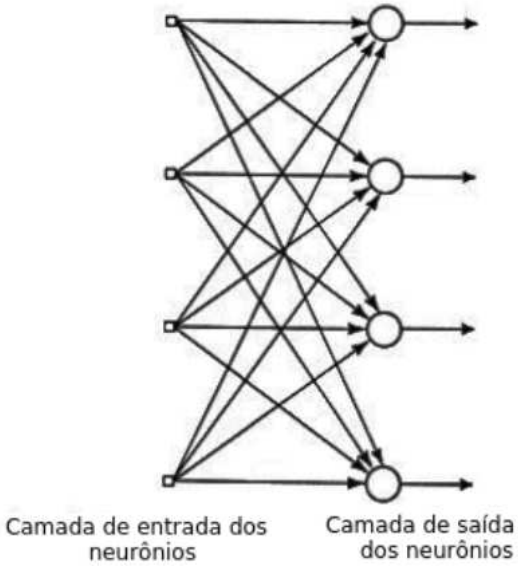

Figura 24 – Rede alimentada adiante ou acíclica com uma única camada de neurônios

Fonte: Haykin (2001)

A segunda classe são as redes alimentadas diretamente com múltiplas camadas que se distinguem pela presença de uma ou mais camadas ocultas, tendo neurônios denominados de ocultos, que têm a função de intervir entre a entrada externa e a saída da rede de uma maneira útil (SILVA et al., 2015). Podendo ter mais de uma camada oculta, o que dá mais poder a rede, já que cada neurônio fica responsável por uma tarefa.

A transmissão entre os neurônios se dá primeiramente no sinal aplicado aos nós de fonte da camada de entrada, o sinal de saída dessa camada será aplicado na entrada da próxima camada e assim sucessivamente, até chegar na última camada que terá a saída global para o impulso aplicado na entrada.

A Figura 25 apresenta uma rede neural de múltiplas camadas com uma única camada oculta, denominada uma rede 10-4-2 porque tem 10 neurônios de fontes, 4 neurônios ocultos e 2 neurônios de saída (KOVACS, 2002). Tem-se também as redes neurais recorrentes, elas apresentam pelo menos um laço de realimentação, formando ciclos (SILVA et al., 2015), como visto na Figura 26 (HAYKIN, 2001).

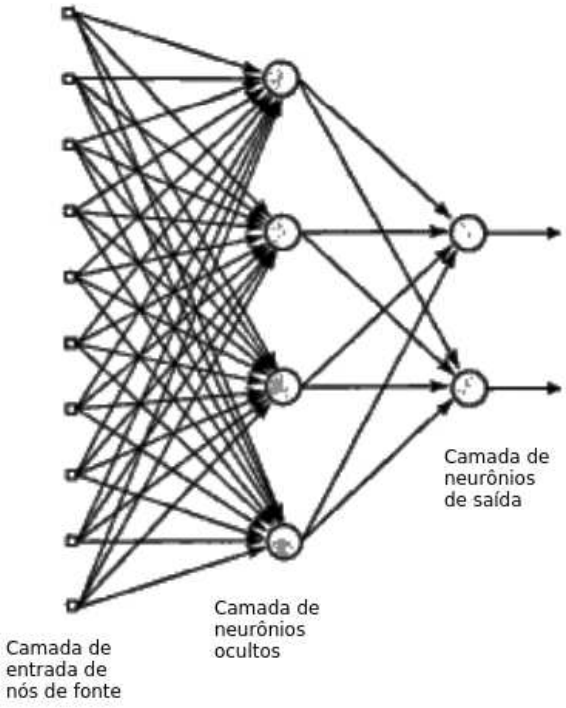

Figura 25 – Rede alimentada adiante ou acíclica totalmente conectada com uma camada oculta e uma camada de saída

Fonte: Haykin (2001)

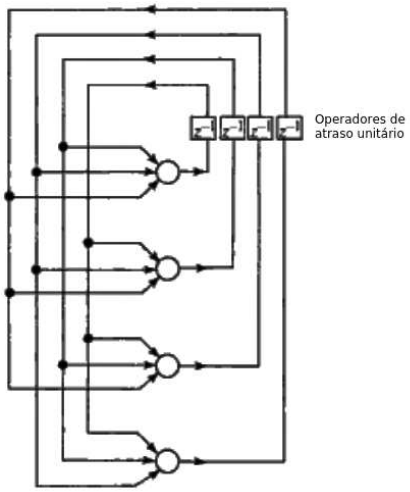

Figura 26 – Rede recorrente sem laços de auto-realimentação e sem neurônios ocultos

Fonte: Haykin (2001)

Outro modelo de arquitetura de redes, são as redes neurais convolucionais CNNs, composta por: neurônios de entrada, camadas escondidas e a camada de saída totalmente conectada Figura 27.

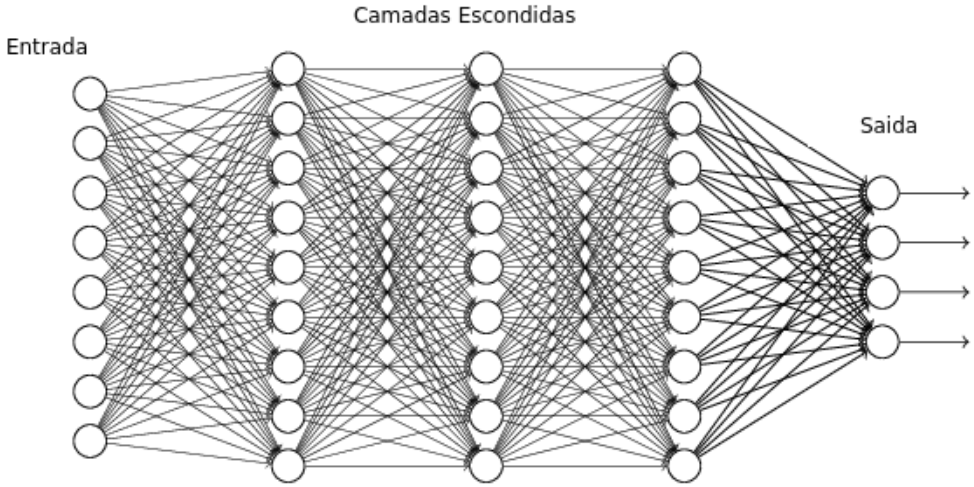

Figura 27 – Exemplo de rede neural convolucional

Fonte: Nielsen (2015)

2.8.1 CNNs - *Convolutional Neural Networks*

No contexto de inteligência artificial e aprendizado de máquina existem às CNNs. Esse tipo de rede se inspirou no funcionamento do córtex visual. O córtex possui pequenas regiões de células que são sensíveis a regiões especificas do campo visual (HUBEL; WIESEL, 1962). Hubel e Wiesel mostraram que alguns neurônios do cérebro responderam apenas na presença de bordas de determinada orientação (WURTZ, 2009). Por exemplo alguns neurônios disparados quando expostos a bordas verticais e outros com bordas horizontais ou na diagonal. Em seus estudos eles descobriram que esses neurônios estavam organizados em uma arquitetura em colunas e que juntos produziam a percepção visual. Essa ideia de componentes especializados dentro de um sistema com tarefas especificas é a base das CNNs, um modelo de rede muito utilizada em reconhecimento de padrões em imagens (VARGAS et al., 2015). Uma CNN é composta por uma sequência de camadas. Além da camada de entrada, que normalmente é composta por uma imagem com largura, altura e profundidade (RGB), existem três camadas principais: camada convolucional, camada de *pooling* e camada totalmente conectada. Além disso, após uma camada de convolução é comum uma camada de ativação (VARGAS et al., 2015).

2.8.2 Camada Convolucional

A primeira camada de uma CNN é a camada convolutiva, composta por um mapa de filtros capazes de aprender de acordo com o treinamento (NUMPYDL, 2015). Para efeito de explicação da convolução, será dado um exemplo de uma imagem 32x32 e profundidade 3 (RGB). Imagine a luz de uma lanterna apontada no canto superior esquerdo da imagem, essa luz abrange o espaço de uma área de 5x5 deslizando por todas as partes da imagem. Essa lanterna em termos de aprendizagem de máquina é conhecida como filtro, e a região que a luz atinge é o campo receptivo. A medida que o filtro se deslocar em torno da imagem de entrada, ocorre a multiplicação de matrizes envolvendo os valores de pixel originais da imagem, já que o filtro é uma matriz 5x5x3. Ao todo são feitas 75 multiplicações que ao final geram um único número,

representado quando o filtro está no lado esquerdo da imagem, esse processo é repetido em toda imagem. Ao final é gerado uma matriz de 28 x 28 x 1 que é o mapa de ativação, isso quer dizer que existem 784 locais diferentes que um filtro 5 x 5 pode esta contido em uma imagem 32 x 32. Esses 784 números são mapeados para uma matriz de 28 x 28, exemplo na Figura 28.

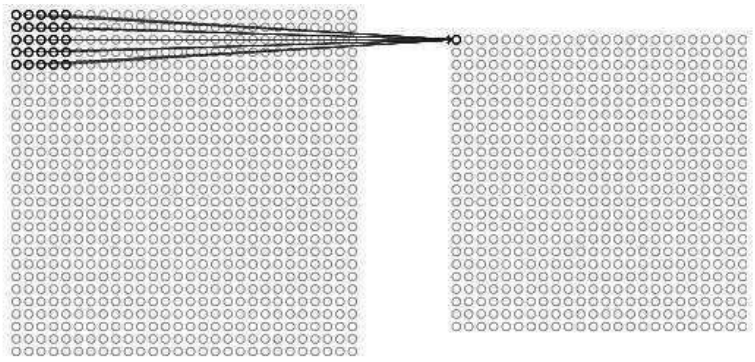

Figura 28 – Exemplo de um mapa de filtro 5x5
Fonte: Vargas et al. (2015)

Esses mapas de filtros buscam a detecção de características da imagem como: bordas, retas, cores simples e curvas (DESHPANDE, 2015). Um exemplo pode ser visto na Figura 29 em que se tem um mapa de filtro de curva com as dimensões 7 x 7 x 3.

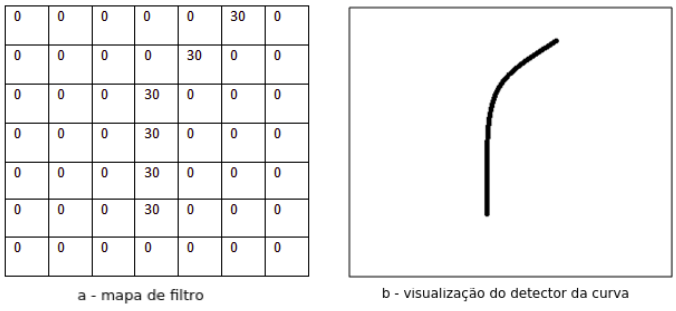

Figura 29 – Exemplo mapa de filtro de curva
Fonte: Deshpande (2015)

Esse filtro é passado em uma imagem em busca da detecção de uma curva compatível, como é observado no exemplo a seguir, Figura 30. A matriz de pixels da imagem original será multiplicada pela matriz de filtro: (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600. No caso em que o valor resultante da multiplicação de matrizes seja grande, significa que há a presença de uma curva compatível com o filtro. Caso o filtro passe em uma área da imagem sem a curva compatível, o valor resultante da multiplicação das matrizes será próximo a zero.

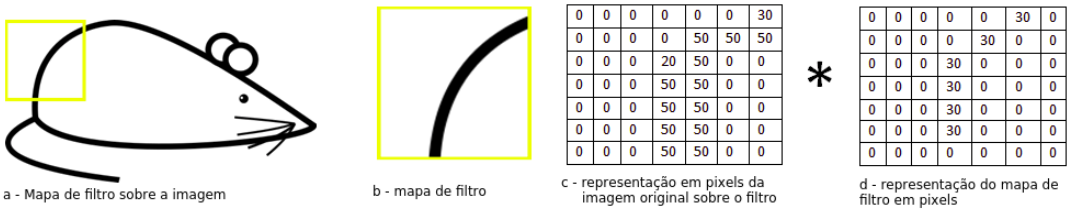

Figura 30 – Exemplo de detetcção de curva

Fonte: Deshpande (2015)

Ao final dessa camada de convolução é gerado o mapa de ativação, esse mapa mostrará a chance de haver uma curva na imagem. Seguindo o exemplo apresentado, o mapa de ativação 26 x 26 x 1 terá o valor de 6600, o que mostra a presença da curva ativando o filtro. Sendo apenas um filtro de linhas retas que se curvam para a direita, é possível ter outros filtros, sendo que quanto mais filtros, maior a profundidade do mapa de ativação e mais informação sobre a entrada (NUMPYDL, 2015).

2.8.3 Modelo Clássico

Uma rede neural convolucional não é só composta da camada convolutiva, existem camadas intercaladas que visam fornecer a não-linearidade e preservar as dimensões da imagem. O modelo clássico é dividido entre: entrada, convolução, *Pc* e *Pooling* (NUMPYDL, 2015).

2.8.4 ReLU (*Rectified Linear Unit*)

Para trabalhar com imagens além das camadas de convoluções, tem-se a função de ativação ReLU, essa função introduz a não linearidade, tornando o aprendizado mais rápido.

Ela faz isso trocando valores de pixels negativos da imagem por zero (VARGAS et al., 2015). Como visto na Figura 31.

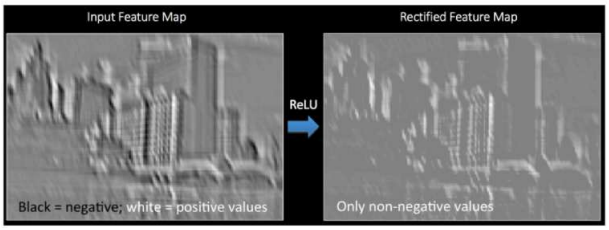

Figura 31 – Exemplo da Função ReLU
Fonte: Vargas et al. (2015)

2.8.5 Camada *Pooling*

Em redes convolucionais se faz uso de camada de *pooling*, logo após uma cada de convolução. Essa técnica visa reduzir o tamanho espacial das matrizes resultantes da convolução. O que reduz a quantidade de parâmetros a serem aprendidos na rede, contribuindo para evitar a memorização (VARGAS et al., 2015). Essa camada opera independente em cada um dos canais do resultado da convolução. É necessário estabelecer o tamanho do filtro da camada de *pooling*, na Figura 32 é utilizado um filtro 2x2 para criar uma camada de *maxpooling*, basicamente é uma operação com matrizes que extrair o maior elemento do quadrante que é aplicado o filtro.

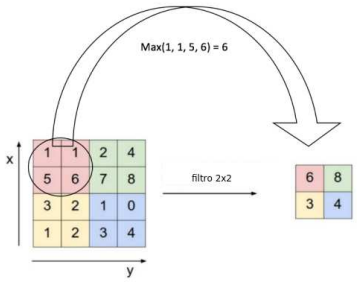

Figura 32 – Camada de Pooling
Fonte: Vargas et al. (2015)

Como mostrado, a primeira camada convolucional detecta características de baixo nível, como bordas e curvas. Essas características não são o suficiente para detectar objetos complexos como mãos e olhos (DESHPANDE, 2015), então são utilizadas outras camadas

convolucionais ligadas em sequência. Basicamente a saída da primeira camada convolucional será a entrada da segunda camada convolucional, nessa nova camada a entrada é o mapa de ativação que resulta da primeira camada. Então cada camada da entrada está basicamente descrevendo os locais na imagem original para onde determinados recursos de baixo nível aparecem. Aplicando um conjunto de filtros na segunda camada a saída será ativada com características de nível superior. Os tipos dessas características podem ser semicírculos, quadrados ou figuras complexas.

2.8.6 Camada Totalmente Conectada

Com a detecção de objetos complexos, a última camada é totalmente conectada, seja qual for sua entrada: ReLU, *Pooling* ou Convolucional. Essa última camada recebe um volume de entrada e produz um vetor N dimensional onde N é o número de classes que o programa deve escolher (NUMPYDL, 2015). Por exemplo, tentar classificar dígitos, o N seria 10 já que há 10 dígitos. Cada número desse vetor dimensional N representa a probabilidade de uma determinada classe. Se o vetor resultante é [0 .1 0 0 0 0 0 0 0 .75] o que significa que há 10% de chance de ser o número 1 e que há 75% de chance de ser 9, essa é uma das maneiras de apresentar a saída da rede neural convolucional. Essa camada totalmente conectada examina quais características de alto nível se correlacionam mais fortemente a uma determinada classe e tem pesos específicos que em conjunto das camadas anterior, obtém as probabilidades corretas para diferentes classes (DESHPANDE, 2015).

2.8.7 Aprendizado

Com o modelo de rede neural estabelecido é possível partir para o aprendizado, dentro das redes neurais, tem-se três tipos de aprendizado (FACELI et al., 2011). O primeiro é o supervisionado, que basicamente existe a presença de um superior que conhece a saída desejada para cada exemplo. Um bom exemplo é um adulto (supervisor) tentando ensinar o que é uma bola para uma criança, então ele mostra o objeto para a criança e junto passa a informação que aquilo que ela está vendo é uma bola, no final o cérebro da criança cria sinapses associadas a

esse objeto. Outro treinamento é o não supervisionado, não existe um supervisor, é aquele que para fazer modificações nos valores das conexões sinápticas não se usam as informações sobre a resposta da rede, isto é se a resposta está correta ou não. Outra forma de aprendizado é o por reforço nesse método o objetivo é reforçar ou recompensar uma ação considerada positiva e punir uma considerada negativa.

2.8.8 Algoritmo *Backpropagation*

Dentro do aprendizado supervisionado tem-se técnicas como *Backpropagation*, que é um algoritmo de treinamento utilizado em redes multicamadas. Foi criado em 1986 por Rumelhart, Hilton e Williams, esse algoritmo permitia ajustar os pesos em uma rede com mais de uma camada, resolvendo o problema de aprendizado do reconhecimento de padrões (SILVA et al., 2015). Nesse tipo de algoritmo a função de ativação precisa ser contínua, diferenciável e de preferência, não decrescente (FACELI et al., 2011).

O *Backpropagation* possui duas fases, a fase para frente (*forward*) e para trás (*backward*). Na fase *forward*, cada objeto de entrada é apresentado à rede. O objeto é primeiramente recebido por cada um dos neurônios da primeira camada intermediária da rede, quando é ponderado pelo peso associado a suas conexões de entrada correspondentes. Cada neurônio nessa camada aplica a função de ativação a sua entrada total e produz um valor de saída, que é utilizado como valor de entrada pelos neurônios da camada seguinte. Esse processo continua até que os neurônios da camada de saída produzam cada um seu valor de saída, que é então comparado ao valor desejado para a saída desse neurônio (FACELI et al., 2011).

A diferença entre esses valores indica o erro cometido pela rede para o objeto apresentado, esse erro é utilizado na fase *backward* para ajustar seus pesos de entrada, ocorrendo da camada de saída até a primeira camada intermediária (MITCHELL, 1997). No algoritmo diferentes critérios de parada podem ser utilizados, como, o número máximo de ciclos ou uma taxa máxima de erro (FACELI et al., 2011).

2.8.9 Algoritmo *Dropout*

O *Dropout* é uma técnica de redes neurais, que faz uso da eliminação aleatória de neurônios durante o processo de aprendizagem (BALDI; SADOWSKI, 2013). O algoritmo faz

alterações na rede buscando ter maior desempenho em previsões fora da amostra. Imagine o treinamento de uma rede neural, com entrada x e saida esperada y Figura 33.

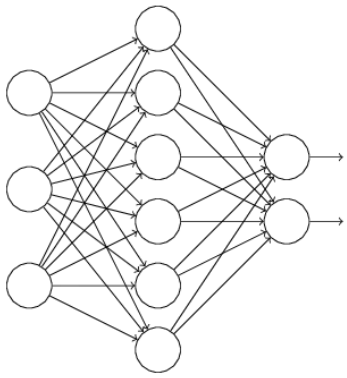

Figura 33 – Exemplo de rede neural
Fonte: Book (2018)

O normal é utilizar o algoritmo de *Backpropagation*, mas com o *Dropout* esse processo é modificado. No *Dropout* alguns neurônios ocultos na rede são eliminados aleatoriamente, deixando os neurônios de entrada e saída fixos Figura 34.

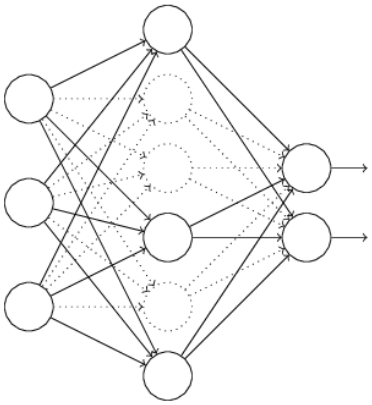

Figura 34 – Exemplo de rede neural após o *Dropout*
Fonte: Book (2018)

Após isso é aplicado o *Backpropagation* normalmente (BOOK, 2018). Finalizado o algoritmo são restaurados os neurônios removidos, sendo aplicado novamente o algoritmo de *Dropout* e depois o *Backpropagation*. Ao repetir esse processo a rede aprenderá pesos sob condições em que parte dos neurônios ocultos foram excluídos. No momento que se executar a rede completa, mais neurônios ocultos estarão ativos, todo esse processo reduz o *Overfitting* da rede

(BOOK, 2018). O *Overfitting* é um termo da estatística utilizado para descrever um modelo que se ajusta aos seus dados, mas é ineficaz para prever novos resultados em dados desconhecidos (MONTES, 2014).

2.8.10 YOLO e Googlenet

Com os elementos de uma rede neural convolucional determinados, foi feito a escolha de dois modelos de rede para realizar o estudo, no caso Googlenet e o YOLO. A escolha foi feita principalmente pela forma diferente de manipulação das imagens entre cada rede. As próximas seções serão para detalhar o funcionamento de cada uma.

2.8.11 YOLO - *You Only Look Once*

O YOLO é um sistema de detecção de objetos em tempo real, quer dizer que é capaz de identificar objetos em vídeos não apenas em imagem. Para realizar essa tarefa ele utiliza o conceito de "dividir para conquistar", ele faz uso de partes da imagem para identificar o objeto e sua posição (FARHADI et al., 2015). Na Figura 35 (a) é apresentado a imagem antes de utilizar o YOLO, e na Figura 35 (b) é apresentado o resultado após passar na rede. Na imagem são apresentadas as caixas que representam as classes treinadas na rede, além das cores para identificar os objetos são associados as caixas o rótulo correspondentes.

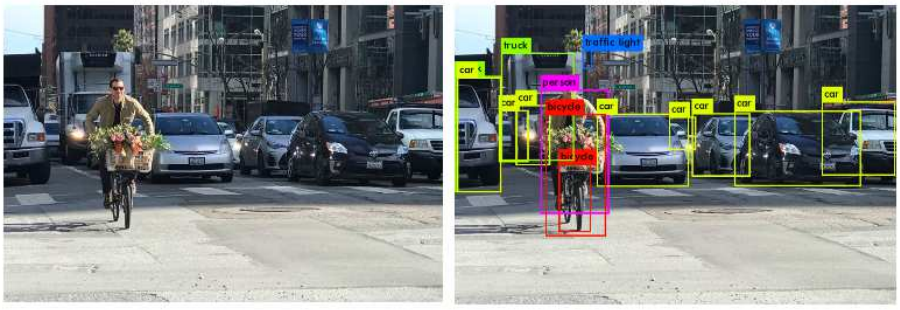

Figura 35 – Exemplo de saída do YOLO

Fonte: Farhadi et al. (2015)

O YOLO inicialmente divide a imagem de entrada em *S* x *S* conhecidos como *grid*, a Figura 36 apresenta a divisão da imagem por 7 x 7. Cada célula do *grid* prediz somente um objeto treinado na rede (FARHADI et al., 2015). Por exemplo o *grid* amarelo, tenta predizer o objeto pessoa no centro da células dos *grids*.

Figura 36 – Funcionamento dos *Grids*

Fonte: Farhadi et al. (2015)

Cada *grid* prediz um número fixo de caixas delimitadoras, elas são responsaveis por englobar o objeto. Na Figura 37 são apresentadas duas caixas delimitadoras azuis, utilizadas para localizar onde está a pessoa. As caixas delimitadoras possuem 5 elementos (x,y,w,h) e o valor de confiança. O valor de confiança diz respeito a probabilidade de uma caixa conter um objeto e sua precisão (FARHADI et al., 2015). Sendo seus outros elementos:

- w – largura da imagem;
- h – altura da imagem;
- x e y – corresponde o deslocamento entre *grids*.

O mapa de característica resultante da rede, pode ser calculado com a seguinte expressão: *(S,S,Bx5 + C)*. O *S* corresponde a divisão da imagem em *grids*, o *B* diz respeito ao número de caixas delimitadoras e *C* o número de classes treinadas.

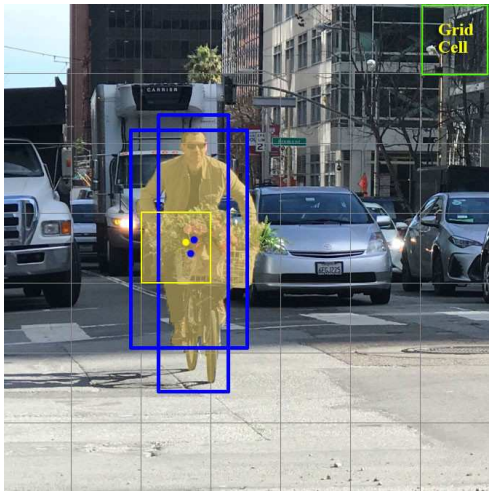

Figura 37 – Duas caixas delimitadoras

Fonte: Farhadi et al. (2015)

Na Figura 38 tem o exemplo da convolução da imagem 448 x 448 x 3 com *grids* de 7 x 7 o seu mapa resultante. Para fazer a predição final o *yolo* coloca um valor *default* de 0.25 em relação do valor de confiança (FARHADI et al., 2015), portanto todos os valores acima desse valor, serão identificados com a classe correspondente.

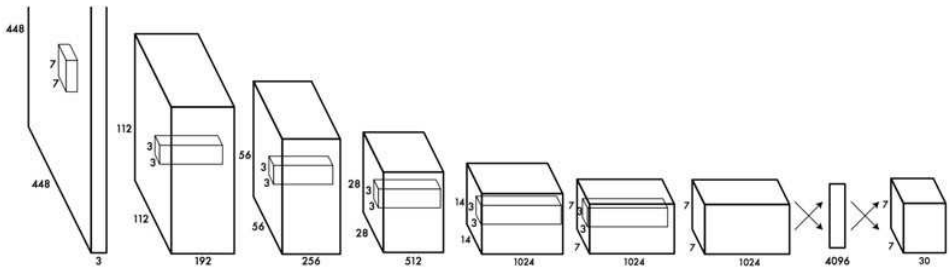

Figura 38 – Estrutura YOLO

Fonte: Farhadi et al. (2015)

O valor de confiança para cada caixa é computado com a sequinte Equação (13):

$$Vc = Vcc * Pc \tag{12}$$

Sendo Vc o valor de confiança, Vcc valor da caixa delimitadora e Pc a probabilidade

da classe. O complemento da Equação (14,15 e 16):

$$Vcc = P_r(objeto) \cdot IoU \tag{13}$$

$$Pc = P_r(classe_i|objeto) \tag{14}$$

$$Vc = P_r(classe_i) \cdot IoU \tag{15}$$

- $P_r(objeto)$ - É a probabilidade da caixa conter um objeto;
- IoU - É a intersecção entre a predição da caixa e *ground truth*, que são as imagens rotuladas;
- $P_r(classe_i|objeto)$ - É a probabilidade do objeto pertencer a uma classe, dado a presença de um objeto;
- $P_r(classe_i)$ - É probabilidade de um objeto pertencer a uma classe.

O YOLO possui 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas, possuem algumas convoluções 1 x 1 utilizadas para reduzir a profundidade dos mapas de caracteristicas (FARHADI et al., 2015), assunto melhor tratado no tópico Googlenet.

2.8.12 Função de perda do YOLO

O YOLO prediz múltiplas caixas delimitadoras por *grid*. Para computar a perda por verdadeiro positivo, é necessário que só uma delas seja responsável pelo o objeto. Nesse propósito é selecionado o maior IoU com o *ground truth* (FARHADI et al., 2015). Essa estratégia leva especialização entre as previsões da caixa delimitadora, em que cada previsão melhora a previsão de determinados tamanhos e proporções da caixa. YOLO usa a soma de erros ao quadrado entre as predições e o *ground truth* para calcular a função de perda (FARHADI et al., 2015). A função de perda é composta por: *Classification loss*, *Localization loss* e *Confidence loss*.

Classification loss: Se um objeto é detectado, o *classification loss* em cada *grid* é o erro ao quadrado das probabilidades condicionais para cada classe, apresentado na Equação 17:

$$\sum_{i=0}^{S} \mathbb{I}_i^{obj} \sum_{c \varepsilon classes} (p_i(c) - \hat{p}_i(c))^2 \tag{16}$$

$\mathbb{I}_i^{obj}$ 1 se um objeto aparecer na célula i, de outra forma é 0.

$p_i(c)$ denota a probabilidade condicional para classe c está na célula i.

Localization loss: Sua função é medir os erros nos locais e tamanhos previstos da caixa delimitadora, apresentado na Equação 18:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2] \tag{17}$$

$\mathbb{I}_{ij}^{obj}$ 1 se o *j* da caixa na célula *i*, é responsável por detectar o objeto, 0 o contrário.

λ_{coord} Aumenta o peso para a perda na coordenada da caixa delimitadora.

A equação busca não ponderar erros absolutos em caixas grandes e caixas pequenas de forma igual, basicamente o erro de 2 pixels em uma caixa grande é igual a de uma caixa pequena (HUI, 2018). Para resolver esse problema, o YOLO prevê a raiz quadrada da largura *w* e da altura *h*. Para finalizar é utilizado o valor de perda por λ_{coord} (*default* valor 5).

Confidence loss: Se um objeto é detectado em uma caixa, o *Confidence loss* apresentada na Equação 19.

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}(C_i-\hat{C}_i)^2 \tag{18}$$

C_i É o valor de confiança da caixa *j* na célula *i*.

Caso o objeto não seja detectado em uma caixa, o *Confidence loss* é apresentada na Equação 20:

$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{noobj}(C_i-\hat{C}_i)^2 \tag{19}$$

$\mathbb{I}_{ij}^{noobj}$ complemento de $\mathbb{I}_{ij}^{obj}$.

λ_{noobj} O peso é menor quando detectar o fundo da imagem.

$\hat{C}_i$ É o valor de confiança da caixa *j* na célula *i*.

Como a maioria das caixas não contém objetos, isso causa um problema de desequilíbrio de classe, ou seja, a rede é mais treinada com *grids* sem objetos. Buscando diminuir esse problema a perda é ponderada por um fator de λ_{noobj} (padrão: 0.5). No final adicionado o *localization*

loss, *confidence loss* e *classification loss* a equação de perda fica:

$$\sum_{i=0}^{S} \mathbb{I}_{i}^{obj} \sum_{c \epsilon classes} (p_i(c) - \hat{p}_i(c))^2 +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] +$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 +$$

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

2.8.13 *Non-maximal suppression*

No processo de localização de um objeto o YOLO pode fazer dupla detecções em um mesmo objeto, isso ocorre porque no processo de divisão em *grids* e procura de centro do objeto, a rede vai gerar mais de um ponto de centro (FARHADI et al., 2015). Na Figura 39 é apresentado um exemplo de rede treinada para identificar carros, inicialmente a imagem é dividida em um *grid* 8x8 sendo procurado o centro do objeto ao *grid* correspondente. No caso a rede não sabe localizar em qual *grid* está o centro do objeto na imagem, ela gera mais de um, como é apresentado na figura, nela o objeto é localizado três vezes.

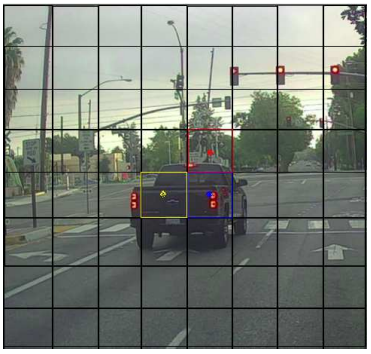

Figura 39 – A rede procurando o centro do objeto

Fonte: Farhadi et al. (2015) - Adaptado

O algoritmo *non-maximal suppression* visa suprimir essa duplicação de centros, basicamente ele remove os caixas delimitadores com baixo valor de confiança (HUI, 2018). Na Figura 40, é apresentado o resultado do algoritmo.

Figura 40 – Resultado do *Non-max-suppression*
Fonte: Farhadi et al. (2015) - Adaptado

A sequência do algoritmo:

1. Classificar as previsões pelo índice de confiança;
2. Inicia-se pelas pontuações mais altas, ignore qualquer previsão atual se encontrar previsões prévias que tenha a mesma classe e $IoU > 0.5$, com a previsão atual;
3. Repetição do passo 2 até que todas as previsões sejam verificadas.

2.8.14 YOLOv2

A segunda versão do YOLO foi desenvolvida visando melhorar a precisão, ao mesmo tempo em que torna mais rápido. Entre suas mudanças esta a não utilização do algoritmo *dropouts*, já que utiliza *batch normalization*, que é uma técnica para melhorar o desempenho e a estabilidade de rede neurais, permitindo que redes profundas mais sofisticadas funcionem na prática (FARHADI et al., 2015). A idéia é normalizar as entradas de cada camada de tal forma que elas tenham uma ativação de saída média de zero e desvio padrão de um (COLLIS, 2017).

Depois de normalizado a entrada da rede neural, não precisa mais se preocupar com a escala de valores ser extremamente diferente. Essa versão também é mais rápida para o treinamento.

2.8.15 Convolução com *anchor boxes*

No processo de início da localização do objeto, o YOLO faz estimativas arbitrárias de caixa delimitadora, esse processo pode funcionar bem em alguns objetos, não em todos como é apresentado Figura 41.

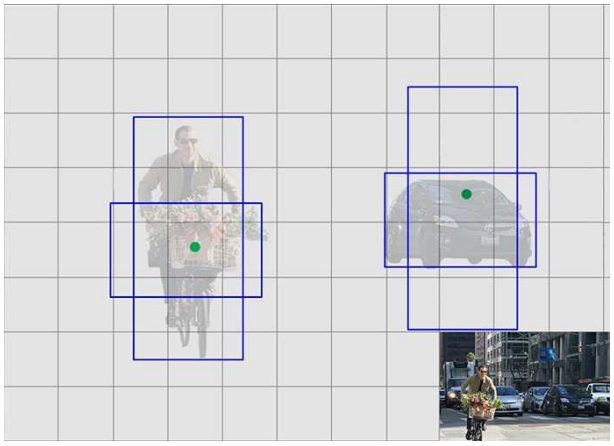

Figura 41 – Mais de uma previsão
Fonte: Farhadi et al. (2015)

No mundo real, a caixa delimitadora não é arbitrária, carros tem contornos muito similares e pedestre uma proporção de aproximada de 0,41 perante ao carro. No caso só é necessário um "chute"para acertar o contorno do objeto, o treinamento será mais estável se começar com diversas suposições comuns a objetos da vida real. Na Figura 42 é apresentado um exemplo de utilização de 5 caixas com diferentes formato.

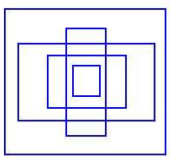

Figura 42 – Mais de uma previsão
Fonte: Farhadi et al. (2015)

No YOLOv2 em vez de prever 5 caixas delimitadoras, a rede preve compensações para cada uma das caixas, outro processo utilizado é restringir os valores de deslocamento (FARHADI et al., 2015). Com isso é mantido uma diversidade das previsões e faz com que cada previsão se concentre em uma forma especifica, tornando o treinamento mais estavel. Nessa versão o YOLO faz uso do algoritmo K-means, para calcular as caixas delimitadoras. O algoritmo é um método de agrupamento que visa particionar *n* observações dentre *k* grupos, em que cada observação pertence ao grupo mais proximo da média (NOGARE, 2017). Como na imagem estão sendo identificadas caixas em vez de pontos, no algoritmo não é possivel utilizar a distância espacial regular para medir as distâncias do ponto de dados. No YOLOv2 é utilizado o IoU.

2.8.16 Previsão da localização

O YOLO prevê 5 parametros *(tx,ty,tw,th e to)* para fazer previsões, além disso aplica a função sigma para restringir o valores fora do *range* da caixa (FARHADI et al., 2015).

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$P_r(objeto) * IOU(b, objeto) = \sigma(t_o)$$

Seus significados:

t_x, t_y, t_w, t_h são predições feitas pelo YOLO;
c_x, c_y é o topo do canto esquerdo do *anchor*;
c_w, c_h é a largura e altura do *anchor*;
c_x, c_y, cw, ch são normalizados pela largura e altura da imagem;
b_x, b_y, b_w, b_h são predições da caixa delimitadora *anchor*;
$\sigma(t_o)$ é o valor de confiança da caixa. Na Figura 43 é apresentada a equação na pratica, com os *anchors* em azul, prevendo a caixa delimitadora tracejada (HUI, 2018).

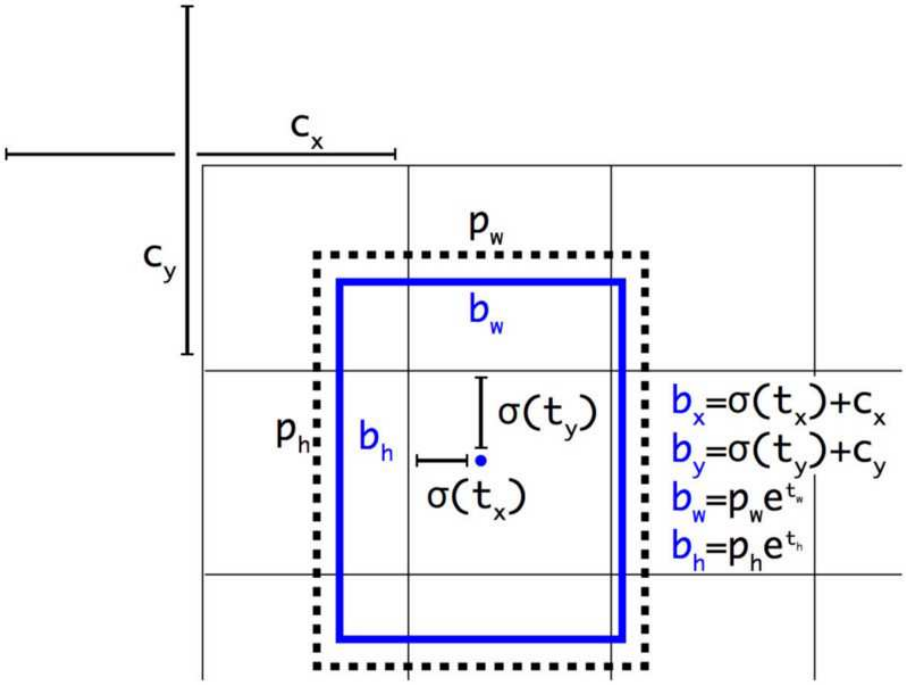

Figura 43 – Equação na prática

Fonte: Redmon (2010)

2.8.17 YOLOv3

A versão mais atual é denominada YOLOv3, entre as mudanças esta classificação multi-label. Por exemplo, os rótulos de saída podem ser “pedestres” e “secundários”, que não são exclusivos. O YOLOv3 substitui a função softmax por classificadores logísticos independentes para calcular a probabilidade de a entrada pertencer a um rótulo específico (FARHADI et al., 2015). Em vez de usar o erro quadrático médio no cálculo da perda de classificação, o YOLOv3 usa função de perda de entropia cruzada binária para cada rótulo (REDMON, 2010). A função entropia cruzada interpreta o sinal de treinamento e as saídas da rede como probabilidades e o algoritmo minimiza a diferença entre estas probabilidades. Isso também reduz a complexidade de computação, evitando a função softmax. O YOLOv3 prevê um valor para cada caixa delimitadora usando a regressão logística. A nova versão faz predição em três escalas, que são dadas precisamente pela redução da resolução das dimensões da imagem de entrada por 32, 16 e 8, respectivamente. Todas essas alterações visaram a

melhoria da precisão do sistema de detecção, no entanto é mais lento que versões anteriores (KATHURIA, 2018).

2.8.18 Googlenet

A Googlenet é uma rede neural convolucional que ficou famosa depois de ganhar o Imagenet 2014, que é uma competição de reconhecimento de imagem, ela é uma rede basicamente composta por 22 camadas de profundidade, onde seus criadores tiveram como objetivo melhorar a utilização dos recursos de computação pela rede ao mesmo tempo que aumenta profundidade e largura dela, onde faz muito uso de filtros convolucionais (SZEGEDY et al., 2015). O principal diferencial da arquitetura Googlenet é o Módulo Inicial, onde sua ideia é cobrir uma área maior, mas manter também uma boa resolução para pequenas informações da imagem, para isso ela utiliza conceito de convolução por volume. Convolução por volume é uma técnica utilizada em redes neurais convolucionais que visa diminuir as dimensões da imagem utilizando só o necessário (NUMPYDL, 2015). A Figura 44 representa a entrada de uma imagem com 6 cm de altura, 6 cm de largura e possui 3 canais de cor é uma 6 x 6 x 3 Figura 44 (A) para diminuir as dimensões da imagem e com isso trabalhar só com parâmetros necessários na rede, é utilizada um filtro com 3 cm de altura e 3 cm de largura e a profundidade do filtro deve ser igual a profundidade da imagem de entrada 3 x 3 x 3, Figura 44 (B). A camada de filtro é passada em toda imagem em 3x3 como mostrado na Figura 44, ao final o resultado é o mapa de características da imagem Figura 44 (C). O filtro é utilizado para identificar elementos da imagem, como foi tratado no tópico de camada convolucional, normalmente é utilizado mais de um filtro por imagem o que implica na alteração no mapa de características. No exemplo foi utilizado apenas um filtro, caso fosse utilizado 6 filtros a nomenclatura do mapa do filtro iria mudar para 4 x 4 x 6. O mapa de característica é calculado com a Equação 21:

$$N_1 x N_2 x N_c * F_1 x F_2 x N_c = (N_1 - F_1 + 1) x (N_2 - F_2 + 1) x N_f \tag{20}$$

O N_1 é altura da imagem, N_2 largura da imagem, N_c número de canais da imagem, F_1 altura do filtro, F_2 largura do filtro e N_f é o número de filtros. Por isso ao aplicar uma imagem 6 x 6 x 3 e aplicar um filtro 3 x 3 x 3 o resultado é 4 x 4 x 1 (NGUYEN, 2018).

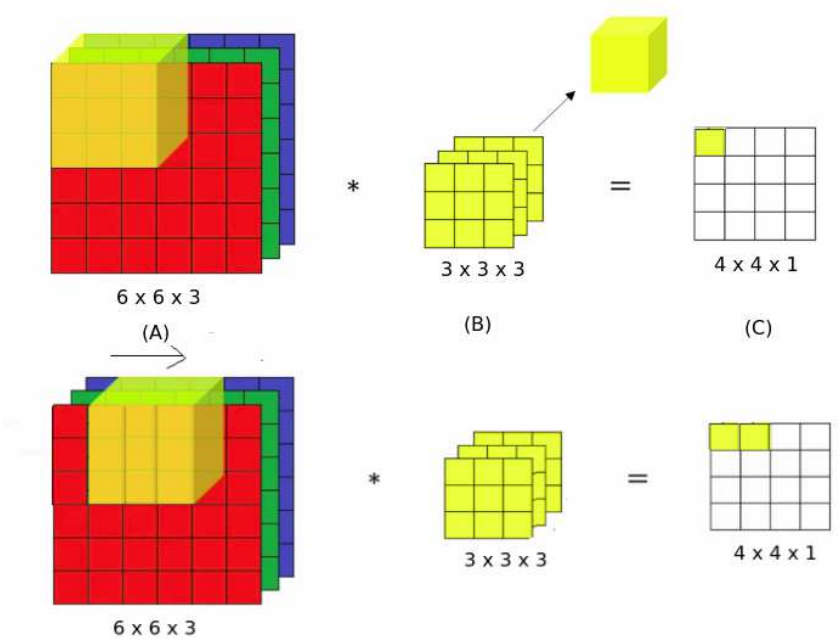

Figura 44 – Convolução por Volume

Fonte: Nguyen (2018)

Esses conhecimentos foram levados em conta na projeção da Googlenet, na criação do primeiro Módulo inicial os engenheiros estavam com dúvidas de qual configuração de filtro utilizar: 5 x 5, 3 x 3, 1 x 1 ou uma camada de *maxpooling* (NGUYEN, 2018). Eles decidiram na época utilizar todas ao mesmo tempo e ao final juntar todos os mapas de característica, deixando a rede decidir qual é o melhor parâmetro como visto na Figura 45. Com isso aumentou a profundidade do mapa de característica 64+128+32+32 = 256 ficando 28 x 28 x 256, mas os engenheiros perceberam que esse modelo tem um alto custo computacional devido ao número de operações de matrizes para gerar o mapa de características (NGUYEN, 2018).

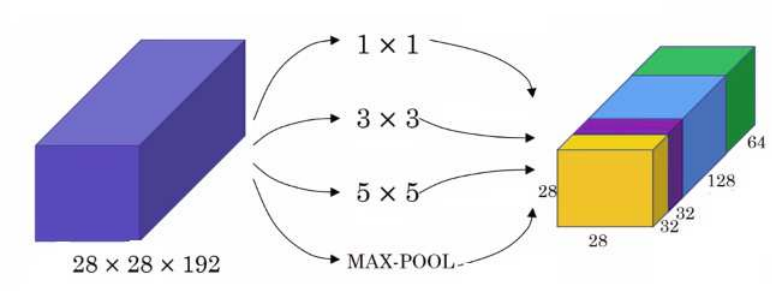

Figura 45 – Módulo Inicial

Fonte: Szegedy et al. (2015)

A solução apareceu na segunda versão do Módulo Inicial, foi notado que as convoluções pesadas como 5 x 5 e 3 x 3 geram um alto custo computacional, o exemplo na Figura 46 nela tem-se uma imagem de 28 x 28 x 192, que vai ser aplicada 32 filtros com dimensão de 5 x 5 x 192. A multiplicação entre as matrizes resultará em: 28 x 28 x 32 x 5 x 5 x 192 = 120.422.400 operações que devem ser realizadas para gerar o mapa de característica, um valor inviável já que esse é apenas um filtro dos muitos que uma rede neural convolucional pode ter. A ideia foi utilizar convoluções 1 x 1 antes de convoluções mais pesadas como: 5 x 5 e 3 x 3 visando diminuir o número de operações (SZEGEDY et al., 2015).

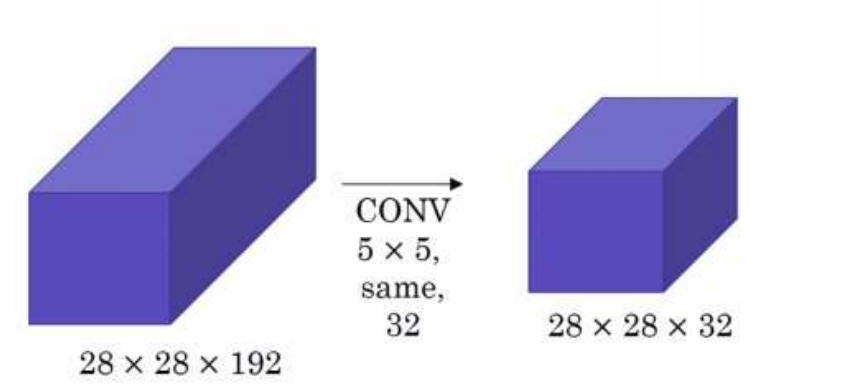

Figura 46 – Custo computacional

Fonte: Nguyen (2018)

Uma convolução 1 x 1 tem utilidade quando se utilizar profundidade como na Figura 47. A convolução 1 x 1 é aplicada em 36 partes diferentes da imagem, multiplicando todos os filtros por canais da imagem, além disso é aplicado a função ReLu junto a convolução, ao final esse produto será somado e originará 36 campos possíveis do mapa de característica Figura 47.

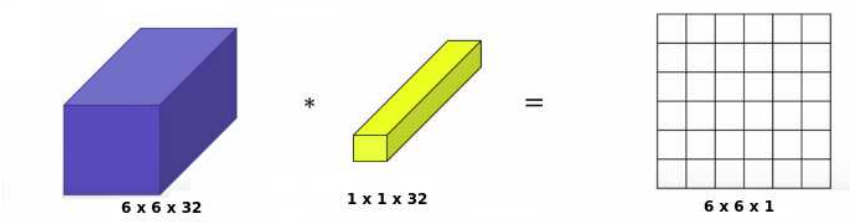

Figura 47 – Convolução 1 x 1

Fonte: Szegedy et al. (2015)

Retornando ao exemplo que gerou 120.422.400 operações, antes de mandar a imagem para um filtro 5 x 5 x 192, ela passa por um filtro 1 x 1 x 192, as dimensões ainda são as mesmas como visto da Figura 48. O custo dessa nova operação é dividido em duas parte, a primeira é a

imagem multiplicada por filtro 1 x 1, no caso diminuir o número de filtros utilizados no caso são 16 filtros de 1 x 1 x 192 e a imagem 28 x 28 x 192. O resultado da primeira etapa é 192 * 28 * 28 * 192 = 2.408.448 operações, a segunda etapa é o mapa de característica 28 x 28 x 32 resultante por 32 filtros de 5 x 5 x 16 o resultado é: 28 * 28 * 32 * 5 * 5 * 16 = 10.035.200. Juntando as duas partes: 2.408.488 + 10.035.200 = 12.443.648 operações, com a convolução 1 x 1 houve uma redução de 107.978.752 operações que o computador tem que fazer.

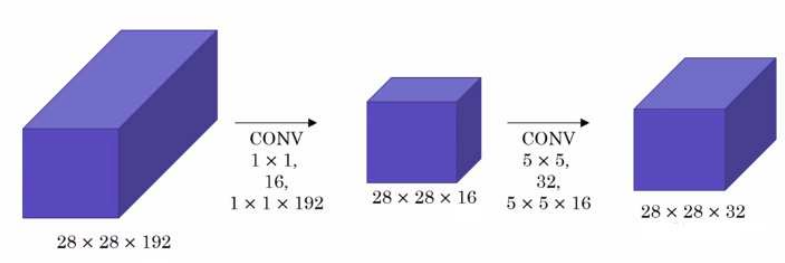

Figura 48 – Convolução 1 x 1 com 5 x 5

Fonte: Nguyen (2018)

Com essa nova configuração o Módulo inicial ficou como da Figura 49, com convoluções 1 x 1 antes das 3 x 3 e 5 x 5, em paralelo se manteve uma camada de *maxpooling* e uma camada de convolução 1 x 1, ao final todos os mapas de características são juntos como no primeiro Módulo inicial (NGUYEN, 2018).

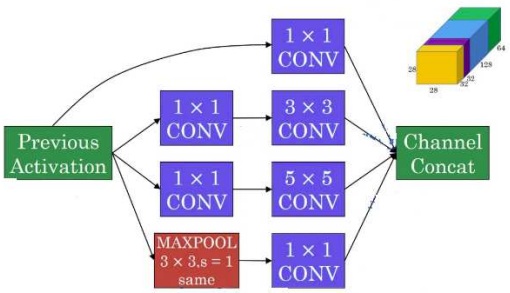

Figura 49 – Novo módulo inicial

Fonte: Nguyen (2018)

Na Figura 50 é mostrado a arquitetura Googlenet em que o círculo vermelho diz respeito aos nove Módulo inicial da rede, é uma rede com 22 camadas apenas considerando as convolucionais com parâmetros. Tem-se uma camada linear com Softmax como classificador que vai normalizar a saída entre 0 e 1, e a função de ativação utilizada é a ReLU (*Rectified Linear Unit*) que varia do 0 ao infinito, e possui um aprendizado rápido e custo computacional mais baixo (VARGAS et al., 2015).

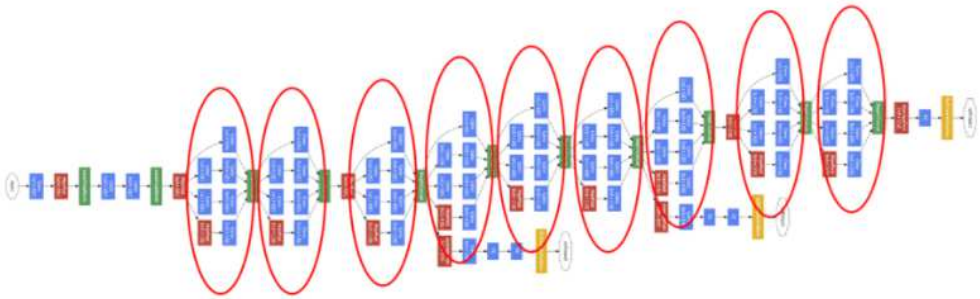

Figura 50 – Exemplo de Topologia Googlenet

Fonte: Santos (2017)

2.9 AVALIADOR DE CLASSIFICADORES

Os resultados das redes YOLO e Googlenet foram submetidos a uma matriz de confusão, dela foi calculado a cobertura e a precisão do modelo. A matriz de confusão é um método estatístico que permite avaliar e visualizar o desempenho de um algoritmo (WITTEN; FRANK, 2011). Um exemplo de matriz de confusão pode ser visto na Tabela 2, em que VP (verdadeiro positivo), FP (falso positivo), FN (falso negativo) e VN (verdadeiro negativo) .

Tabela 2 – Exemplo matriz de confusão

		Real	
		Classe A	Classe B
Predito	Classe A	VP	FP
	Classe B	FN	VN

Fonte: (WITTEN; FRANK, 2011)

O VP é o número de classes reais preditas, FP é o número de classes reais não encontradas, FN é o número de vezes que a classe A foi dada como outro objeto e VN é o número de objetos que não pertencem a classe A e não foram preditos como sendo dela. Os valores permitiram calcular a precisão e a cobertura. A precisão é a razão entre o número de classe corretamente classificadas pela soma delas e as classes não encontradas, Equação 22. A cobertura é encontrada pela razão das classes corretamente classificadas pela soma total delas no modelo, Equação 23. Determinado a precisão e a cobertura, já é possível calcular a média harmônica Equação 24. A média harmônica é uma média mais precisa que a média aritmética

simples (FACELI et al., 2011), o que torna os resultados mais seguros.

$$Precisão = \frac{VP}{VP + FP} \tag{21}$$

$$Cobertura = \frac{VP}{VP + FN} \tag{22}$$

$$MédiaHarmônica = \frac{2 * (Precisão * Cobertura)}{Precisão + Cobertura} \tag{23}$$

2.10 TRABALHOS CORRELATOS

Em problemas de classificação como identificação espécies florestais tem-se buscado muitas soluções utilizando o *deep learning*, devido aos seus resultados. Existem também metodologias de trabalho diferentes do *deep learning*, como o uso de técnicas de processamento de imagens e reconhecimento de padrões. Essas técnicas fazem uso de características macroscópica e microscópicas entre os exemplos tem-se o trabalho que utiliza a madeira como objeto de estudo aplicando técnicas de extração de características em imagens do exemplar, as técnicas utilizadas foram: análise de cor, GLCM (*Gray-Level Co-Occurrence Matrix*), histograma de borda, Fractais, LBP (*Local Binary Pattern*), LPQ (*Local Phase Quantization*) e Gabor, a técnica resultou em uma taxa de reconhecimento de 99,49% (FILHO, 2012).

Outro trabalho com tema relacionado, é o que propõe analise e identificação baseada a extração de características de textura a partir de imagens microscópicas de epiderme da folha (ODEMIR et al., 2017). Foram utilizadas 32 espécies, fazendo uso de técnicas de extração de textura. Ao final a abordagem obteve 96% de taxa de sucesso. Abordando outro elemento da espécie que é a folha (PIRES, 2017). Através das folhas são aplicadas técnicas de segmentação na imagem, para depois ser aplicada extração de características, foi utilizado a técnica GLCM para essa tarefa, no total foram retirados 30 descritores. Os classificadores utilizados foram MLP (*Multilayer Perceptron*), SMO (*Sequential Minimal Optimization*) e o LibSVM (*Library for Support Vector Machines*) a técnica obteve 75,4% de reconhecimento (PIRES, 2017).

Utilizando o *deep learning* como método principal de identificação tem-se trabalhos que basicamente utilizam as redes neurais convolucionais para identificar as melhores característica da folha para reconhecer uma espécie (MAYO; REMAGNINO, 2016), no experimento foi utilizada a topologia de rede neural Imagenet, foi trabalhado com as

características venação, textura e contornos da folha, foi verificado que o contorno sozinho não é um bom atributo para determinar a espécie, ao contrário da venação da folha que se mostrou um atributo mais influente para identificar espécies. Analisando ainda a folha tem-se o trabalho que usa redes convolucionais para controle de ervas daninhas, que visa detectar essa espécie no gramado, foi utilizado imagens com 256x256 pixels e a arquitetura utilizada foi a AlexNet o resultado foi de 75% de precisão (PEARLSTEIN et al., 2017).

Tem-se também trabalhos utilizando rede neurais convolucionais mas com dados de grande dimensão, imagens tiradas do alto de fazendas, o que exigiu um detalhe a mais em relação a iluminação exigindo um pré-processamento na imagem para ser mandada a rede, sendo composta de 5 camadas convolucionais sendo utilizada a função de ativação ReLU e ao final aplicada a função Softmax, o experimento obteve precisão de 97,47% (YALCIN; RAZAVI, 2016).

3 MATERIAS E MÉTODOS

Nesse capítulo serão definidas as ferramentas e os procedimentos a serem utilizados para o desenvolvimento da pesquisa, essa etapa será dividida entre dois tópicos principais ferramentas e sua utilização.

3.1 FERRAMENTAS

Com as imagens das folhas adquiridas foram utilizadas redes neurais convolucionais para realizar a classificação. No projeto foi aplicado dois modelos de redes: o modelo de rede utilizando o Googlenet e o YOLO ambos tratados na fundamentação teórica.

Para essa tarefa foi utilizado o Tensorflow[1]. O Tensorflow é uma biblioteca de código aberto, desenvolvido pela Google, destinada a inteligência artificial, em que todo o projeto ocorreu através de estudos de redes neurais. A biblioteca utiliza o aprendizado de máquina, permitindo que as máquinas aprendam com as ações e ampliem sua capacidade de decisão. A ferramenta está licenciada pela Apache 2.0 e disponível no GitHub[2], foi projetada inicialmente para trabalhar com Python e C++. A arquitetura flexível permite implantar computação para uma ou mais CPUs ou GPUs em uma área de trabalho, servidor ou dispositivo móvel com uma única API (TENSORFLOW, 2017), foi utilizada a versão 1.0 do Tensorflow.

Para fazer uso do Tensorflow foi utilizado o Python[3], que é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Entre as características que comprovam esses objetivos tem-se: baixo uso de caracteres especiais, o uso de indentação para marcar blocos, quase nenhum uso de palavras-chave voltadas para a compilação e coletor de lixo para gerenciar automaticamente o uso da memória. Python suporta múltiplos paradigmas de programação, além de ter uma

[1] https://www.tensorflow.org/
[2] https://github.com/
[3] https://www.python.org/

grande biblioteca, que contém classes, métodos e funções. E por último é importante dizer que Python é uma linguagem multiplataforma (PYSCIENCE-BRASIL, 2017).

No caso será utilizado o Python 3 devido a sua compatibilidade com o Tensorflow. Para trabalhar com Python e visualizar dados será utilizado o Jupyter notebook[4], que é uma aplicação web de código aberto que permite criar e compartilhar documentos que contenham código, equações, visualizações e texto narrativo. Os usos incluem: limpeza e transformação de dados, simulação numérica, modelagem estatística, visualização de dados, aprendizado automático de máquinas e etc, sua interface pode ser vista na Figura 51.

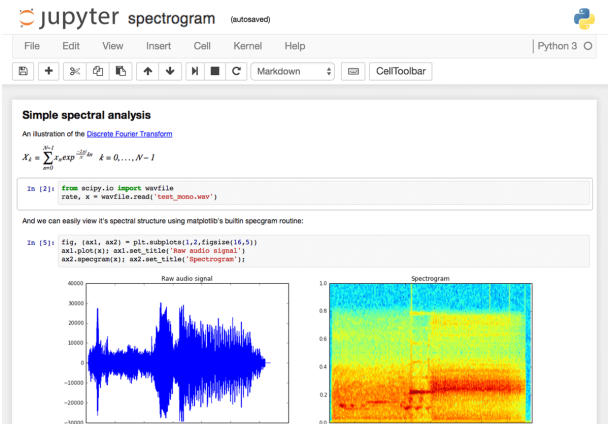

Figura 51 – Interface do Jupyter

Fonte: Jupyter (2017)

Foi utilizado também o NumPy[5] que é um pacote voltado para computação científica, que permite trabalhar com arranjos, vetores e matrizes de N dimensões, de uma forma comparável e com uma sintaxe semelhante ao software proprietário Matlab (PYSCIENCE-BRASIL, 2017), será utilizado para trabalhar com os valores gerados pelas imagens (NUMPY, 2017). Visando facilitar a instalação e o uso de bibliotecas foi utilizado o Anaconda[6]. O Anaconda é uma plataforma de ciência de dados que faz uso do Python, na parte de IDEs ela vem com o Jupyter, Spyder, Jupyterlab e RStudio. Na parte de análise de dados, com ela vem NumPy, SciPy, Numba, Pandas e DASK. Já para tratar da visualização ela contêm o Bokeh, HoloViews, Datashader e o Matplotlib e por fim para aprendizado de máquina tem-se o próprio TensorFlow, H20.ai, Theano e etc (ANACONDA, 2017). O sistema operacional utilizado foi

[4] http://jupyter.org/
[5] http://www.numpy.org/
[6] https://www.anaconda.com/download/

o [7]Ubuntu 16.4, que é um sistema operacional de código aberto, construído a partir do núcleo Linux, baseado no Debian (outra partição do linux) (UBUNTU, 2017).

3.1.1 Hardware

Foi utilizado um notebook com processador intel core 5, 4 GB de memória RAM e 500 GB HD. Sendo feito uso de um servidor contido no laboratório de redes da Universidade Tecnológica Federal do Paraná campus Medianeira. Esse servidor possui 16 GB de RAM e um 1 TB (Terabyte) de HD.

3.2 UTILIZAÇÃO DAS FERRAMENTAS

A abordagem utilizada (Figura 52) no projeto, foi inicialmente fazer a aquisição de imagens. A segunda etapa é fazer o aumento da base de imagens com algoritmos, visando melhorias no treinamento, após essa etapa é realizado treino das redes neurais. Com os treinos concluídos, é feita a análise dos resultados para melhorar o desempenho do modelo e apresentá-lo.

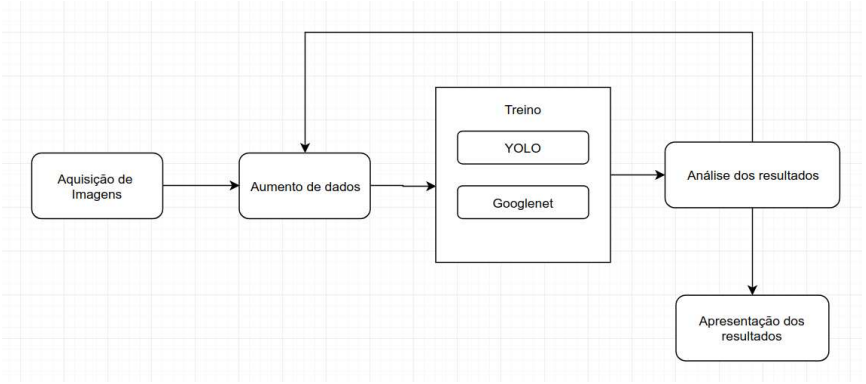

Figura 52 – Metodologia

Fonte: Autoria Própria

[7]https://www.ubuntu.com/download

3.2.1 Aquisição de Imagens

Para fazer o reconhecimento das espécies pela folha, são necessárias imagens dos exemplares e em grande quantidade. Todos os exemplares utilizados foram coletados na Universidade Tecnológica Federal do Paraná câmpus Medianeira, devido sua facilidade de coleta e acesso ao responsável pela plantação, o professor Agostinho Zanini do departamento de ciências biológicas e ambientais. Ao todo foram coletadas 29 espécies sendo tiradas 100 fotos frente e verso das folhas, Tabela 3. A numeração da Tabela 3 diz respeito a localização das espécies nas Figuras 53 e 54.

Figura 53 – Local 1

Fonte: Maps (2018)

Figura 54 – Local 2

Fonte: Maps (2018)

Tabela 3 – Tabela com a quantidade de imagens por espécie.

Número	Nome Cientifico	Nome Popular	Qtd de Folhas
1	*Persea Americana*	Abacateiro	100
2	*Eriobotrya Japnica Lind*	Ameixa de inverno	100
3	*Psidium Rufum*	Araça Roxo	100
4	*Annona Montana*	Araticum	100
5	*Annona Squamosa*	Atemoia	100
6	*Cojoba Arborea*	Brinco de indio	100
7	*Coffea*	Cafeeiro	100
8	*Pera Heteranthera*	Cafezinho	100
9	*Anacardium Occidentale*	Cajueiro	100
10	*Peltophorum dubium*	Canafistula	100
11	*Nectandra Megapotamica*	Canelinha	100
12	*Cerasus*	Cerejeira	100
13	*Prunus Serrulata*	Cerejeira Japão	100
14	*Salix Babylonica*	Chorão	100
15	*Lle Paraguariensis*	Erva mate	100
16	*Annona Coriácea*	Fruta do conde	100
17	*Psidium Guajava*	Goiabeira	100
18	*Annona Muricata*	Graviola	100
19	*Syzygium Cumini*	Jambolão	100
20	*Leucaena Leucocephala*	Leucena	100
21	*Citrus Limon*	Limão bergamota	100
22	*Tibouchina Mutabilis*	Manacá da serra	100
23	*Brunfelsia Uniflora*	Manacá de cheiro	100
24	*Mangifera Indica*	Mangueira	100
25	*Licania Tomentosa*	Oiti	100
26	*Dypsis Lutescens*	Palmeira areca	100
27	*Paubrasilia Echinata*	Pau Brasil	100
28	*Aspidosperma Polyneuron*	Peroba rosa	100
29	*Eugenia uniflora*	Pitanga	100
	Total		2900

Fonte: Autoria própria

As imagens estão salvas no padrão .jpg com o nome popular seguido de quatro digítos (goiabeira0001.jpg) com 4608 pixels de largura e 3456 pixels de altura. As imagens do projeto contém uma escala junto a folha como mostrado na Figura 55, essa escala é utilizada para controlar cores e tamanho da imagem.

Figura 55 – Folha da Goiabeira
Fonte: Autoria própria

Na aquisição das imagens, foi utilizado o estúdio de fotografia, apresentado por (PIRES, 2017). Apresentado na Figura 56. Suas dimensões são de 40 centímetros quadrados, e em suas laterais internas foram colados três segmentos de fitas de led, que produzem alta luminosidade com pouca energia, assim sendo possível usar uma bateria, ou uma fonte de 12 volts para alimentação elétrica podendo ser transportada com facilidade. Para evitar reflexos nas imagens as paredes da caixa foram pintadas de preto com exceção do fundo que é branco, em que a folha é posicionada é comprimida por uma chapa de vidro, para mantê-la plana e fixa.

Figura 56 – Caixa para fotografia de folhas
Fonte: Pires (2017)

Seguindo a numeração das espécies florestais apresentadas na Tabela 3, a Figura 57 apresenta a base de imagens.

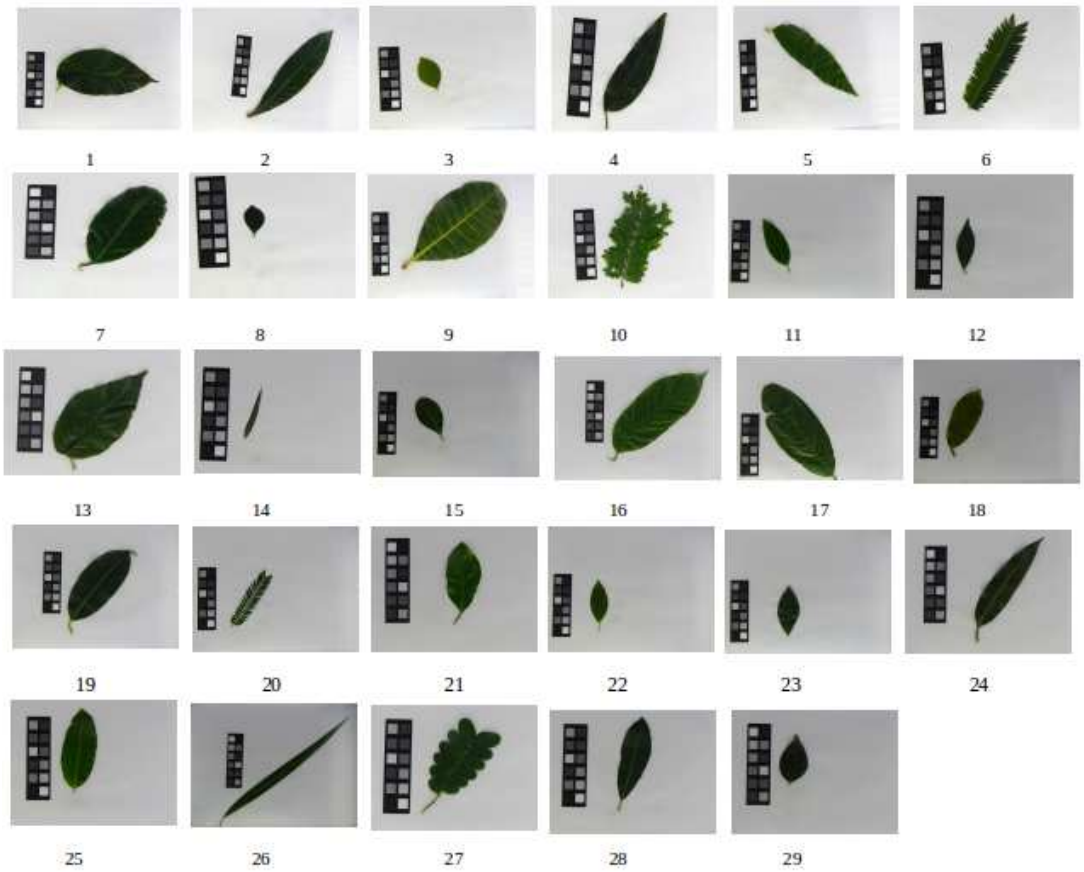

Figura 57 – Base de imagens

Fonte: Autoria própria

3.2.2 Aumento de dados

Para melhorar a taxa de acerto das redes neurais convolucionais é necessário um grande número de imagens sobre cada classe. Esse fato se torna um problema, já que a retirada de folhas em larga escala pode danificar a espécie. Buscando contornar esse problema foi desenvolvido um *script* que aumenta a base, criando novas imagens fazendo alterações na imagem original. O *script* foi criado utilizando linguagem python 3.5.2 e a biblioteca keras. Em relação ao keras foi importado a classe ImageDateGenerator, que gerar dados de imagem em lote em tempo real, utilizando um exemplar (KERAS, 2016). As alterações nas imagens são feitas nos parâmetros da classe, elas são:

- *Rotatation range* - O valor em graus (0-180), um intervalo para girar aleatoriamente imagens;

- *Width shift range* e *height shift* - Os intervalos dentro dos quais se podem deslocar imagens na vertical ou na horizontal;
- *Shear range* - Os intervalos para aplicar aleatoriamente transformações de cisalhamento, uma transformação que desloca cada ponto em uma direção fixada, por um valor proporcional à sua distância com sinal de uma reta que é paralela à direção;
- *Zoom range* - Os intervalos para aplicar aleatoriamente *zoom* nas imagens;
- *Horizontal and Vertical flip* – Inverter metades das imagens;
- *Zoom range* - Os intervalos para aplicar aleatoriamente *zoom* nas imagens;
- *Fill mode* - Uma estratégia usada para preencher os pixels recém-criados, que podem aparecer após uma rotação ou uma alteração de largura / altura.

Um exemplo pode ser visto na Figura 58, passado a imagem original da goiabeira (a) sendo criado randomicamente quatro imagens.

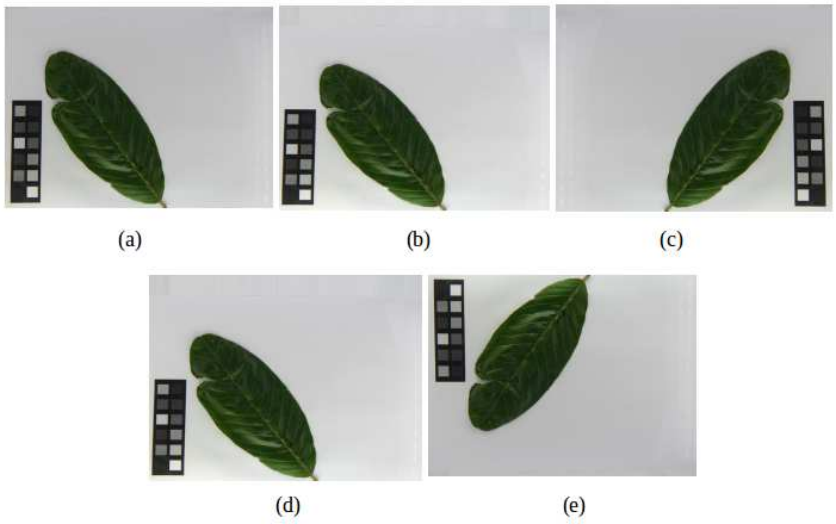

Figura 58 – Exemplo da geração das imagens

Fonte: Autoria própria

3.2.3 Treino

Nessa etapa serão apresentados os parâmetros e metodologias utilizados para a realização do treino da rede YOLO e Goooglenet. Foram realizados quatro treinos, visando a melhoria do desempenho das redes.

3.2.3.1 Googlenet

Seguindo o artigo publicado Szegedy (SZEGEDY et al., 2015) e Agarwal (AGARWAL, 2017), foi desenvolvido um *script* em python da Googlenet. Nessa tarefa inicialmente houve uma alteração na base de imagens, foram criados dois arquivos no padrão csv (treino.csv e validacao.csv). No arquivo csv cada coluna corresponde um pixel da imagem, sendo a primeira coluna o rótulo da imagem, Figura 59.

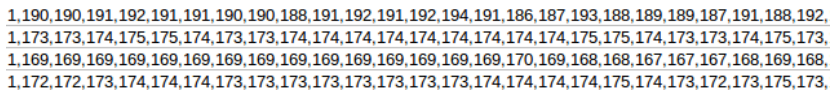

Figura 59 – Abacateiro no padrão csv

Fonte: Autoria própria

Para o desenvolvimento da rede foram utilizados os seguintes parâmetros:

- *Batch size* - Define o número de amostras que serão propagadas pela rede;
- *Reduce 1x1* - Número de mapas de caracteristica para cada convolução que precede uma convolução maior;
- *Dropout* - Taxa que seleciona neurônios da rede que serão ignorados.

A rede foi desenvolvida com 9 módulos inicias, com *Batch size* de 50, *Reduce 1x1* de 104 e taxa de *Dropout* de 0.5. Os parâmetros utilizados são menores que os apresentados no artigo (SZEGEDY et al., 2015), visando um treinamento mais rápido mesmo com uma precisão menor, já que não foi utilizado GPUs em todo processo. A representação da arquitetura pode ser vista na Figura 60. Na primeira coluna são apresentados os tipos de convolução que se dividem entre camadas de módulo iniciais, *pooling* e *softmax*. A segunda coluna apresenta o tamanho dos filtros e os *strides* que são as distâncias entre duas posições consecutivas do processo de convolução (PACHECO, 2017). Por último a coluna de mapas características gerados em cada camada de convolução (SZEGEDY et al., 2015).

Tipo	filtros/ stride	tamanho da saída
convolution	7×7/2	112×112×64
max pool	3×3/2	56×56×64
convolution	3×3/1	56×56×192
max pool	3×3/2	28×28×192
inception (3a)		28×28×256
inception (3b)		28×28×480
max pool	3×3/2	14×14×480
inception (4a)		14×14×512
inception (4b)		14×14×512
inception (4c)		14×14×512
inception (4d)		14×14×528
inception (4e)		14×14×832
max pool	3×3/2	7×7×832
inception (5a)		7×7×832
inception (5b)		7×7×1024
avg pool	7×7/1	1×1×1024
dropout (40%)		1×1×1024
linear		1×1×1000
softmax		1×1×1000

Figura 60 – Estrutura da Googlenet

Fonte: Szegedy et al. (2015)

3.2.3.2 YOLO

No caso do YOLO foi utilizado *Darkflow* que é uma implementação em python que permitir o uso do *framework Darknet* no tensorflow (JONES, 2018). O *Darknet* é um *Framework open source* para treinamento de redes neurais desenvolvido em python e CUDA. No *Darknet* é opcional o uso do CUDA e do OpenCV. O CUDA é uma plataforma de computação paralela desenvolvida pela NVIDIA (CUDA, 2018). Diferente da Googlenet para realizar o treinamento da rede do YOLO foi necessário retirar o recorte das coordenadas do objeto a ser treinado. Para isso foi desenvolvido um *script* em python responsável por gerar um arquivo do tipo xml com tamanho da imagem, coordenadas do objeto e a classe correspondente, Figura 61.

```
- <annotation>
    <folder>teste</folder>
    <filename>Abacateiro_0_55.jpg</filename>
    <segmented>0</segmented>
  - <size>
      <width>640</width>
      <height>480</height>
      <depth>3</depth>
    </size>
  - <object>
      <name>Abacateiro</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>74</xmin>
        <ymin>31</ymin>
        <xmax>581</xmax>
        <ymax>442</ymax>
      </bndbox>
    </object>
  </annotation>
```

Figura 61 – Exemplo de recorte

Fonte: Autoria Própria

Em relação a estrutura da rede escolhida foi a tiny-YOLO versão 2 lançado em 2016, devido a sua velocidade de treinamento, mais rapido que o YOLOv3. O YOLO versão 2 tem entre seu diferencial perante as demais versões o número menor de camadas, resultando em uma melhorar no desempenho da rede. Sua arquitetura é baseada na rede Darknet-19, sua arquitetura é composta por 19 camadas convolucionais intercaladas com mais 5 camadas que aplicam o *max-pooling* (REDMON, 2010). Nas duas redes existe a presença da camada *Softmax* que visa normalizar entre 0 e 1 a saída em problemas de multiclasses, quanto mais proximo de 1 melhor a resposta do modelo em relação a entrada. A estrutura da rede é vista na Figura 62.

Tipos	Filtros	Stride/ tamanho	Saída
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Figura 62 – Estrutura do YOLO

Fonte: Redmon (2010)

A última camada do YOLO fica responsável pela classificação dos objetos, antes de iniciar o treinamento essa camada foi alterada seguindo a Equação 25:

$$filtros = (classe + 5) * 5 \tag{24}$$

Como foram utilizadas 29 classes o número de filtros utilizados na última camada foram 170, o *stride* de tamanho 1 e o *batch size* de 64, seguindo o padrão do *Tiny-Yolo* versão 2.

4 RESULTADOS E DISCUSSÃO

Neste capítulo será apresentado o resultado especifico de cada rede, apresentando exemplares que tiveram melhores resultados na própria rede e entre elas.

4.0.0.1 YOLO

Com a base de imagens e recortes no servidor, foi iniciado o treinamento. A primeira rede treinada foi o YOLOv2 com 5800 imagens, sendo 580 para validação. No treino foram utilizadas 5000 iterações, o treino levou sete dias para ser realizado, apresentando VP = 414, FP e FN igual a 166, a média harmônica ficou em 71,3%, Figura 63.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	7			13																									
2		5		9																								6	
3			10												10														
4				15														5											
5					10											10													
6						15																					5		
7							8									4	4	4											
8								14									6												
9									20																				
10										20																			
11											15												5						
12												13											7						
13													15								5								
14														20															
15	6														14														
16				3												17													
17							6										14												
18																4		16											
19	8																		12										
20																				20									
21							7														13								
22																						20							
23										8													12						
24																3								17					
25																						4			10			6	
26																										20			
27																			4								16		
28										6																		14	
29			5									3																	12

Figura 63 – Matriz de confusão 1 - YOLO

Fonte: Autoria Própria

Visando melhorar os resultados foram aumentados o número de imagens, passando de 5800 para 34800 com 2900 para validação. Na rede do YOLOv2 o número de iteração foi aumentado para 18000. Os valores apresentados pela rede foram: VP = 2502, FP e FN igual 398. A média harmônica ficou em 86,2%. Os valores de precisão, cobertura e média harmônica também foram calculados por classe na Tabela 4, Figura 64.

Tabela 4 – precisão, cobertura e média harmônica em detalhes - YOLO

	Nome Cientifico	**Nome Popular**	**Precisão**	**Cobertura**	**M. Harmônica**
1	*Persea Americana*	Abacateiro	63,0%	64,2%	72,6%
2	*Eriobotrya Japnica Lind*	Ameixa de inverno	57,0%	100%	72,6%
3	*Psidium Rufum*	Araça Roxo	100%	80,0%	88,8%
4	*Annona Montana*	Araticum	96,0%	56,1%	70,8%
5	*Annona Squamosa*	Atemoia	97%	100%	98,4%
6	*Cojoba Arborea*	Brinco de indio	100%	100%	100%
7	*Coffea*	Cafeeiro	55,0%	55,0%	55,0%
8	*Pera Heteranthera*	Cafezinho	95,0%	100%	97,4%
9	*Anacardium Occidentale*	Cajueiro	100%	100%	100%
10	*Peltophorum dubium*	Canafistula	100%	84,0%	91,3%
11	*Nectandra Megapotamica*	Canelinha	98%	100%	98,9%
12	*Cerasus*	Cerejeira	83,0%	89,2%	86,0%
13	*Prunus Serrulata*	Cerejeira Japão	78,0%	100%	87,6%
14	*Salix Babylonica*	Chorão	100,0%	100%	100%
15	*Lle Paraguariensis*	Erva mate	81,0%	100%	89,5%
16	*Annona Coriácea*	Fruta do conde	94%	80,3%	86,6%
17	*Psidium Guajava*	Goiabeira	83,0%	76,8%	79,8%
18	*Annona Muricata*	Graviola	98%	91,5%	94,6%
19	*Syzygium Cumini*	Jambolão	84,0%	85,7%	84,8%
20	*Leucaena Leucocephala*	Leucena	100%	100%	100%
21	*Citrus Limon*	Limão bergamota	72%	76,5%	74,2%
22	*Tibouchina Mutabilis*	Manacá da serra	100%	83,3%	90,9%
23	*Brunfelsia Uniflora*	Manacá de cheiro	81,0%	81,0%	81,0%
24	*Mangifera Indica*	Mangueira	97%	100%	98,4%
25	*Licania Tomentosa*	Oiti	61,0%	100%	75,7%
26	*Dypsis Lutescens*	Palmeira areca	100%	100%	100%
27	*Paubrasilia Echinata*	Pau Brasil	86,0%	100%	92,4%
28	*Aspidosperma Polyneuron*	Peroba rosa	78,0%	72,2%	75,0%
29	*Eugenia uniflora*	Pitanga	65,0%	100%	88,8%

Fonte: Autoria Própria

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	63			37																									
2		57		32																								11	
3			100																										
4				96														4											
5					97											3													
6						100																							
7							55									20	20	5											
8								95									5												
9									100																				
10										100																			
11											98												2						
12												83											17						
13													78								22								
14														100															
15	19														81														
16				6												94													
17							17										83												
18																2		98											
19	16																		84										
20																				100									
21							28														72								
22																						100							
23										19													81						
24																3								97					
25																						20			61			19	
26																										100			
27																			14								86		
28										22																		78	
29			25									10																	65

Figura 64 – Matriz de confusão 2 - YOLO

Fonte: Autoria Própria

Analisando a tabela 4 e a matriz de confusão (Figura 64), é possivel determinar as três piores espécies classificadas: Cafeeiro (7), Oiti (25), Ameixa de inverno (2). O Cafeeiro só classificou 55 imagens de 100 na classe correta, as outras 45 imagens se dividiram em três classes: Fruta do conde (16), Goiabeira (17) e Graviola (18). Devido ao grande número de falso negativo (FN) seu valor de cobertura também é baixo, 55,0%. O caso Ameixa de inverno é semelhante, apresentou um valor baixo de precisão = 57,0%, mas nenhuma espécie foi associada a ela, cobertura = 100%. Foram classificadas 32 imagens como Araticum e 11 como Peroba rosa (28), no lugar de Ameixa de inverno. A principal conclusão que se chegou, foi que o resultado baixo se deu principalmente pela semelhança das folhas envolvidas, Figura 65.

No caso do Oiti ocorreu o erro de 21 imagens serem classificadas como Manacá da serra (22) e 19 como Peroba rosa. A média harmônica do Oiti ficou acima de 72%, diferente da Ameixa de inverno e Cafeeiro, o Oiti também não teve nenhuma espécie associada a ela.

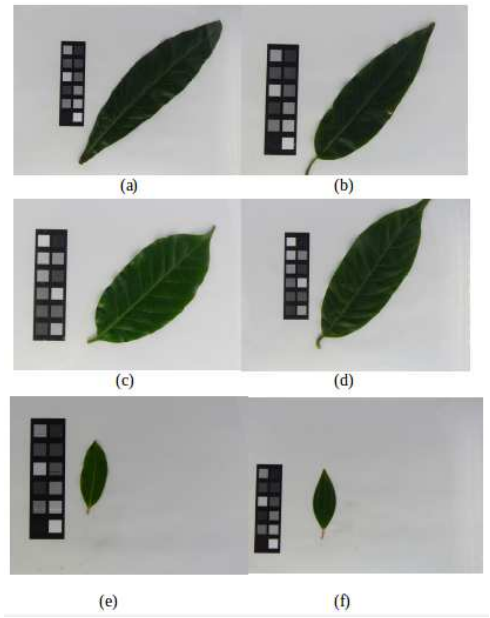

Figura 65 – (a) - Ameixa de inverno, (b) - Araticum, (c) - Cafeeiro, (d) - Fruta do conde, (e) - Oiti, (f) - Manacá da serra

Fonte: Autoria Própria

No YOLO tiveram 8 espécies que apresentaram 100% de precisão e média harmônica acima de 90%: Araça roxo, Brinco de indio, Cajueiro, Canafistula, Chorão, Leucena, Manacá da serra e Palmeira areca. Entre essas espécies o Araça roxo e a Canafistula foram as que apresentaram o menor valor de cobertura 80,0% e 84,0%, respectivamente, devido a um maior valor de espécie classificadas como elas (25 e 19). Todas as espécies podem ser vistas na Figura 66.

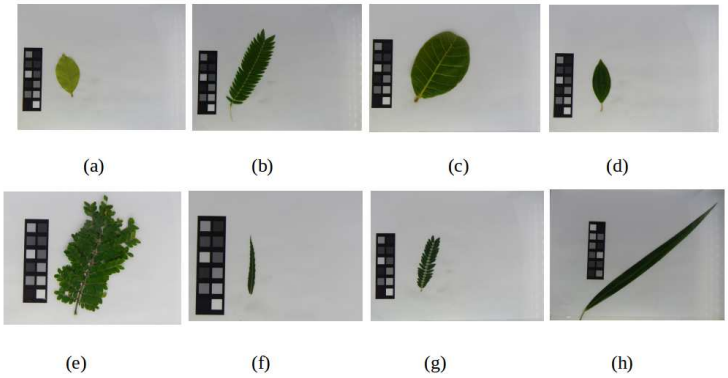

Figura 66 – (a) - Araça roxo, (b) - Brinco de indio, (c) - Cajueiro, (d) - Canafistula, (e) - Chorão, (f) - Leucena, (g) - Manacá da serra, (h) - Palmeira areca

Fonte: Autoria Própria

As demais espécies apresentaram valor de precisão acima ou igual a 63%, sendo

elas: Araticum, Cafezinho, Cerejeira, Cerejeira japão, Erva mate, Goiabeira, Jambolão, Limão bergamota, Manacá de cheiro, Pau brasil, Peroba rosa e Pitanga. No YOLO foram feitos dois grandes treinos, já apresentados no tópico anterior. Nesse modelo foi possível utilizar uma das aplicações do YOLO, que é reconhecer a posição do objeto na imagem, na Figura 67 é mostrado o reconhecimento da espécie com pior resultado do modelo.

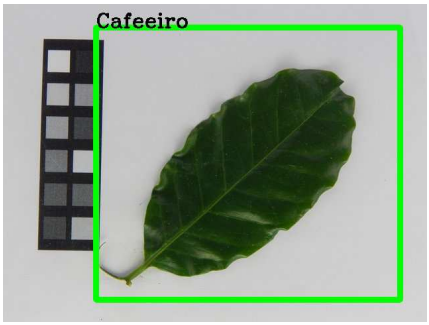

Figura 67 – Reconhecimento do Cafeeiro

Fonte: Autoria Própria

4.0.0.2 Googlenet

No primeiro treino da Googlenet foram utilizados 5800 imagens, sendo 580 para validação, com número de iteração em 2000, levando oito dias para se concluir o treino. A rede apresentou VP = 461 , FP e FN = 119, a média harmônica do modelo ficou em 79,4%, Figura 68.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	10			10																									
2		7		9																								4	
3			11												9														
4				16														4											
5					12											8													
6						16																						4	
7							10											10											
8								16									4												
9									20																				
10										20																			
11											17												3						
12												15											5						
13													16								4								
14														20															
15	4														16														
16				2												18													
17							5										15												
18																3		17											
19	6																		14										
20																				20									
21							2														18								
22																						20							
23										4													16						
24																2								18					
25																									14			6	
26																										20			
27																			2								18		
28										4																		16	
29												5																	15

Figura 68 – Matriz de confusão 1 - Googlenet

Fonte: Autoria Própria

No segundo treino além do aumento da quantidade de imagens, o número de iteração foi alterado para 4000, levando a ser concluído em nove dias. Os valores apresentados pela rede foram: VP = 2633, FP e FN = 267. A média harmônica ficou em 90,7%. Os valores de precisão, cobertura e média harmônica também foram calculados por classe, apresentados na Figura 69 e Tabela 5.

Tabela 5 – precisão, cobertura e média harmônica em detalhes - Googlenet

	Nome Cientifico	**Nome Popular**	**Precisão**	**Cobertura**	**M. Harmônica**
1	*Persea Americana*	Abacateiro	73,0%	100,0%	84,3%
2	*Eriobotrya Japnica Lind*	Ameixa de inverno	75,0%	100%	85,7%
3	*Psidium Rufum*	Araça Roxo	100%	74,0%	85,1%
4	*Annona Montana*	Araticum	98,0%	64,0%	77,4%
5	*Annona Squamosa*	Atemoia	100%	100%	100%
6	*Cojoba Arborea*	Brinco de indio	100%	100%	100%
7	*Coffea*	Cafeeiro	72,0%	75,0%	73,4%
8	*Pera Heteranthera*	Cafezinho	98,0%	82,3%	89,4%
9	*Anacardium Occidentale*	Cajueiro	100%	100%	100%
10	*Peltophorum dubium*	Canafistula	100%	83,3%	90,9%
11	*Nectandra Megapotamica*	Canelinha	94%	100%	96,9%
12	*Cerasus*	Cerejeira	90,0%	95,7%	92,7%
13	*Prunus Serrulata*	Cerejeira Japão	91,0%	100%	95,2%
14	*Salix Babylonica*	Chorão	82,0%	100%	90,1%
15	*Lle Paraguariensis*	Erva mate	80,0%	100%	88,8%
16	*Annona Coriácea*	Fruta do conde	100%	94,3%	97,0%
17	*Psidium Guajava*	Goiabeira	94,0%	88,6%	91,2%
18	*Annona Muricata*	Graviola	100%	98,0%	99,0%
19	*Syzygium Cumini*	Jambolão	97,0%	100%	98,4%
20	*Leucaena Leucocephala*	Leucena	100%	98%	99%
21	*Citrus Limon*	Limão bergamota	76%	100%	86,3%
22	*Tibouchina Mutabilis*	Manacá da serra	100%	90,0%	94,7%
23	*Brunfelsia Uniflora*	Manacá de cheiro	90,0%	100,0%	94,7%
24	*Mangifera Indica*	Mangueira	98%	79,6%	87,8%
25	*Licania Tomentosa*	Oiti	78,0%	81,2%	79,5%
26	*Dypsis Lutescens*	Palmeira areca	100%	100%	100%
27	*Paubrasilia Echinata*	Pau Brasil	89,0%	100%	94,1%
28	*Aspidosperma Polyneuron*	Peroba rosa	82,0%	90,1%	85,8%
29	*Eugenia uniflora*	Pitanga	76,0%	77,5%	78,7%

Fonte: Autoria Própria

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	73			27																									
2		75																						25					
3			100																										
4				98														2											
5					100																								
6						100																							
7				28			72																						
8								98										2											
9									100																				
10										100																			
11											94											6							
12												90					10												
13													91															9	
14								18						82															
15			15												80							5							
16																100													
17																6	94												
18																		100											
19								3											97										
20																				100									
21							24														76								
22																						100							
23									10														90						
24																				2				98					
25																									78				22
26																										100			
27																		11									89		
28																									18			82	
29			20									4																	76

Figura 69 – Matriz de confusão 2 - Googlenet

Fonte: Autoria Própria

Os resultados da Googlenet mostraram que a pior espécie classificada apresentou valor de precisão de 72,0% e média harmônica 73,0%, no caso o Cafeeiro. As espécies Ameixa de inverno e Oiti apresentaram um valor de precisão de 75% e 78%, sendo que Ameixa de inverno apresentou valor de cobertura 100%. Analisando o valor de cobertura o Araticum (4) novamente teve um valor baixo 64,0%, o segundo menor valor de cobertura foi o Araça roxo (3) 74,0%. O Araça roxo teve 15 imagens da espécie Erva mate (15) classificadas como ela, Figura 70 mostra a semelhança entre as duas espécies.

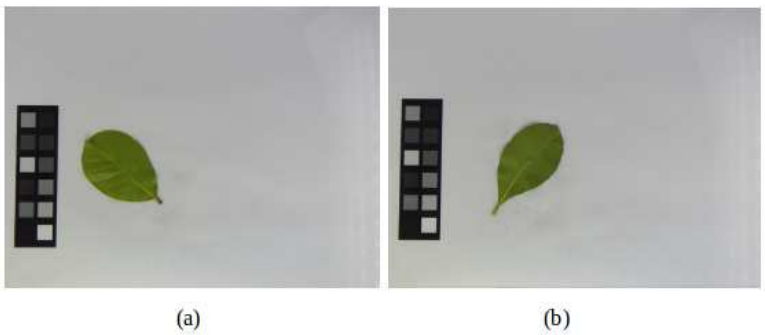

Figura 70 – (a) - Araça roxo e (b) - Erva mate

Fonte: Autoria Própria

Nesse modelo nove espécies apresentaram 100% de precisão, com média harmônica superior 85%: Araça roxo, Atemoia, Brinco de índio, Cajueiro, Canafistula, Fruta do conde, Graviola, Leucena, Palmeira areca. Na rede Googlenet três espécies apresentaram 100% de precisão, diferente da rede YOLO, Figura 71.

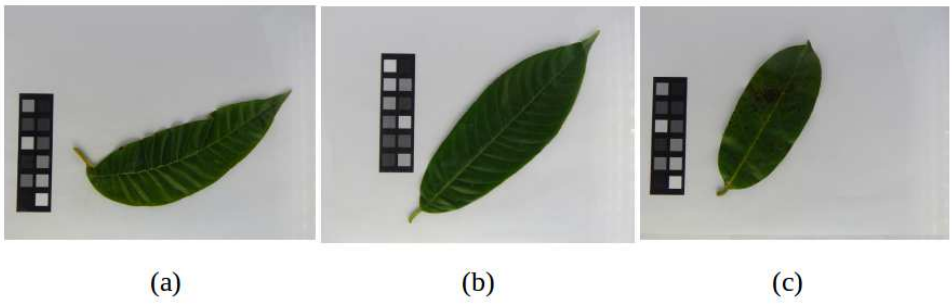

Figura 71 – (a) - Atemoia, (b) - Fruta do conde, (c) - Graviola

Fonte: Autoria Própria

Além dessas espécies que apresentaram resultado de 100%, treze espécies apresentaram valor de precisão e média harmônica superior ou igual 80% e 87%, respectivamente. Sendo elas: Araticum, Cafezinho, Canelinha, Cerejeira, Cerejeira japão, Chorão, Erva mate, Goiabeira, Jambolão, Manacá de cheiro, Mangueira, Pau Brasil e Peroba rosa, Figura 72.

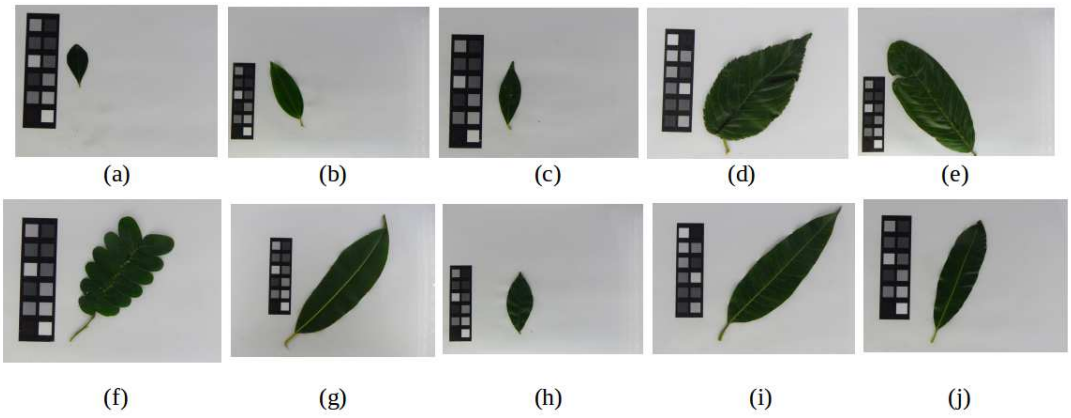

Figura 72 – (a) - Cafezinho, (b) - Canelinha, (c) - Cerejeira, (d) - Cerejeira japão, (e) - Goiabeira, (f) - Pau Brasil, (g) - Jambolão, (h) - Manacá de cheiro, (i) - Mangueira, (j) - Peroba rosa

Fonte: Autoria Própria

Um ponto importante para diferença de precisão entre as espécies é a semelhança entre as folhas. Essa semelhança ocorre em espécies que são muito parecidas como é caso da Ameixa de inverno e o Araticum, o que aumenta o número de erros. Além disso existe a semelhança de folhas especificas com outras espécies, devido a variação de formato, tamanho e cor que ocorre em uma mesma espécie. O maior exemplo dentro do modelo, é a espécie cafeeiro que devido a variação de formato, cor e tamanho da folha levaram a classifica-la a outras espécies como a fruta do conde, graviola e goiabeira. Lembrando que o aumento de número de imagens, utiliza

uma imagem de origem para gerar novas imagens, isso quer dizer se uma imagem é classificada incorretamente com aumento de dados ela irá gera outras imagens com mesmos resultados. O que explica o aumento de imagens classificadas incorretamente, após o aumento da base. Nas duas redes ficou claro que o formato das folhas, foi uma característica com melhores resultados no modelo. São elas: Araça roxo, Atemoia, Brinco de indio, Cajueiro, Canafistula, Fruta do conde, Leucena, Manacá da serra e Palmeira areca. São espécies com formato diferente entre elas e as demais.

4.0.0.3 Experimentos do YOLO

O YOLO consegue detectar o tipo do objeto e a posição dela em uma cena em imagem e vídeo. Foram feitos alguns testes no modelo treinado visando testar o seu desempenho, o primeiro teste foi passar imagens de folha seguindo o protocolo da coleta, Figura 73.

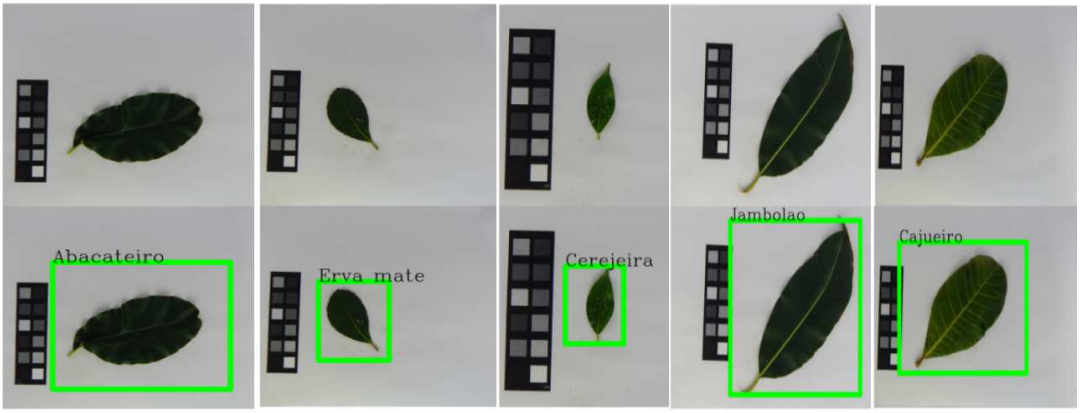

Figura 73 – Espécies reconhecidas

Fonte: Autoria Própria

Na figura é possível além de identificar o nome da folha a sua posição na cena. O segundo teste foi reconhecer mais de uma espécie na imagem e sem seguir o protocolo da coleta. No caso foi tirada a foto sem o estúdio e sem a chapa de vidro que posiciona a folha, Figura 74.

Figura 74 – Espécies reconhecidas fora do estudio

Fonte: Autoria Própria

O último teste foi em vídeo, utilizando iluminação ambiente e fora do estúdio. O modelo não obteve os mesmos resultados como foi na utilização das imagens, isso levando em conta que não foi testado no mesmo protocolo da coleta, Figura 75.

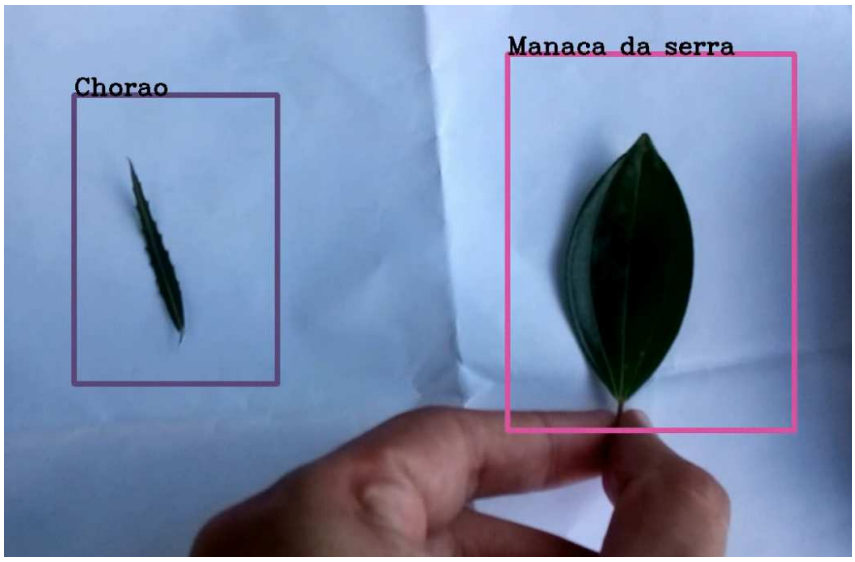

Figura 75 – Teste em video

Fonte: Autoria Própria

5 CONCLUSÕES

Ao realizar treinamentos e comparar o desempenho das redes YOLO e Googlenet, ficou claro o melhor resultado da rede do Google. A diferença entre as duas redes em precisão, cobertura e média harmônica, ficou respectivamente em 4,5, 4,8 e 4,7 pontos percentuais. Os resultados da rede da Googlenet se mostraram expressivos, mesmo com um conjunto de imagens amplo e semelhante, já que foram classificadas 29 classes com algumas espécies muito parecidas. Esse fato só torna mais viável tanto a rede Googlenet como o YOLO a serem utilizadas na classificação de outros objetos.

O YOLO apresentou um resultado abaixo da Googlenet, no modelo 86% das imagens foram classificadas corretamente. Além disso a variedade de aplicação que a rede YOLO permite ao usuário a torna interessante, já que seu uso na imagem não só classificou como também reconheceu a posição da espécie na imagem. O reconhecimento por vídeo também se mostrou um quesito interessante, mesmo que o modelo treinado não obteve um resultado tão satisfatório quanto o uso de imagem.

A falta da utilização da GPU foi um obstáculo que limitou bastante o número de treinos a serem realizados e os parâmetros das redes, mas não impediu a conclusão do trabalho. A primeira sugestão de trabalhos futuros seria a utilização de GPUs, com certeza iria melhorar o desempenho do YOLO na reprodução de videos e reconhecimento de objetos.

Analisando os resultados apresentados, com e sem o aumento de número de imagens, ficou claro a importância de uma grande base de imagens, para trabalhar com redes neurais convolucionais. A diferença nas redes YOLO e Googlenet ficou, respectivamente, 14,9 e 11,3 pontos percentuais. O algoritmo desenvolvido atingiu o objetivo, aumentando a base sem danificar os exemplares.

Durante todo o processo ficou claro as vantagens da rede neural, em um projeto anterior com mesma aplicação sem uso do deep learning, o tempo mais utilizado estava na etapa de codificação. Com o recurso das redes o tempo passou a ser maior na etapa de aquisição e alterações dos dados a serem treinados, restando ao final apenas a comparação de resultados. Mesmo assim a utilização da rede neural não é uma tarefa trivial, a escolha dos parâmetros para treinamento visando uma melhor acurácia é quase na tentativa e erro.

A sugestão para trabalhos futuros seria a utilização do modelo treinado do YOLO em

outras plataformas, como o smartphone. O android studio já permite a utilização do tensorflow associado ao modelo de treino do YOLOv2, com o uso das câmeras dos smartphones poderia criar um aplicativo que reconheceria em vídeo a espécie. Outra sugestão seria a utilização do modelo em drones, já que existe um grande número de espécies não catalogadas, explorando áreas de díficil acesso.

REFERÊNCIAS

AGARWAL, M. **Funções de Ativação**. 2017. Disponível em: <https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b>. Acesso em: 20 de abril de 2017.

ANACONDA. **The Most Popular Python Data Science Platform**. 2017. Disponível em: <https://www.anaconda.com/>. Acesso em: 8 de outubro de 2017.

BALDI, P.; SADOWSKI, P. J. Understanding dropout. In: **Neural Information Processing Systems 2013**, 2013. Disponível em: <https://papers.nips.cc/paper/4878-understanding-dropout>.

BOOK, D. L. **Capítulo 23 – Como Funciona o Dropout?** 2018. Disponível em: <http://deeplearningbook.com.br/capitulo-23-como-funciona-o-dropout/>. Acesso em: 09 de setembro de 2018.

COLLIS, J. **Glossary of Deep Learning: Batch Normalisation**. 2017. Disponível em: <https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82>. Acesso em: 20 de junho de 2017.

CUDA. **CUDA Zone**. 2018. Disponível em: <https://developer.nvidia.com/cuda-zone>. Acesso em: 11 de setembro de 2018.

DACOM. **Funções de Ativação**. 2016. Disponível em: <http://www.decom.ufop.br/redes-neurais-funcoes-de-ativacao/>. Acesso em: 20 de setembro de 2017.

DESHPANDE, A. **A Beginner's Guide To Understanding Convolutional Neural Networks**. 2015. Disponível em: <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>. Acesso em: 14 de março de 2018.

FACELI, K. et al. **Inteligência Artificial Uma Abordagem de Aprendizado de Máquina**. Rio de Janeiro: LTC, 2011. 370-378 p.

FARHADI, A.; GIRSHICK, R.; REDMON, J. You only look once: Unified, real-time object detection. In: IEEE. **University of Washington**. [S.l.], 2015.

FERRI, M. G. **Botânica: Morfologia externa das plantas: organografia**. São Paulo: Nobel, 1970. 10-32 p.

FILHO, P. L. de P. Reconhecimento de espécies florestais através de imagens macroscópica. In: **UFPR**, 2012. p. 9–48.

FOOD; NATIONS, A. O. of the united. **Global Forest Resources Assessment**. [S.l.], 2005.

GARAGE engineers. **Introduction to Image Processing**. 2017. Disponível em: <https://www.engineersgarage.com/articles/image-processing-tutorial-applications>. Acesso em: 19 de setembro de 2017.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de imagens**. São Paulo: Pearson, 2008. 454-513 p.

HAYKIN, S. **Redes Neurais Principios e prática**. São Paulo: bookman, 2001. 30-80 p.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **Pattern Recognition**, v. 70, p. 1–13, 1962. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/>.

HUI, J. **Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3**. 2018. Disponível em: <https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>. Acesso em: 08 de março de 2017.

JONES, R. P. **DarkFlow**. 2018. Disponível em: <https://medium.com/@richardpricejones/darkflow-9bdc9f9b818e>. Acesso em: 11 de maio de 2018.

JUPYTER. **What is the Jupyter Notebook?** 2017. Disponível em: <http://jupyter.org/>. Acesso em: 8 de outubro de 2017.

KATHURIA, A. What's new in yolo v3? In: . [s.n.], 2018. Disponível em: <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>.

KERAS. **Image Preprocessing**. 2016. Disponível em: <https://keras.io/preprocessing/image/>. Acesso em: 08 de setembro de 2018.

KOVACS, Z. L. **Redes Neurais Artificiais. Fundamentos e Aplicações**. São Paulo: Livraria da Fisica, 2002. 50-52 p.

LENT, R. **Cem bilhoes de Neuronios? Conceitos Fundamentais de Neurociencia**. Rio de Janeiro: Atheneu, 2010.

LOPES, S. G. B. C. **Biologia 1**. São Paulo: Saraiva, 1995. 100 p.

LORENZI, H. **Introdução à Botânica: Morfologia**. São Paulo: Instituto Plantarum, 2013. 10-52 p.

MAPS, G. **Google Maps**. 2018. Disponível em: <https://www.google.com.br/maps/@-25.2860641,-54.1167419,14z>. Acesso em: 03 de setembro de 2017.

MARCHIORI, J. N. C. **Elementos de Dendrologia**. Santa Maria: UFSM, 1995. 10-20 p.

MAYO, S. J.; REMAGNINO, P. How deep learning extracts and learns leaf features for plant classification. **Autonomous Robots**, p. 1–13, 2016.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. 432 p.

MONTES, M. C. Sobreajuste - overfitting. In: CIEMAT. **Ciemat - Centro de Investigaciones Energéticas Medioambientales y Tecnológicas**. [S.l.], 2014.

NGUYEN, N. **Inception Network Motivation**. 2018. Disponível em: <https://nhannguyen95.github.io/coursera-deep-learning-course-4-week-2/>. Acesso em: 15 de março de 2018.

NIELSEN, M. **Deep Learning**. 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/chap6.html>. Acesso em: 01 de março de 2018.

NOGARE, D. **Entendendo como funciona o algoritmo de Cluster K-Means**. 2017. Disponível em: <http://www.diegonogare.net/2015/08/entendendo-como-funciona-o-algoritmo-de-cluster-k-means/>. Acesso em: 20 de março de 2017.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial. 2a ed**. Rio de Janeiro: Campus, 2004. 40-50 p.

NUMPY. **PyScience-Brasil**. 2017. Disponível em: <http://pyscience-brasil.com/numpy>. Acesso em: 8 de outubro de 2017.

NUMPYDL. **Convolution Neural Networks**. 2015. Disponível em: <http://numpydl.readthedocs.io/en/0.2.0/tutorials/CNN_Part1/?highlight=re>. Acesso em: 04 de março de 2018.

ODEMIR, B. et al. Leaf epidermis images for robust identification of plants. In: **Instituto de Física de São Carlos**. [S.l.: s.n.], 2017. p. 2–10.

PACHECO, A. **Redes Neurais Convolutivas – CNN**. 2017. Disponível em: <http://www.computacaointeligente.com.br/artigos/redes-neurais-convolutivas-cnn/>. Acesso em: 11 de agosto de 2017.

PEARLSTEIN, L.; KIM, M.; SETO, M. Convolutional neural network application to plant detection, based on synthetic imagery. In: **Proceedings - Applied Imagery Pattern Recognition Workshop**. [s.n.], 2017. Disponível em: <www.scopus.com>.

PEDRINI, K.; SCHWARTZ, W. R. **Análise de Imagens Digitais**. São Paulo: Thomson, 2008. 151-199 p.

PINHEIRO, A. L. **Fundamentos de taxonomia e dendrologia tropical**. Santa Maria: SIF, 2000. 70-80 p.

PIRES, W. O. Reconhecimento de espécies florestais utilizando técnicas de processamento de imagem. In: **7º Seminario de extenção e inovação, Londrina**. [S.l.: s.n.], 2017.

PYSCIENCE-BRASIL. **Python: O que é? Por que usar?** 2017. Disponível em: <http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>. Acesso em: 8 de outubro de 2017.

REDMON, J. C. **YOLO: Real-Time Object Detection**. 2010. Disponível em: <https://pjreddie.com/darknet/yolo/>. Acesso em: 20 de setembro de 2017.

RICH, E.; KNIGHT, K.; NAIR, S. B. **Artificial Intelligence**. Canada: McGraw-Hill, 1983. 70-79 p.

SANTOS, L. A. **GoogleNet Artificial Inteligence**. 2017. Disponível em: <https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html>. Acesso em: 20 de setembro de 2017.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas. Fundamentos Teóricos e Aspectos Práticos**. São Paulo: Artliber, 2015. 150-170 p.

SZEGEDY, C.; LIU, W.; JIA, Y. Going deeper with convolutions. In: IEEE. **IEEE/ Boston, MA, USA**. [S.l.], 2015.

TENSORFLOW. **Big Data + Deep Learning = Google TensorFlow**. 2017. Disponível em: <http://www.cienciaedados.com/big-data-deep-learning-google-tensorflow/>. Acesso em: 8 de outubro de 2017.

UBUNTU. **Ubuntu: The leading operating system for PCs, IoT devices, servers ...** 2017. Disponível em: <https://www.ubuntu.com/>. Acesso em: 8 de outubro de 2017.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: IEEE. **IEEE/ Conferencia Universidade Federal de Fluminence de Niterói**. [S.l.], 2015.

WITTEN, I. H.; FRANK, E. **Data Mining Practical Machine Learning Tools and Techniques Third Edition**. San Francisco: ELSEVIER, 2011. 1-40 p.

WURTZ, R. H. Recounting the impact of hubel and wiesel. **Pattern Recognition**, v. 32, p. 1–20, 2009. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2718241/>.

YALCIN, H.; RAZAVI, S. Plant classification using convolutional neural networks. In: **2016 5th International Conference on Agro-Geoinformatics, Agro-Geoinformatics 2016**. [s.n.], 2016. Disponível em: <www.scopus.com>.