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

VALÉRIA NUNES DOS SANTOS

RECONHECIMENTO DE OBJETOS EM UMA CENA UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2018

VALÉRIA NUNES DOS SANTOS

RECONHECIMENTO DE OBJETOS EM UMA CENA 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

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

RECONHECIMENTO DE OBJETOS EM UMA CENA UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Por

VALÉRIA NUNES DOS SANTOS

Este Trabalho de Conclusão de Curso foi apresentado às 07:45h do dia 13 de junho 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

SANTOS, Valéria Nunes. RECONHECIMENTO DE OBJETOS EM UMA CENA UTILIZANDO REDES NEURAIS CONVOLUCIONAIS. 65 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2018.

Identificar e classificar objetos em uma imagem é um processo que tem chamado a atenção nos últimos anos devido a evolução das redes neurais artificias. Hoje existem redes neurais artificiais capazes de serem treinadas com *N* classes e ainda manter uma acurácia alta para predição de objetos. As redes neurais convolucionais são as principais quando se trata de imagens, pois foram projetadas com foco em localizar objetos em cenas. Com isso, tanto a classificação quanto a localização de latas e garrafas nos quais são os objetos propostos se torna possível através das redes neurais convolucionais. É importante frisar que a base do treinamento de uma rede neural convolucional se resume em cálculo de larga escala o que por sua vez requer muito processamento, logo o uso de ferramentas que tragam benefícios em relação ao custo de processamento como é o caso das GPUs e da tecnologia CUDA foi um critério indiscutível, pois sem estes não seria possível utilizar as configurações ideais da rede e também impactaria negativamente no tempo de treinamento da mesma. Por fim, os resultados obtidos com o treinamento da rede neural convolucional para detectar latas e garrafas atingiu as expectativas, considerando que a rede alcançou uma média de precisão de 87,55%, em um intervalo de tempo de aproximadamente quatro dias de treinamento.

Palavras-chave: processamento de imagens, rede neural artificial, convolução, CNN

ABSTRACT

SANTOS, Valéria Nunes. RECOGNITION OF OBJECTS IN A SCENE USING CONVOLUTIONAL NEURAL NETWORK. 65 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2018.

Identifying and classifying objects in an image is a process that has attracted attention in recent years due to the evolution of artificial neural networks. Today there are artificial neural networks capable of being trained with N classes and still maintain a high accuracy for predicting objects. The convolutional neural networks are the main ones when it comes to images because they are designed with a focus on finding objects in scenes. Thus, both the classification and location of cans and bottles in which the proposed objects are made possible through convolutional neural networks. It is important to stress that the training base of a convolutional neural network is summarized in large-scale computation, which in turn requires a lot of processing, so the use of tools that bring benefits in relation to the cost of processing such as GPUs and of CUDA technology was an indisputable criterion, since without them it would not be possible to use the ideal network configurations and also negatively impact the training time of the network. Finally, the results obtained with convolutional neural network training to detect cans and bottles reached expectations, considering that the network reached an average accuracy of 87.55%, in a time interval of approximately four days of training.

Keywords: image processing, artificial neural network, convolution, CNN

AGRADECIMENTOS

Agradeço a minha querida Bisavó Maria Sustene que sempre me abençoou e me ajudou todas as semanas que nos despedíamos. Ao meu pai Paulo Nunes que foi quem mais me incentivou e se mostrou orgulhoso por eu estar em uma Universidade. A minha mãe Terezinha Skumra que sempre fez o possível para me ajudar durante os dias que estive longe de casa. Ao meu irmão Matheus Nunes pelas vezes que me distraia enquanto eu estudava mas ao mesmo tempo proporcionava pensamentos mais simples e lógicos, e que me acudia das vezes que esquecia algo em casa. Pelo meu namorado Julio Cesar que me fazia pensar varias vezes sobre o mesmo assunto para ter certeza sobre o que eu fazia. Para minhas amigas Juliane Dias, Márcia Ribas, Juliane Wollmann, e meu amigo Lucas Hofer que sempre se mostraram orgulhosos por eu ainda estar estudando, o que me fazia sentir anseio de continuar. Para meu amigo Carlos que sempre me ajudou com seus conhecimentos em resolver pepinos em trabalhos e sempre foi meu parceirão nos trabalhos em dupla, e juntamente com Marcos Thiago, Felipe Meller, Gabriela Michellon, Marcela Marques, e Alex Makayama que proporcionaram horas de conversas/momentos/experiências agradáveis durante o tempo em que permaneci na Universidade, e ao Crow, Douglas, Gustavo, Adriano, Lucas, Agnaldo, e Alex que proporcionaram conversas/momentos/experiências de uma boa amizade fora da Universidade.

Agradeço também ao meu orientador Pedro Luiz de Paula Filho, que além de me confiar um trabalho tão importante, foi o professor que mais me motivou e incentivou durante o curso, e também ao meu Co-orientador Arnaldo Candido Junior que me socorreu várias vezes com dúvidas sobre o conteúdo e sempre sendo muito paciente. Um agradecimento especial também aos demais professores nos quais compartilharam seu conhecimento não só de conteúdo mas também de vida, que levarei comigo em minha jornada.

Serei sempre mais feliz que ontem e menos que amanhã.

LISTA DE FIGURAS

LISTA DE SIGLAS

AI	Inteligência Artificial do inglês Artificial Intelligence
AP	Média de Precisão do inglês Avarage Precision
CNN	Redes Neurais Convolucionais do inglês Convolutional Neural Network
CPU	Unidade Central de Processamento do inglês Central Processing Unit
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
FPS	Quadros por segundo do inglês Frames Per Second
GPU	Unidade de Processamento Gráfico do inglês Graphics Processing Unit
IoU	Intersecção sobre a União do inglês Intersection over Union
LRF	Campo Receptivo Local do inglês Local Receptive Field
mAP	Média da Média de Precisão do inglês Mean Avarage Precision
MLP	Multilayer Perceptron
PDI	Processamento Digital de Imagens
RNA	Rede Neural Artificial do inglês Artificial Neural Network
RPN	Rede de proposta de região do inglês Region Proposal Network
SSD	Single Shot MultiBox Detector
TDNN	Time-Delay Neural Network
YOLO	You Only Look Once

SUMÁRIO

1 INTRODUÇÃO

Dos anos 60 até os dias atuais, a área de processamento digital de imagens (PDI) cresceu rapidamente. Explorando áreas como medicina, programas espaciais, industrial, ciência biológica, arqueologia, entre outras. PDI já é capaz de propor soluções como restauração e realce de imagens de objetos já degradados, auxiliar no reconhecimento de padrões em imagens de satélite, realçar contraste e codificar níveis de intensidade de cores em imagens radiográficas, melhorar imagens adquiridas de experimentos em áreas como microscopia eletrônica, entre inúmeras outras (GONZALEZ; WOODS, 2008).

Já em áreas que visam identificação de objetos em uma cena, PDI influência tecnologias como identificação automática de caracteres escritos em diversos idiomas a partir de documentos, identificação e rastreamento de alvos em imagens de satélite na área militar, navegação de veículos autônomos, detecção de obstáculos no trajeto percorrido por robôs, contagem e identificação de células sanguíneas em lâminas de microscopia, entre outras (PEDRINI; SCHWARTZ, 2008).

Já as Redes Neurais Artificiais (RNA), surgiu pela primeira vez no ano de 1943 em um trabalho realizado por *Warren S. McCulloch* e *Walter H. Pitts* que visavam desenvolver um modelo matemático baseado no comportamento do neurônio biológico. Após alguns anos, surgiram outros autores que contribuíram na evolução dessa ideia. Em 1949 com *Donald O. Hebb* aplicando regras de aprendizado que são descritas nas redes como atualização de pesos sinápticos. Em 1958 com *Frank Rosenblatt* surge então o primeiro modelo de rede neural implementado chamado de *perceptron* (BEALE; JACKSON, 1990).

Contudo, em 1969 *Marvin Minsky* e *Seymour Papert* apontaram uma falha no uso do *perceptron*, descrevendo-a no livro *Perceptrons* o que suspendeu as pesquisas com RNAs em função da desmotivação e desencorajamento. Porem, cerca de 14 anos após as pesquisas com RNAs serem abandonadas, *John Hopfield* traz o assunto de RNAs novamente ao interesse nos campos de pesquisa com seus artigos *Neural networks and physical systems with emergent collective computational abilities* escrito em 1982, e *Neurons with graded response have collective computational properties like those of two-state neurons* escrito em 1984 (YADAV et al., 2015).

Segundo Pedrini e Schwartz (2008) as RNAs de hoje têm baixa dependência a domínios específicos, isso significa que a mesma rede pode ser utilizada em problemas distintos. Os pontos que mais despertam interesse no uso das RNAs é sua capacidade de aprender a partir de exemplos, ou seja, sem necessidade de interferência externa, e também a habilidade de generalizar a informação aprendida. Com o potencial que uma RNA dispõe, inúmeras pesquisas visam aplica-la para dar soluções a problemas de ações repetitivas e também a problemas de classificação e/ou reconhecimento de padrões em *big data*.

Como o objetivo principal desse trabalho é reconhecer objetos em imagens utilizando redes neurais artificiais, o modelo de rede neural artificial fundamental é conhecido como rede neural convolucional (CNN), na qual é uma combinação entre PDI e RNA. O objetivo de uma rede CNN é aprender sobre objetos para que posteriormente possa classifica-los em cenas diversas representadas através de fotos, vídeos ou até mesmo em tempo real. O processo de aprendizagem se resume em treinar a rede com imagens que contenham os objetos que ela precisa reconhecer. Essas imagens devem conter o objeto em diferentes situações, tais como ângulo, escala, oclusão, foco, ambiente, etc. Com isso a própria rede extrai características que julga necessária para o reconhecimento deste objeto. Conforme a rede treina, as características dos objetos vão se tornando mais evidentes para ela. Até chegar em um ponto onde a rede já sabe quais as características ela deve buscar na imagem quando estiver procurando por este objeto.

1.1 OBJETIVOS GERAIS E ESPECÍFICOS

Esse trabalho tem como objetivo treinar uma rede neural convolucional para que a mesma seja capaz de localizar latas e garrafas em imagens. Para facilitar o entendimento desse processo é necessário abordar os temas de Convolução em processamento de imagens, e *Deep Learning* em redes neurais artificiais. Buscando atingir o objetivo principal, os objetivos requeridos são:

- Buscar ferramentas fundamentais para treinamento de uma rede CNN;
- Buscar ferramentas de auxílio para treinamento de uma rede CNN;
- Montar a base de imagens que contenham latas e garrafas para a fase de treinamento, e também a base de imagens que não necessariamente contenha latas e/ou garrafas para a fase de classificação;

- Buscar um modelo de rede neural convolucional que atenda ao objetivo de trabalhar com somente duas classes;
- Configurar o modelo aderido para atender ao objetivo;
- Iniciar a fase de treinamento e juntamente a de classificação para entender as deficiências do aprendizado com o objetivo de melhora-lo;

1.2 ORGANIZAÇÃO DO DOCUMENTO

Esse documento está organizado da seguinte maneira. O Capítulo 2 aborda os conceitos relacionados a imagem que servem de base para conceituar tanto o processamento digital de imagens como os conceitos mais primitivos de uma rede neural convolucional. Já o Capítulo 3 descreve uma breve introdução sobre Redes Neurais Artificiais, *Deep Learning*, e Redes Neurais Convolucionais, abordando os principais conceitos que relacionam as 3 áreas tendo como foco principal as CNNs. Os materiais e métodos necessários para o desenvolvimento estão expostos no Capitulo 4. E os Resultados e Discussões no Capitulo 5, por fim a Conclusão é o Capitulo 6.

2 PROCESSAMENTO DIGITAL DE IMAGENS

O termo processamento digital de imagens, está presente na computação desde 1920 onde sua primeira aplicação surgiu da necessidade de melhoramento de imagens digitalizadas para jornais enviadas por meio de cabo submarino de Londres para Nova York (GONZALEZ; WOODS, 2008). Hoje, PDI se aplica a inúmeras áreas de conhecimento tais como Astronomia, Biologia, Matemática, Medicina, Física, etc. Alguns exemplos práticos de aplicação de PDI são: monitoramento de trafego (SITARAM et al., 2015), automação industrial (BENALIA et al., 2016), biometria facial (AKINLOLU, 2016), aplicações militares (PATIL; PETE, 2015), medicina (BIDHULT et al., 2016), entre outros.

Primeiramente é necessário entender o que é uma imagem digital. Uma imagem é uma função bidimensional $f(x,y)$, em que x e y são coordenadas *espaciais* (plano), e a amplitude de f no ponto (x,y) é chamada de *intensidade* ou *nível de cinza* da imagem nesse ponto. Quando essa função existe em uma quantidade finita e discreta, caracteriza-se então como uma *imagem digital* (GONZALEZ; WOODS, 2008).

Logo, é valido afirmar que uma imagem digital é composta por um número finito de elementos, e cada elemento possui sua localização e valor especifico. Esses elementos são conhecidos na literatura de processamento de imagens como Pixel (MIRANDA, 2006). As informações que o pixel armazena podem variar de acordo com o tipo de imagem que está sendo processada. Como o foco desse trabalho está nas imagens que possuem propriedades baseadas na cor e intensidade da luz, não convém entrar em detalhes dos demais tipos de imagem.

Os pixeis possuem alguns relacionamentos importantes entre eles, logo é essencial entender esse relacionamento para compreender uma imagem digital. Contudo, as próximas seções abordam os relacionamentos básicos entre os pixeis.

2.1 VIZINHANÇA

A vizinhança entre pixeis, é dividida em três partes, *horizontais e verticais*, *diagonais*, e a *união* das duas. Dado um pixel p na coordenada (x,y) seus quatro vizinhos horizontais e verticais se encontram nas coordenadas $(x+1,y),(x-1,y),(x,y+1),(x,y-1)$ e expresso por $N_4(p)$. Já os vizinhos *diagonais* são outros quatro pixeis encontrados nas coordenadas $(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1)$ e são expressos por $N_D(p)$. Por fim, a união dessas partes $N_8(p) = N_4(p) \cup N_D(p)$ formam a *vizinhança-8* de p, expresso por $N_8(p)$. Cada pixel é uma unidade de distância de (x,y), logo alguns vizinhos de $N_4(p)$,$N_D(p)$ e $N_8(p)$ ficarão para fora da imagem se (x,y) estiver na borda da imagem (GONZALEZ; WOODS, 2008; PEDRINI; SCHWARTZ, 2008).

2.2 ADJACÊNCIA

Pedrini e Schwartz (2008) afirmam que a adjacência entre dois pixeis ocorre quando esses pixeis são conexos de acordo com a vizinhança escolhida, ou seja, um pixel p é adjacente ao pixel q se algum elemento de q estiver no conjunto da vizinhança de p. Porém, segundo Gonzalez e Woods (2008) o uso da *adjacência-8* ($q \in conjunto\ N_8(p)$), gera muita ambiguidade, e para remover essas ambiguidades existe a *adjacência-m*.

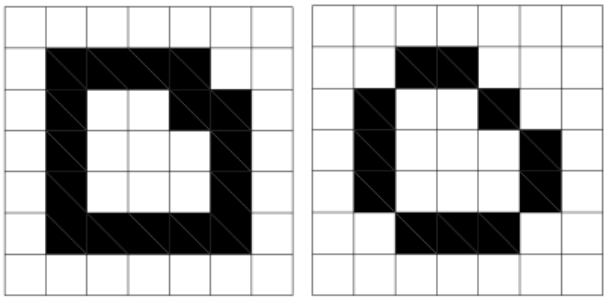
A *adjacência-m* (adjacência mista) ocorre quando dois pixeis p e q com valores pertencendo a V adequam-se em uma das seguintes regras:

1. q estiver em $N_4(p)$, ou
2. q estiver em $N_D(p)$ e o conjunto $N_4(p) \cap N_4(q)$ não contiver nenhum pixel cujos valores pertençam a V.

Onde V é o simbolo para representar o conjunto de valores de intensidade que define a adjacência. O termo ambiguidade mencionado anteriormente, indica a possibilidade de haver mais de uma conexão entre dois pixeis.

2.3 CAMINHO

Um *caminho* na imagem entre os pixeis p e q onde as coordenadas são (x_0, y_0) e (x_n, y_n) respectivamente, é uma sequência de pixeis distintos com coordenadas $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$, onde n é o comprimento do caminho, e (x_i, y_i) é adjacente a $(x_{i+1},\ y_{i+1})$, assim como $(x_{i+1},\ y_{i+1})$ é adjacente a $(x_{i+2},\ y_{i+2})$, assim sucessivamente, tal que i = 1,2,...,n-1. Ao considerar a relação de vizinhança-4 então existe um caminho-4, assim como para vizinhança-8 existe um caminho-8 (PEDRINI; SCHWARTZ, 2008). A Figura 1 representa um exemplo desses caminhos, e permite perceber que o caminho-4 possui comprimento 16, e o caminho-8 possui comprimento 11.



(a) (b)

Figura 1 – Exemplos de caminhos. (a) caminho-4; (b) caminho-8. Onde cada subquadrado representa um pixel, porém os quadrados pretos representam um caminho e a adjacência aplicada sobre eles, e os brancos completam o restante da imagem.

Fonte: Adaptado de (PEDRINI; SCHWARTZ, 2008).

2.4 CONECTIVIDADE

A *conectividade* entre pixeis depende de dois critérios, a vizinhança e o valor de similaridade (intensidade de cinza, cor, ou textura). Por exemplo, em uma imagem binária, onde os pixeis podem assumir os valores 0 ou 1, se o critério de vizinhança entre os dois pixeis for *vizinhança* $-$ 4, esses pixeis poderão ser considerados conexos somente se possuírem o mesmo valor. Logo, a conectividade é um conceito que auxilia a estabelecer limites de objetos

e componentes de regiões em uma imagem (PEDRINI; SCHWARTZ, 2008).

2.5 REGIÃO

Uma *região* é um subconjunto de pixeis conexos em uma imagem. A união de duas regiões podem ser consideradas adjacentes se esta formar um conjunto conexo, porém se não forem adjacentes, então serão $disjuntas$. As regiões consideram apenas as adjacências-4 e -8.(GONZALEZ; WOODS, 2008).

Para exemplificar, suponha que R represente uma região, e K seja o número de regiões disjuntas, R_k, $k = 1, 2, \ldots, K$, nenhuma das quais toca a borda da imagem. Expressando por R_u a união de todas as regiões K e por $(R_u)^c$ seu complemento. Todos os pontos em R_u são chamados de $frente(foreground)$ da imagem, e todos os pontos em $(R_u)^c$ são chamados de $fundo(background)$ da imagem (GONZALEZ; WOODS, 2008).

2.6 FRONTEIRA

A fronteira também é conhecida como borda ou contorno. A $fronteira$ de uma região R, é o conjunto de pontos adjacentes aos pontos do complemento de R. Em outras palavras, a fronteira são os pontos que não fazem parte da região.

Existe também uma distinção importante entre $contorno\ interno$ e $contorno\ externo$, devido a existência de algoritmos seguidores de contorno ($border\ following$), que geralmente são formulados para seguir o contorno externo da região na tentativa de formar um caminho fechado. Como exemplo, se considerar o contorno interno da região de valor 1 ilustrado na Figura 2, o contorno será a própria região suprimindo assim o conceito de caminho fechado, por outro lado se considerar o contorno externo da mesma região, têm-se o caminho fechado ao redor dessa região.

Apesar do conceito de *borda* e $fronteira$ serem dados como sendo o mesmo, segundo Gonzalez e Woods (2008), existe uma diferença importante entre esses conceitos, pois a $fronteira$ de uma região forma um caminho fechado e assim, é um conceito “global”, já as

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Figura 2 – Exemplificação da diferença entre contorno externo e interno.
Fonte: (GONZALEZ; WOODS, 2008)

bordas são formadas por pixeis com valores cujas derivadas excedem um limiar pré-definido, dessa maneira, se torna um conceito "local" baseado em uma medida de descontinuidade de nível de intensidade em um ponto.

2.7 MEDIDAS DE DISTÂNCIA

Segundo Gonzalez e Woods (2008), Pedrini e Schwartz (2008) é importante saber que, apesar de muitas aplicações demandarem o cálculo da distância entre dois componentes de uma imagem, não há uma forma única para se definir distância em imagens digitais. Primeiramente, qualquer métrica de distância D deve satisfazer as seguintes propriedades:

Para os pixeis p, q e z, com coordenadas (x_0, y_0), (x_1, y_1), e (x_2, y_2), respectivamente, D é uma *função distância* ou *medida de distância* se:

- $D(p,q) \geq 0 \qquad (D(p,q) = 0 \; se \; p = q)$,
- $D(p,q) = D(q,p)$ e
- $D(p,z) \leq D(p,q) + D(q,z)$.

Algumas medidas frequentemente usadas em análise de imagens digitais, que satisfazem as propriedades acima são:

1. *Distância Euclidiana*. A D_e entre p e q é definida como:

$$D_e(p,q) = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \tag{1}$$

Para a medida D_e, os pixeis com uma distância menor ou igual a algum valor r formam um disco de raio r centrado em p;

2. *Distância D_4* ou *Distância city block*. A D_4 entre p e q é definida como:

$$D_4(p,q) = |x_0 - x_1| + |y_0 - y_1| \tag{2}$$

Nesse caso, os pixeis com distância D_4 de p menor ou igual a um valor r formam um losango centrado em p;

3. *Distância D_8* ou *Distância Chessboard* ou ainda *Distância Suprema*. A D_8 entre p e q é definida como:

$$D_8(p,q) = \max(|x_0 - x_1|, |y_0 - y_1|) \tag{3}$$

Nesse caso, os pixeis com distância D_8 DE p menor ou igual a um valor r formam um quadrado centrado em p;

As distâncias D_4 e D_8 entre p e q envolvem apenas coordenadas dos pontos, portanto são independentes de quaisquer caminhos que possam existir entre os pontos. Porém, ao optar por considerar a adjacência-*m*, a distância D_m entre dois pontos é definida como o caminho-*m* mais curto entre os pontos, assim, a distância entre dois pixeis dependerá não só dos valores dos pixeis ao longo do caminho, mas também dos valores dos pixeis vizinhos (GONZALEZ; WOODS, 2008).

Para exemplificar pode ser observado a Figura 3, onde os pontos com bordas escuras recebem adjacência de valor 1, e os demais 0. Na Figura 3(a) a extensão do caminho-m mais curto entre P_0 e P_4 é igual a 2. Porém na Figura 3(b) a extensão do caminho-m mais curto passa a ser 3, assim como na Figura 3(c). E por último, na Figura 3(d) a extensão do caminho mais curto, passa a ser 4. É possível notar que essa variação da distância mais curta entre os mesmos pontos, se dá pela modificação do valor da adjacência dos pixeis vizinhos.

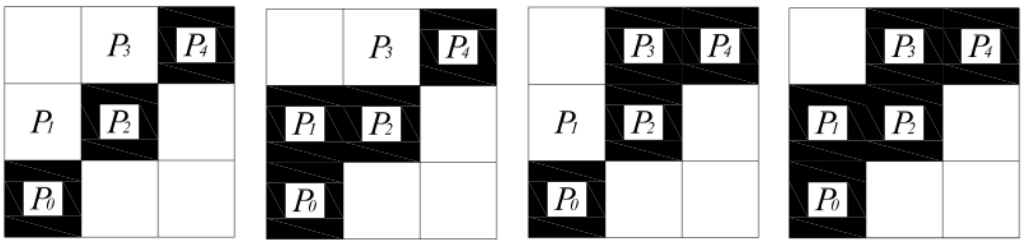


Figura 3 – Exemplificação de interferência da adjacência em medidas da distância.
Fonte: Adaptado de (GONZALEZ; WOODS, 2008)

2.8 ÁREAS DE APLICAÇÃO

Segundo Gonzalez e Woods (2008) o processamento de imagens decorre de duas áreas principais de aplicação, sendo a primeira a melhoria de informação visual para a interpretação humana, e a segunda o processamento de dados em imagens, para percepção automática através de máquinas.

Como até então não existe um acordo geral entre os autores em relação ao ponto onde termina o processamento de imagens e onde começa a visão computacional, tem-se um paradigma útil que auxilia na classificação de processos computacionais, chamados de processos de nível baixo, médio e alto.

O processo de nível baixo, é caracterizado pelo fato da saída ser a mesma imagem de entrada, porém com alguns ajustes como redução de ruido, realce de contraste, aguçamento de imagens. O processo de nível médio, é caracterizado pelo fato da saída ser um atributo extraído da imagem de entrada, esse processo envolve tarefas como a segmentação. Já o processo de nível alto envolve reconhecer o objeto, ou o conjunto de objetos, "dar sentido"da mesma maneira que ocorre com as funções cognitivas associadas a visão (GONZALEZ; WOODS, 2008).

Conforme a definição do processo de nível médio, fica evidente que o caminho que deve ser seguido nesse trabalho, é baseando em segmentação de imagens, por isso, a próxima seção abordará o tema.

2.9 SEGMENTAÇÃO DE IMAGENS

Segundo as definições de Solomon e Breckon (2013) "O objetivo básico da segmentação é, portanto, dividir a imagem em regiões mutuamente exclusivas as quais possamos associar rótulos que façam sentido", ou seja, ter como entrada uma imagem mas como saída ter elementos extraídos dessa imagem. Entendendo o conceito de segmentação, o foco em reconhecimento de objetos em uma cena tem favorecido o surgimento de aplicações práticas tais como: Detecção automática de rachaduras em pontes (ABDEL-QADER et al., 2003), patologia digital (MADABHUSHI; LEE, 2016), segmentação em imagens médicas de

ultra-som (GUPTA; ANAND, 2017), analise de qualidade da carne baseado na coloração (ADI et al., 2015), entre outros.

Entretanto, a segmentação de imagens é uma das tarefas mais difíceis no processamento de imagens, a precisão da segmentação determina o sucesso ou o fracasso final dos procedimentos de análise computadorizada, logo, algoritmos de segmentação fracos ou inconsistentes acarretarão falhas no processamento (GONZALEZ; WOODS, 2008; SOLOMON; BRECKON, 2013). Isso porque, não há sentido nenhum em se aplicar a segmentação, se o nível de detalhes do objeto extraído, não corresponder fielmente as características do mesmo para que possa ser interpretado.

Existem duas abordagens convencionais na aplicação de segmentação, são elas a *descontinuidade* e a *similaridade*. A descontinuidade se baseia em diferença brusca do tom em grupos de pixeis, que eventualmente indicam bordas. Já a similaridade se baseia em agrupar pixeis em uma dada região com base nas características similares entre eles (GONZALEZ; WOODS, 2008; SOLOMON; BRECKON, 2013; PEDRINI; SCHWARTZ, 2008; PITERI; RODRIGUES, 2011).

As próximas seções abordarão as principais técnicas de segmentação baseadas nos valores de intensidade dos pixeis das imagens, sendo elas a detecção de descontinuidade, limiarização, e segmentação de regiões.

2.9.1 Detecção de Descontinuidades

Segundo Gonzalez e Woods (2008) em relação a detecção de descontinuidade pode se afirmar que:

1. As bordas mais grossas em uma imagem geralmente são produzidas por derivadas de primeira ordem;
2. Já os detalhes mais finos, como linhas finas, pontos isolados e ruídos são produzidos de modo mais eficaz por derivadas de segunda ordem;
3. As derivadas de segunda ordem produzem uma resposta de borda dupla nas transições de rampa e de degrau de intensidade;
4. O sinal da segunda derivada auxilia para determinar se uma transição em uma borda vai do claro para escuro ou vice-versa.

As literaturas de Gonzalez e Woods (2008) e Pedrini e Schwartz (2008) subdividem

a descontinuidade encontrada nas imagens digitais em três tipos básicos, sendo eles pontos, segmentos de retas, e bordas. A utilização de uma máscara é um método comum de identificação de descontinuidade, esse processo é conhecido como convolução.

Para exemplificar os cálculos necessários, supondo uma máscara w com tamanho 3x3 pixeis, ilustrada na Figura 4. O cálculo do ponto central da região onde a máscara está posicionada, consiste em calcular a soma dos produtos dos coeficientes dessa máscara pelos níveis de cinza da região compreendida pela mesma, representada pela Equação 4:

$$R = w_1 z_1 + w_2 z_2 + ... + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{4}$$

Na qual z_i é o nível de cinza associado com o coeficiente w_i da máscara. R é a resposta da máscara posicionada sobre um ponto da imagem. Nos casos em que a máscara é posicionada em pixeis da borda, a resposta é calculada utilizando a vizinhança parcial apropriada, conforme os métodos comentados na Seção 2.10.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 4 – Máscara com dimensão de 3x3 pixeis.
Fonte: (GONZALEZ; WOODS, 2008)

2.9.1.1 Detecção de Pontos Isolados

Até então, é sabido que a busca por pontos isolados se torna mais eficaz com a utilização da derivada de segunda ordem, mas também é necessário tornar sutil a imagem (GONZALEZ; WOODS, 2008). Essa combinação é conhecida como Laplaciano, e é expressa na Equação 5.

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{5}$$

Essa expressão pode ser implementada usando a máscara ilustrada na Figura 5, e pode-se

afirmar que um ponto foi detectado se $|R|$ nesse ponto exceder um limiar T (não negativo) estabelecido. Considerando que os pontos de saída sejam rotulados como 1, e os demais 0. A saída é obtida baseada na Função 6.

$$g(x,y) = \{_{0} \quad 1 \; se \; |R(x,y)| \geq T, \quad 0 \; caso \; contrário \tag{6}$$

Na qual g é a imagem de saída e $|R|$ é dado pela Equação 4. Intuitivamente, um ponto isolado tem uma intensidade muito diferente do seu entorno, logo, pode ser facilmente detectável por esse tipo de máscara.

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

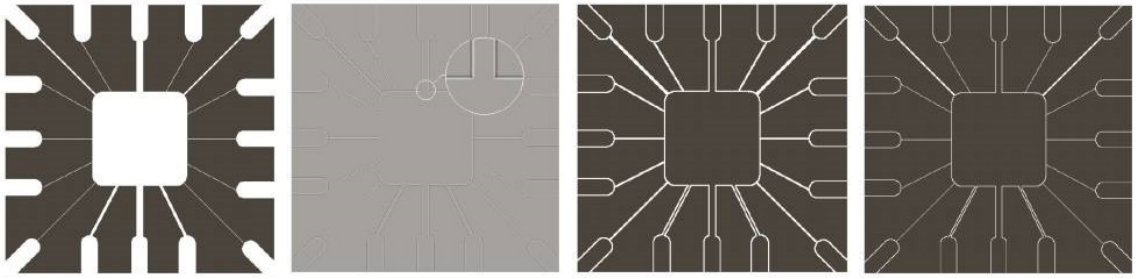
Figura 5 – Máscara com dimensão de 3x3 pixeis.
Fonte: (GONZALEZ; WOODS, 2008)

2.9.1.2 Detecção de Retas

Para detecção de retas em uma imagem, também se considera a derivada de segunda ordem assim como a máscara Laplaciano, porém, o efeito de linha dupla gerado pela segunda derivada deve ser tratado. Dado a imagem ilustrada na Figura 6 , deve ser considerado que:

A Figura 6(a) ilustra uma imagem binária de uma conexão $wire-bond$ de um circuito eletrônico.

A Figura 6(b) ilustra a mesma imagem submetida a um filtro Laplaciano. De acordo com Gonzalez e Woods (2008) uma imagem Laplaciana contém valores negativos porque no processo de convolução entre imagem e máscara cuja soma dos coeficientes é zero, os pixeis do resultado somarão zero, o que sugere a existência de pixeis positivos e negativos no resultado. Essa afirmação pode ser observada na área ampliada da Figura em questão, onde o cinza médio representa zero, os tons mais escuros de cinza representam valores negativos, e os mais claros os valores positivos, criando assim um efeito de linha dupla.



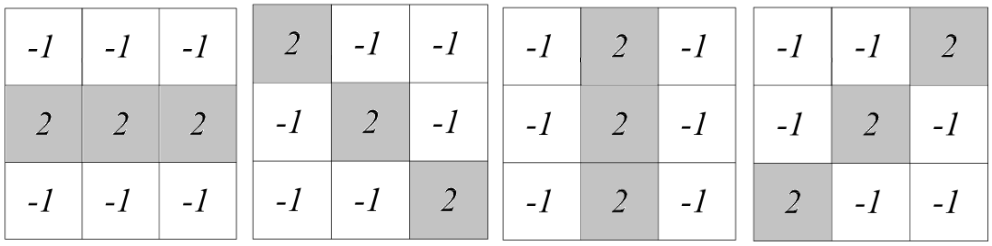
(a) (b) (c) (d)

Figura 6 – Exemplificação da aplicação do filtro Laplaciano.
Fonte: (GONZALEZ; WOODS, 2008)

A Figura 6(c) ilustra o que ocorre quando os valores negativos são tratados como um valor absoluto do filtro Laplaciano, porém não é a maneira mais adequada de se fazer, pois as linhas mais finas são consideravelmente mais uteis.

A Figura 6(d) ilustra a utilização dos valores positivos do filtro Laplaciano, tornando as linhas mais finas.

É fato que segmentos de retas podem ser detectados pelo uso de máscaras e em muitos casos, o interesse está na detecção de linhas em direções especificas. A Figura 7 ilustra as máscaras que auxiliam nessa detecção.



(a) Horizontal (b) +45 (c) Vertical (d) -45

Figura 7 – Máscaras de detecção de linhas em imagens.
Fonte: (GONZALEZ; WOODS, 2008)

2.9.1.3 Detecção de Bordas/Junções

Segundo Gonzalez e Woods (2008) existem três modelos de bordas que são

considerados na elaboração de algoritmos para detecção de bordas, sendo eles representados pela Figura 8.

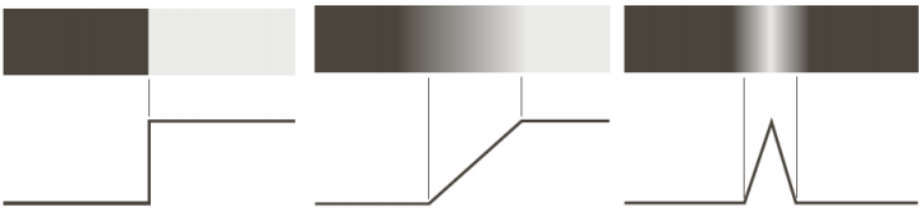


Figura 8 – Modelos de bordas.
Fonte: (GONZALEZ; WOODS, 2008)

A Figura 8(a) ilustra uma *borda em degrau* onde a transição entre dois níveis de intensidade idealmente se dá na distância de 1 pixel. Essas bordas em degrau são comuns em imagens geradas por computador para uso em modelagem de sólidos e animações.

A Figura 8(b) ilustra uma *borda em rampa* onde a transição entre dois níveis de intensidade se dá pelo conjunto de pixeis entre os níveis extremos. Esse conjunto de pixeis é conhecido como rampa, e qualquer pixel contido nessa rampa é considerado parte da borda.

A Figura 8(c) ilustra uma *borda em forma de telhado* ou *roof edge* onde seu modelo é representado por linhas em uma região, determinada pela espessura e a nitidez da linha. A borda em telhado nada mais é do que uma linha com 1 pixel de espessura que atravessa uma região da imagem. Esse modelo de borda geralmente surge em imagens com profundidade, em digitalização de desenhos e em imagens de satélites.

Por fim, é importante ressaltar que esses modelos permitem escrever expressões matemáticas que posteriormente podem ser usadas em algoritmos de processamento de imagens.

2.9.2 Limiarização

A limiarização é uma técnica de segmentação que sugere a classificação dos pixeis de uma imagem de acordo com a especificação de um ou mais limiares.

A limiarização de intensidade é uma técnica que trabalha com apenas dois limiares. Podendo resolver casos onde uma imagem é composta por objetos claros em um fundo escuro.

Uma maneira de extrair esses objetos representados por *g(x,y)* é selecionando um limiar T. Qualquer ponto (x,y) na imagem em que $f(x,y) > T$ é chamado de *ponto do objeto*, e caso contrário, o ponto é chamado de *ponto de fundo*.

$$g(x,y) = \{^{1}_{0} \quad 1 \; se \; f(x,y) > T, \quad 0 \; se \; f(x,y) \leq T \tag{7}$$

Segundo Gonzalez e Woods (2008) o sucesso da limiarização de intensidade se dá devido a largura e profundidade dos vales. Existem alguns fatores que interferem nesses vales, sendo eles:

1. A separação entre picos. (quanto maior a distância entre os picos, maior será a chance de separação entre objeto e fundo);
2. O índice de ruido da imagem (o aumento de ruído amplia o objeto e o fundo);
3. O tamanho relativo dos objetos e do fundo;
4. A uniformidade da fonte de iluminação;
5. A uniformidade das propriedades de reflexão de imagem.

A seleção correta do valor do limiar é crucial para a obtenção de bons resultados no processo de segmentação baseado em limiar (PEDRINI; SCHWARTZ, 2008).

2.9.2.1 Limiarização Global e Local

A utilização de um único limiar global só é possível quando a intensidade dos pixeis do objeto são suficientemente diferentes da intensidade dos pixeis do fundo. Apesar das imagens conterem uma variabilidade de intensidade geralmente suficiente, a influência de variações nos níveis de cinza da imagem através de fatores como iluminação não-uniforme, ruído, entre outros, requerem o cálculo do valor do limiar para cada objeto (GONZALEZ; WOODS, 2008; PEDRINI; SCHWARTZ, 2008).

A utilização de múltiplos valores de limiar é conhecido como *limiarização local*. Uma maneira comum de determinar limiares locais, é através da análise de intensidade dos pixeis em uma região da imagem.

Algumas medidas estatísticas simples para calcular um limiar local são:

1. A média dos valores de intensidade em uma vizinhança local da imagem

$$T = média_v(p) \tag{8}$$

2. A mediana dos valores

$$T = mediana_v(p) \tag{9}$$

3. A média dos valores mínimo e máximo

$$T = \frac{min_v(p) + max_v(p)}{2} \tag{10}$$

Onde v é a vizinhança local de um ponto p na imagem (PEDRINI; SCHWARTZ, 2008).

2.9.3 Segmentação de Regiões

O objetivo da segmentação por regiões, é justamente detectar regiões diretamente nas imagens, ao invés de buscar as bordas que delimitam a região. Diversas propriedades são propostas para caracterizar uma região, tais como intensidade de cinza, cor, informação semântica ou textura (PEDRINI; SCHWARTZ, 2008).

As abordagens a seguir, focam em técnicas de segmentação baseada em encontrar as regiões de modo direto.

2.9.3.1 Crescimento de região

Basicamente o processo inicia com a escolha aleatória, determinística ou selecionada pelo usuário, de um conjunto de pixeis chamados de *sementes*. Essas sementes começam a crescer anexando os pixeis ao redor de acordo com a similaridade entre eles. Pedrini e Schwartz (2008) afirmam que os critérios de similaridade devem ser atualizados para cada novo pixel anexado a região em crescimento.

É evidente que um algoritmo baseado no crescimento de região exige um critério de parada. Logo, esse crescimento deve parar quando não houver mais pixeis satisfazendo os critérios de inclusão na região em questão. Alguns algoritmos estabelecem esses critérios como tamanho, formato, ou semelhança entre a intensidade de um candidato a entrar na região, e a média da intensidade dos pixeis na região (GONZALEZ; WOODS, 2008).

2.9.3.2 Divisão e fusão de região

O processo da divisão e fusão de região, propõe uma alternativa diferente baseado na divisão sucessiva da imagem total em quadrantes menores até que qualquer região satisfaça as propriedades.

Suponha que uma imagem seja representada por P e as propriedades que essa imagem deve obedecer sejam Q. A subdivisão de P será satisfeita quando qualquer região R_i, obedecer a condição $Q(R_i) = VERDADE$. Essa divisão é feita em quadrantes, logo se Q é FALSO para qualquer quadrante é necessário a divisão desse quadrante em subquadrantes. Essa técnica de divisão é conhecida como *quadtrees*, isto é, as árvores onde cada nó possui quatro descendentes. Se só a divisão for usada, ao final normalmente alguns quadrantes conterão propriedades idênticas, com isso, permite-se então a fusão dos mesmos (GONZALEZ; WOODS, 2008).

Por fim, em qualquer fase da divisão um dos seguintes procedimentos pode ser aplicado:

1. Dividir em quatro quadrantes qualquer região R_i para qual a condição $Q(R_i) = FALSO$;
2. Quando não houver mais a possibilidade de dividir, fundir as regiões adjacentes R_j e R_k para as quais $Q(P_jP_k) = VERDADE$;
3. Parar quando a fusão não for mais possível.

Segundo Pedrini e Schwartz (2008) estes métodos apresentados são particularmente úteis na aplicação da segmentação em imagens complexas.

Como visto nessa seção, o processo de convolução está presente em algumas abordagens da segmentação, porém foi feita uma descrição superficial desse processo, portanto, a próxima seção será dedicada a descreve-lo formalmente.

2.10 CONVOLUÇÃO

Como ja visto, convolução é uma operação entre duas imagens, onde há uma imagem de entrada em seu tamanho original e uma imagem menor conhecida como máscara ou matriz de convolução. Essa operação também é conhecida como núcleo (*kernel*), e segundo Piteri e Rodrigues (2011) funciona da seguinte maneira:

1. Criar uma máscara onde cada posição possui um determinado valor;

2. Alocar essa máscara sobre a imagem de entrada em uma posição (x,y);
3. Para cada posição na máscara é necessário executar o produto do valor da máscara pelo valor do pixel;
4. Fazer a somatória desses valores obtidos;
5. Substituir o valor da imagem de saída na posição (x,y) pelo resultado adquirido.

O processo de convolução tem a finalidade de *filtrar* uma imagem, geralmente com o objetivo de suprimir ruídos ou realçar bordas (MIRANDA, 2006). Para duas dimensões, em um domínio continuo o processo de convolução é representado pela Equação 11.

$$g(x,y) = k(x,y) * f(x,y) = \iint_{-\infty-\infty}^{\infty\infty} f(u,v)\,k(x-u,y-v)dudv \tag{11}$$

A forma discreta da Equação 11, é a Equação 12

$$g(x,y) = k(x,y) * f(x,y) = \sum_{u=0}^{u_{max}-1} \sum_{v=0}^{v_{max}-1} f(u,v)k([x-u],[y-v]) \tag{12}$$

e para ambas, f é a imagem periódica de entrada, k é a máscara periódica da convolução, g é a convolução periódica de saída, u_{max} é a largura do filtro, v_{max} a altura e

$[x-u] = (x-u) mod\ u_{max}$

$[y-v] = (y-v) mod\ v_{max}$

Uma forma alternativa para a Equação 12 que evita a operação *mod* é a Equação 13

$$g(x,y) = f(x,y) * k(x,y) = \sum_{u=-u_c}^{u_c} \sum_{v=-v_c}^{v_c} f(x-u,y-v)k(u+u_c,v+v_c) \tag{13}$$

onde (u_c,v_c) fica no centro da máscara, mas para isso acontecer a máscara deve ter dimensão ímpar, o que já é fato no cálculo da convolução no processamento de imagens digitais. Vale ressaltar que a dimensão da máscara influência no gasto computacional devido a operação convolucional ser executada para cada ponto da imagem original. O que leva a concluir que a máscara de convolução é uma janela móvel sobre a imagem.

Outro ponto levantado por Miranda (2006) é que a imagem de saída será menor que a de entrada devido ao ponto convolucional se encontrar no centro da máscara, o que não permite o calculo para a borda da imagem conforme ilustrado na Figura 9.

Nota-se que não será possível calcular duas linhas de pixeis na horizontal e mais duas na vertical, pois ao alocar a máscara nesses pontos, não existirão pixeis correspondentes na imagem de entrada. Porém existem alguns meios de resolver esse problema, o primeiro consiste em efetuar o preenchimento desses pontos não existentes com zero, porém não é o modo mais indicado pois pode gerar resultados distorcidos. Uma segunda alternativa consiste em ignorar

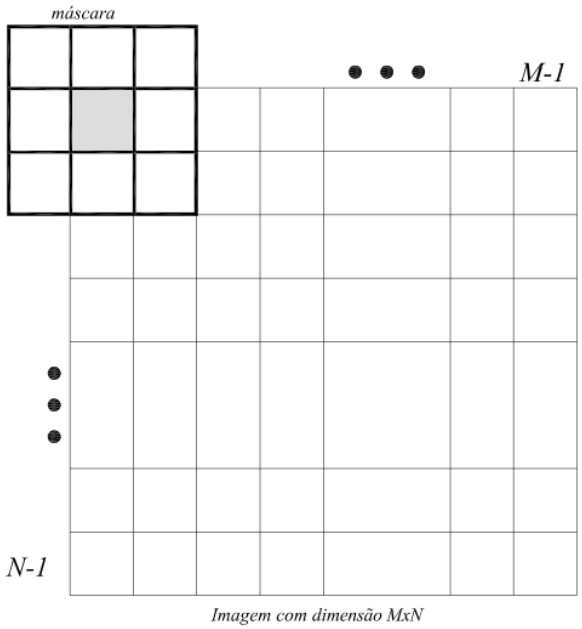


Figura 9 – Problema das bordas.

Fonte: (MIRANDA, 2006)

as bordas, mas isso implica na redução da imagem de saída. Extrapolar apesar de ser uma boa abordagem implica em esforço computacional, e por último a duplicação desses pontos é o método mais indicado por produzir bons resultados (MIRANDA, 2006).

3 REDES NEURAIS ARTIFICIAIS

Segundo Pedrini e Schwartz (2008) o uso das Redes Neurais Artificiais se tornou popular devido ao fato delas possuírem uma baixa dependência de domínios específicos, ou seja, isso permite que a mesma rede seja utilizada em problemas distintos.

Essa flexibilidade de uma RNA atraiu também a área de processamento de imagens, logo, surgiram dois conceitos que serão temas importantes para este trabalho, sendo eles o *Deep Learning* e a *Convolutional Neural Network*.

3.1 CONCEITO DE REDE NEURAL ARTIFICIAL

Segundo Russell (1991) uma RNA é composta por um conjunto de entidades altamente interligadas, chamadas de nós ou unidades. Os nós imitam o elemento biológico conhecido como neurônio. A Figura 10 exemplifica esse nó.

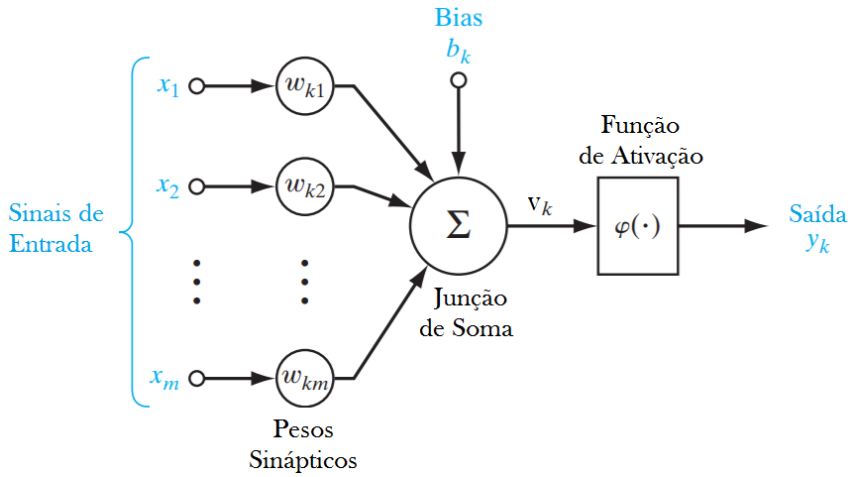


Figura 10 – Representação de um modelo não-linear de um nó denominado *k*.

Fonte: Adaptada de (HAYKIN, 2009)

Note que no modelo neural apresentado na Figura 10 inclui um *bias* aplicado externamente, denotado por *bk*. O *bias bk* tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se é positiva ou negativa, respectivamente (HAYKIN, 2009). Os sinais de entrada $\vec{X} = (x_1, x_2, ..., x_m)$ são valores resultantes da saída de outros nós, e os pesos $\vec{W} = (w_{k1}, w_{k2}, ..., w_{km})$ são valores pertencentes ao neurônio em questão.

Em termos matemáticos, é possível descrever o neurônio apresentado na Figura 10 com as Equações 14,15 e 16.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{14}$$

Onde u_k é a saída do combinador linear devido aos sinais de entrada.

$$y_k = \varphi(u_k + b_k) \tag{15}$$

Onde $\varphi(.)$ é a função de ativação, e y_k é o sinal de saída do neurônio em questão.

O uso do *bias bk* tem o efeito de aplicar uma *affine transformation*, ou seja, permitindo ou preservando relações paralelas de transformações conforme a Equação16 (HAYKIN, 2009).

$$v_k = u_k + b_k \tag{16}$$

Para casos em que a rede é uma *rede linear simples*, a função de ativação pode ser uma função do tipo linear, limiar ou sigmoidal.

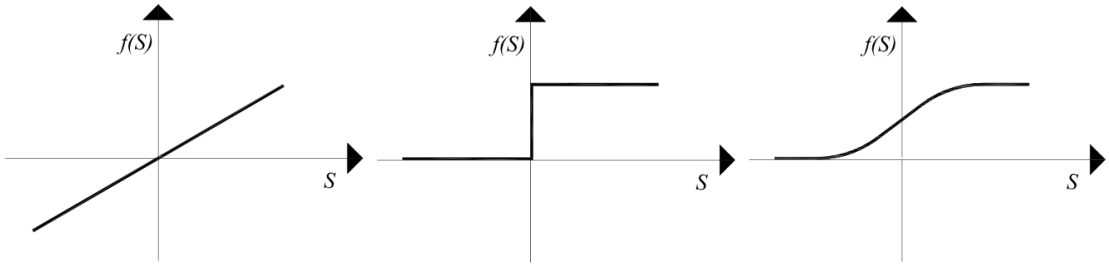
A função linear ilustrada na Figura 11(a) implica em retornar como saída o valor de *S* (FACELI et al., 2011).

Já a Função Limiar (do inglês *Threshold Function*) ilustrada na Figura 11(b) dado uma constante T, estabelece que, se $S > T$ então a saída recebe 1, caso contrario 0 (RUSSELL, 1991) (alternativamente, é possível utilizar o valor -1 ao invés de zero (FACELI et al., 2011)).

Por fim, a função sigmóide ilustrada na Figura 11(c) é a favorita dentre as funções de ativação, pois é uma função diferenciável, e garante um balanceamento adequado seja o comportamento linear ou não-linear. Em resumo, gera uma transição mais suave e contínua (HAYKIN, 2001).

Ao analisar a Figura 12 na qual representa de forma simples uma rede neural artificial é possível concluir que cada nó aceita um conjunto ponderado de entradas e responde com uma saída. A saída de um nó geralmente se torna a entrada de outro nó, as ligações que fazem a conexão entre os nós são unidirecionais, e a ponderação é ajustada pelos próprios nós (RUSSELL, 1991).

Todo o conteúdo sobre RNAs até aqui descrito vem da ideia inicial de



(a) Linear (b) Limiar (c) Sigmoidal

Figura 11 – Representação gráfica das funções de ativação.

Fonte: Adaptada de (FACELI et al., 2011)

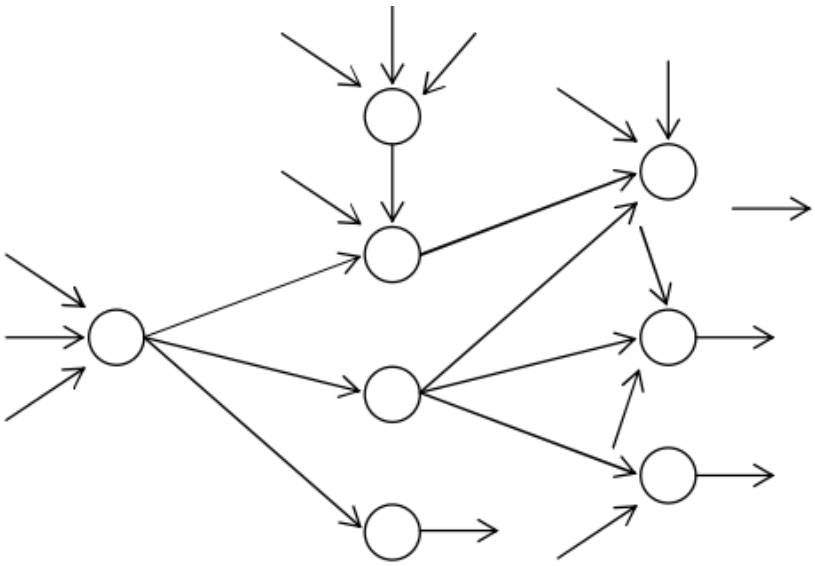


Figura 12 – Estrutura conectada por vários nós representados na Figura10, formando assim uma RNA.

Fonte: Adaptada de (RUSSELL, 1991)

Warren S. McCulloch e *Walter H. Pitts* conhecido como *Perceptron* que até então implementava somente uma camada. Porém ao se trabalhar com uma única camada as RNAs só resolvem problemas linearmente separáveis (MINSKY; PAPERT, 1969 apud RUSSELL, 1991), o que não é essencial já que são problemas de fácil compreensão. Porém as redes neurais artificiais não seriam o que são hoje se resolvessem somente problemas lineares, e a solução vem do uso de mais de uma camada, ou seja, o conceito de multicamadas.

3.2 MULTILAYER PERCEPTRON

As redes do tipo Multilayer Perceptron MLP possuem uma ou mais camadas intermediárias (ocultas), uma camada de entrada e outra de saída, onde a camada de entrada é constituída por nós que conectam os estímulos a rede (KOVACS, 2006). A arquitetura mais comum de uma MLP é a totalmente conectada, onde todos os nós pertencentes a uma camada qualquer estão conectados a todos os nós da camada seguinte (FACELI et al., 2011), conforme ilustrado na Figura 13.

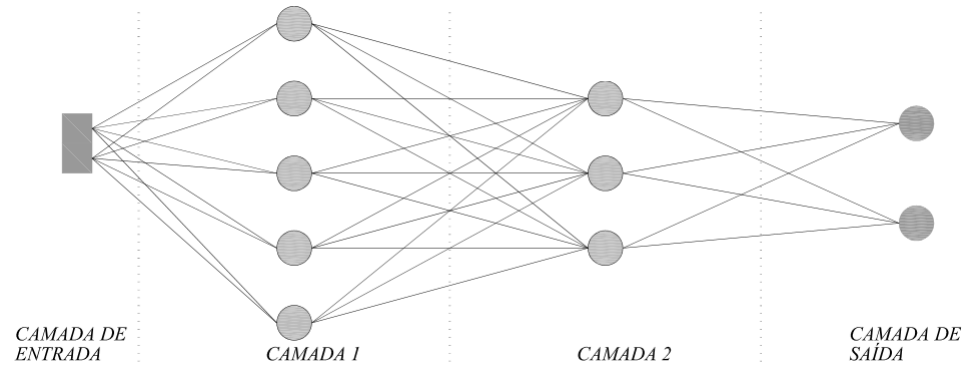


Figura 13 – Representação de um MLP totalmente conectada com duas camadas intermediárias. Fonte: Adaptada de (FACELI et al., 2011)

As MLP preferencialmente utilizam nas camadas intermediárias funções de ativação não lineares, como a função sigmoidal por exemplo. Cada nó realiza uma função especifica, está função é uma combinação das funções implementadas nos nós da camada anterior. O princípio é sempre o mesmo porém a medida que avança de uma camada a outra o processamento realizado e sua função correspondente vão se tornando mais complexos (FACELI et al., 2011).

Cada classe em um determinado problema deve possuir um nó que à represente na camada de saída. O treinamento da rede ocorre de modo que o vetor que contem as respostas desejadas para cada objeto de entrada, tem o valor 1 na posição associada a classe do objeto e 0 nas demais posições. A comparação entre o vetor dos nós na camada de saída e o vetor de valores desejados para essas saídas definem o erro cometido pela rede. Segundo Faceli et al. (2011), existem três casos em que essa comparação pode resultar, sendo eles:

1. Classificação correta - ocorre quando o maior valor produzido pela rede, é gerado pelo nó de saída que corresponde a classe correta do objeto;
2. Erro de Classificação - ocorre quando um nó de outra classe é quem produz o maior valor de saída;

3. Inapta - quando nenhum nó (ou mais de um) produz o maior valor, a rede se torna inábil para prever a classe do objeto.

O algoritmo mais comum para treinamento de MLPs, é o Back-propagation. Este algoritmo foi projetado por David E. Rumelhart juntamente com Geoffrey E. Hinton e Ronald J. Williams em 1986 (RUMELHART et al., 1986 apud FACELI et al., 2011). Hoje existem variações desse algoritmo que permitem uma melhor desempenho dependendo do problema a ser resolvido (FACELI et al., 2011), porém o conceito original ainda é eficiente.

A função de ativação do algoritmo back-propagation precisa ser contínua, diferenciável, e não decrescente (sigmoidal atende aos requisitos). É um algoritmo constituído de duas fases, a fase de ir para a frente ($forward$), e a fase de voltar para trás (*backward*). O processo $forward$ é o mesmo apresentado na explicação anterior sobre a estrutura de um nó, combinando agora com camadas. Em resumo, a primeira camada intermediária recebe os valores dos nós e seus respectivos pesos, aplica a função de ativação, e repassa os valores para os nós da camada seguinte. O nó da camada atual, jamais passará este valor para algum nó da camada anterior, isso porquê o algoritmo é baseado na topologia Feed-Forward[1]. Quando o processo chega na camada de saída é feita uma subtração entre o valor produzido pela rede e o valor desejado como saída, com isso obtêm-se o valor conhecido como erro.

Esse valor do erro é utilizado na fase *backward* para ajustar os pesos da entrada, logo o ajuste vai da camada de saída até a primeira camada intermediária e novamente obedece a regra da topologia Feed-Forward. O ajuste dos pesos de uma rede MLP é feito com a Equação 17 (FACELI et al., 2011; SILVA, 2005).

$$w_{jl}(t+1) = w_{jl}(t) + \eta x^j \delta_l \tag{17}$$

Onde:

- $w_{jl}(t)$ representa o valor do peso sináptico do nó l no tempo t;
- δ_l representa o erro associado ao nó l.
- x^j representa a entrada recebida por esse nó ($j-ésimo$ atributo de entrada ou a saída do $j-ésimo$ nó da camada anterior)
- η é conhecido como taxa de aprendizado, e tem forte influência no tempo necessário para convergência da rede. Caso a taxa de aprendizado seja pequena podem ocorrer muitos ciclos até que alcance um bom modelo. E caso a taxa de aprendizado seja elevada podem ocorrer oscilações que dificultam a convergência.

Para o algoritmo *Back-propagation* a estimativa de erro não pode se restringir somente a camada de saída, logo o algoritmo propõe duas equações para calcular esse erro, sendo estas

[1] A topologia de rede Feed-forward consiste em seguir em uma única direção, ou seja as conexões dos nós de uma camada só podem se conectar aos nós da camada seguinte (LIMA; PINHEIRO, 2014).

a Equação 18 que calcula o erro caso o nó pertença a camada de saída, e a Equação 19 que calcula o erro caso o nó pertença a qualquer camada intermediária.

$$\delta_l = y_l(1-y_l)(d_l - y_l) \tag{18}$$

Onde d_l representa a saída desejada, y_l representa a saída real, e

$$\delta_l = x'_l(1-x'_l)\sum \delta_k w_{lk} \tag{19}$$

Onde k representa todos os elementos posteriores a l. O Back-propagation propõe que essa estimativa do erro seja feita de modo que os nós da presente camada recebem a soma de todos os nós da camada seguinte conectados a ele, e que sejam ponderados conforme o valor do peso associado a essa conexão. Por fim, é necessário impor um critério de parada para o algoritmo Back-propagation. Esse critério pode ser por exemplo, uma quantidade máxima de ciclos, ou atingir uma taxa máxima de erro (FACELI et al., 2011).

De acordo com Braga et al. (2000), o aprendizado do algoritmo Back-propagation ocorre de forma supervisionada. O aprendizado supervisionado é quando a entrada e a saída desejadas são fornecidas por um supervisor externo. Basicamente o supervisor pode interferir no treinamento da rede, informando se a mesma está distinguindo os elementos de entrada de acordo com a saída esperada (BRAGA et al., 2000).

3.3 DEEP LEARNING

Algoritmos de aprendizagem simples não conseguem resolver problemas que atraem o foco da inteligência artificial hoje em dia, tais como reconhecimento de voz e de objetos. O crescimento dimensional dos dados torna-o exponencialmente mais difícil, logo o mecanismo de aprendizado de funções em algoritmos tradicionais são insuficientes em um espaço de alta dimensão. Esses espaços também impõem um alto custo computacional. O conceito de *Deep Learning* foi projetado para superar esses e outros obstáculos (GOODFELLOW et al., 2016).

Deep Learning é uma ferramenta com o objetivo de ensinar o computador como fazer coisas que somente os humanos são capazes de fazer através da percepção. Os algoritmos conceituados como Deep Learning possuem uma arquitetura motivada pela inteligência artificial, que emula o neocórtex do cérebro humano. Segundo Arel et al. (2010) o

neocórtex humano está associado a muitas habilidades cognitivas, mas o interesse não está no reconhecimento de sensações físicas, e sim na capacidade adquirida ao longo do tempo que permite aprender e representar observações com base nas regularidades que apresentam.

Arel et al. (2010) afirmam que além da dimensão espacial que contém dados reais, o *Componente Temporal* também desempenha um papel crítico na representação eficaz da informação, pois a informação é uma sequência observada de padrões que muitas vezes transmite um significado para o observador, e os fragmentos dessa informação postos de forma independente dessa sequência torna difícil a interpretação de partes isoladas. Por fim a captura de dependências presente no espaço-temporal com base em regularidades nas observações é visto como uma meta fundamental em sistemas baseados em Deep Learning.

Algoritmos conceituados em Deep Learning são designados para automatizar a extração de características dos dados sem diferir o nível de abstração do mesmo, ou seja, o algoritmo não é limitado a dados simples, pelo contrário, pode principalmente trabalhar com dados que possuem altos níveis de abstração. Deep Learning é um segmento importante para a inteligência artificial (AI), pois a extração de representações complexas necessárias para as tarefas de AI podem serem feitas sem supervisão, ou seja, são independentes do conhecimento humano, o que atinge o objetivo final da AI (NAJAFABADI et al., 2015).

A arquitetura de um algoritmo Deep Learning possui camadas de modo consecutivo. Cada camada aplica uma transformação não-linear em sua entrada e fornece uma representação de sua saída. Assim o empilhamento de camadas de transformação é a ideia básica desse algoritmo. Quanto mais camadas forem necessárias para a travessia dos dados, mais complexas se tornam as transformações não-lineares (NAJAFABADI et al., 2015).

O objetivo é que essas representações complexas sejam aprendidas de forma hierárquica para que cada camada tenha uma finalidade. Por exemplo, ao fornecer algumas imagens de rostos, na primeira camada pode-se aprender arestas em diferentes orientações. Na segunda camada essas arestas podem se tornar características mais complexas como diferentes partes de um rosto, como lábios, nariz e olhos. Na terceira camada incluí as características da camada anterior para eventualmente aprender outras características ainda mais complexas como formato do rosto. Por fim essas representações podem ser de uso em aplicações de reconhecimento facial (NAJAFABADI et al., 2015).

No entanto é necessário entender que algoritmos Deep Learning não necessariamente constroem uma sequência pré-definida de representações em cada camada, mas sim realizam transformações não-lineares em camadas diferentes (NAJAFABADI et al., 2015).

3.4 REDES NEURAIS CONVOLUCIONAIS

O termo Redes Neurais Convolucionais vem do inglês *Convolutional Neural Network* que vem da família multicamada das Redes Neurais, e foi projetada especialmente para uso de dados bidimensionais, tais como imagens e vídeos. A criação das CNN se deve a um trabalho anterior conhecido como Time-Delay Neural Network (TDNN) que reduzia as exigências de cálculo para aprendizagem através da partilha de pesos em uma dimensão temporal, o que proporcionou a inserção do uso de processamento de séries temporais (AREL et al., 2010). CNNs são redes neurais que utilizam convolução em pelo menos uma camada (GOODFELLOW et al., 2016).

De acordo com Dumoulin e Visin (2016), seja a entrada uma imagem, um som, ou uma coleção desordenada de recursos, e também seja qual for a dimensionalidade dessa entrada, a representação sempre poderá ser achatada em um vetor. E por mais que possuam diferenças peculiares, essas entradas compartilham propriedades importantes tais como:

1. São armazenadas como matrizes multidimensionais;
2. Apresentam um ou mais eixos para cada atributo (imagem possui atributos como eixo de largura e eixo de altura, um vídeo possui também o eixo para o tempo).

Um eixo de canal é usado para acessar diferentes aspectos dos dados. Por exemplo, em uma imagem colorida existem canais vermelhos, verdes e azuis, já em uma faixa de áudio estéreo existem o canal da direita e esquerda. Porém essas propriedades não são exploradas quando uma transformação é aplicada. Isso porque todos os eixos são tratados da mesma maneira e a informação topológica não é levada em consideração. Ainda assim é possível aproveitar a estrutura implícita dos dados, que permitem resolver algumas tarefas de visão computacional e reconhecimento de voz, acarretando no uso das convoluções discretas (DUMOULIN; VISIN, 2016).

A convolução discreta é uma transformação linear que preserva as noções de ordenação. É seletivo, pois apenas algumas unidades de entrada contribuem para a saída, e reutiliza parâmetros, ou seja, os mesmos pesos são aplicados a vários locais na entrada (DUMOULIN; VISIN, 2016). Logo, essas afirmações levam a discussão sobre as camadas de uma rede CNN discreta.

Segundo Nielsen (2015) o formato típico de uma camada da rede CNN consiste em três ideias simples, a primeira é *campo receptivo local*, a segunda *divisão de pesos*, e a terceira *pooling*, que estão exemplificados nas seções a seguir.

3.4.1 Campo Receptivo Local

Como já se sabe, os nós da camada de entrada são conectados aos nós de uma camada oculta. Porém, para este caso as conexões são feitas em pequenas regiões da imagem de entrada, ou seja, cada nó da camada escondida se conecta aos nós de uma região da camada de entrada (NIELSEN, 2015).

Para o nó escondido essa região é conhecida como campo receptivo local (LRF). Cada conexão aprende um peso, e o nó aprende um bias. É possível alegar que cada nó está aprendendo a analisar de forma particular seu campo receptivo local.

O campo receptivo local deve percorrer toda a imagem de entrada. Para cada LRF há um nó na camada escondida. Esse processo ocorre até que todos os LRFs gerem seus respectivos nós na camada escondida. Contudo, conforme apontado na Seção 2.10 as dimensões da camada gerada sempre será menor que as dimensões da camada de entrada. Porém, existem três abordagens (que podem ser trabalhadas juntas) para contornar esse fato. Essas abordagens são conhecidas como $transposed$, $padding$, e $strides$, e estão exemplificadas na Figura 14.

Onde $transposed$ são os nós que simulam uma nova borda contornando a imagem. $Padding$ é o valor que será usado para preencher os novos nós. E $strides$ é a distância entre duas posições consecutivas do processo de convolução.

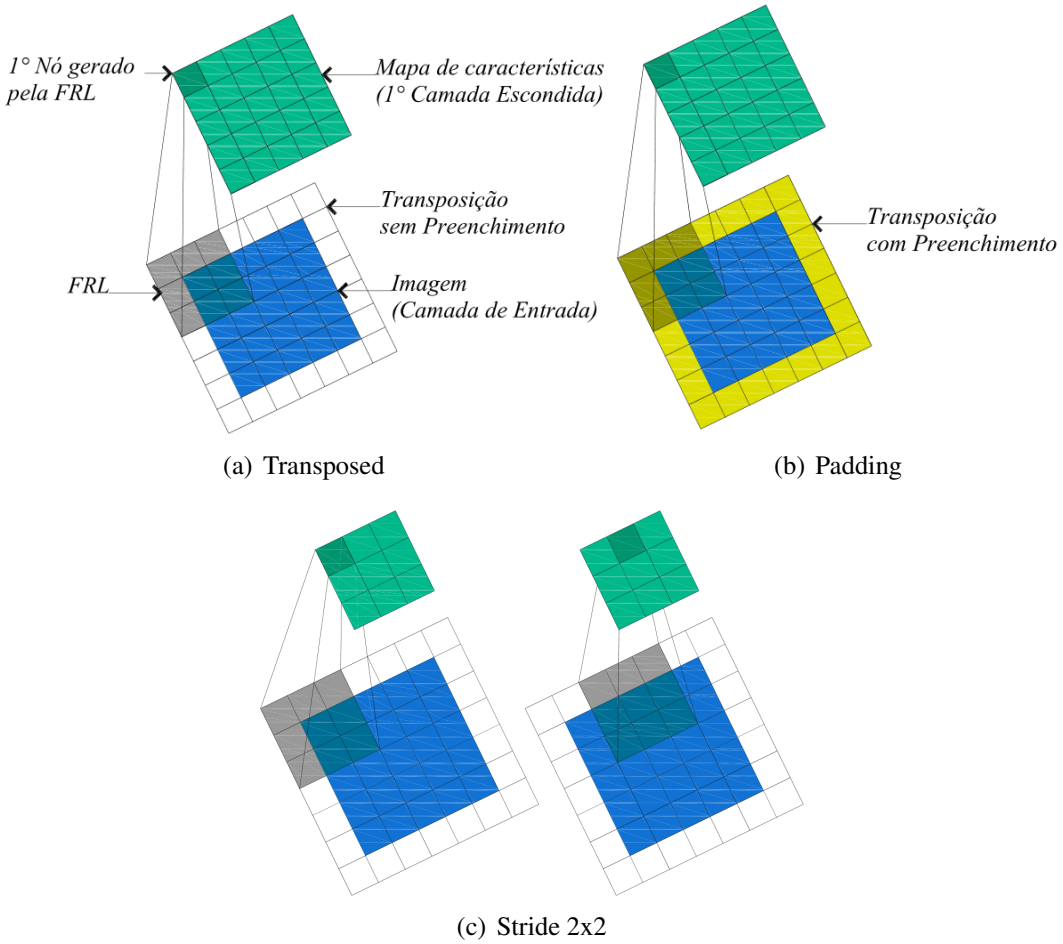
3.4.2 Compartilhamento de Pesos

Como mencionado, cada nó possui um bias e, nxn pesos ligados vindos do LRF. Supondo um exemplo onde será utilizado o mesmo peso e bias para cada um dos nós da camada escondida. Em outras palavras para o j,k-ésimo nó da camada escondida, a saída será dada pela Equação 20.

$$\sigma\left(b+\sum_{l=0}^{n}\sum_{m=0}^{n}w_{l,m}a_{j+l,k+m}\right) \tag{20}$$

Onde σ é uma função de ativação (ex: sigmoidal). b é o valor compartilhado bias. $w_{l,m}$ é uma matriz nxn de pesos compartilhados. E $a_{x,y}$ indica a ativação de entrada na posição x,y.

Com isso, é possível afirmar que todos os nós na primeira camada escondida são capazes de detectar as mesmas características, porém em pontos diferentes da imagem de



(a) Transposed (b) Padding

(c) Stride 2x2

Figura 14 – Formas de adaptação da saída. O campo receptivo local é o conjunto de nós onde a máscara esta posicionada.

Fonte: Adaptado de (DUMOULIN; VISIN, 2016)

entrada (NIELSEN, 2015). Para dar sentido a essa afirmação, supõe-se que o nó escolhe o valor e o bias no campo receptivo local que represente o focinho de um gato. Se o que se busca na imagem é um gato, essa capacidade se torna útil em outros locais da imagem, pois a detecção de características se torna invariante a translação devido aos pesos compartilhados. Ou seja, a rede encontrará o gato independente de onde ele apareça na imagem.

Em resumo, o conjunto de nós gerado por uma LRF é conhecido como mapa de características (*feature map*) ou máscara. As arestas que levam os pesos desse mapa de características são chamadas de pesos compartilhados (*shared weights*). E o bias definido para esse mapa de características é chamado bias compartilhado (*shared bias*). Os pesos e bias compartilhados são usados frequentemente na formulação de um mapa de características.

A estrutura descrita até agora detecta apenas um tipo de característica. Logo, para criar uma estrutura que vá além de uma característica é necessário ter mais de um mapa de característica. A Figura 15 ilustra uma camada convolucional composta por diferentes mapas

de características (NIELSEN, 2015).

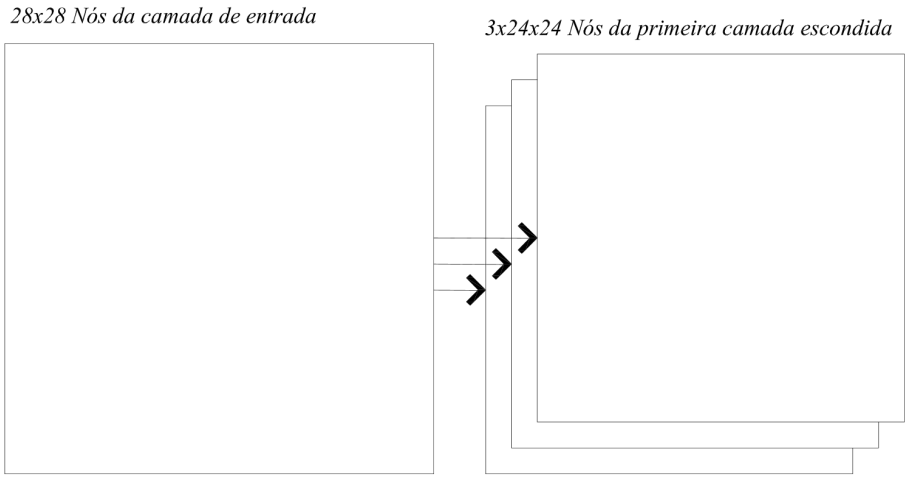


Figura 15 – Composição de uma camada convolucional em diferentes mapas de características onde o quadrado da esquerda representa um conjunto de nós com dimensão de $28x28$ **que através de um conjunto de pesos compartilhados com dimensão de** $5x5$ **e um único bias compartilhado, gerou três mapas de características com dimensão de** $24x24$ **cada um.**

Fonte: Adaptado de (NIELSEN, 2015)

Voltando a analogia da busca por um gato na imagem, ao pensar que o focinho é apenas uma característica que descreve um gato, talvez não seja suficiente para dizer se há ou não um gato na imagem (se houver um gato de costas, ou um coelho que é um animal com focinho semelhante). Para que a rede realmente localize um gato, ela precisa conhecer mais características do felino, tais como rabo, orelhas, patas, bigode, e cada uma dessas características devem pertencer a um mapa de características. Quanto mais características de um gato a rede souber, mais a rede saberá o que é um gato, e mais chances terá de encontrá-lo na imagem, independentemente da posição do mesmo. E isso resume a importância do mapa de características com pesos compartilhados.

3.4.3 Pooling (Agrupamento)

Uma função pooling substitui a saída da rede em um determinado local através de um resumo estatístico das saídas mais próximas. Por exemplo, a operação *max pooling* emite o maior valor dentro de $max(t)$ onde t é um conjunto de valores dentro de uma região conforme

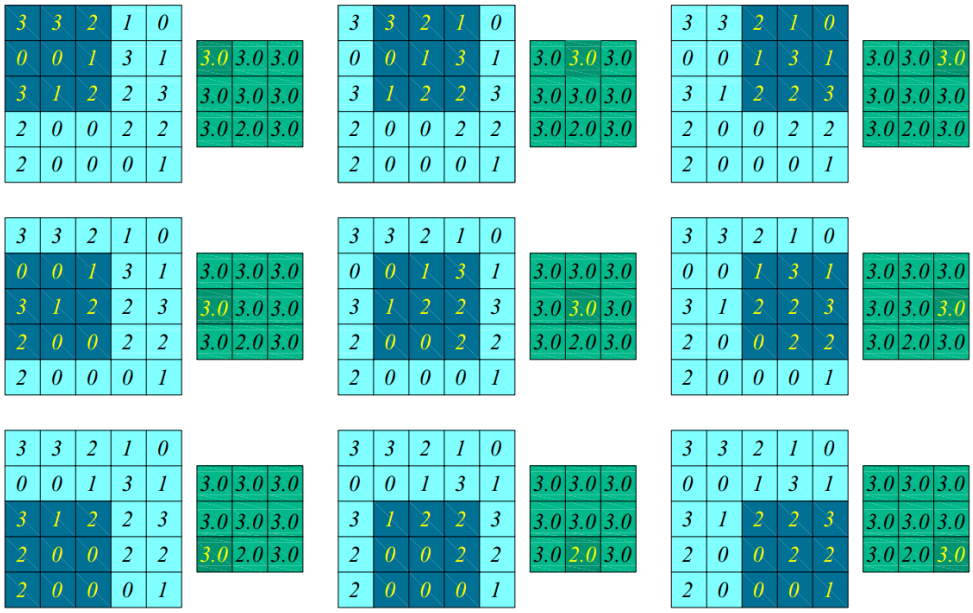


Figura 16 – Representa o processo da função max pooling de uma imagem de entrada com dimensão de 5x5 usando Stride de 1x1, resultando em uma saída de dimensão 3x3.

Fonte: Adaptado de (DUMOULIN; VISIN, 2016)

ilustrado na Figura 16.

Uma outra função pooling popular é a *Avgpool* que calcula a média desse conjunto de valores, ou então a média ponderada com base na distância a partir de um pixel central (GOODFELLOW et al., 2016).

Em todos os casos a função pooling ajuda a fazer com que a representação torne-se invariante para pequenas traduções da entrada. Invariância de tradução significa que ao traduzir a entrada de uma pequena quantidade, os valores da maioria das saídas produzidas pela função pooling não mudam, ou seja, ao aplicar pequenas modificações em uma imagem que contém um gato, a imagem ainda conterá um gato. (GOODFELLOW et al., 2016; NIELSEN, 2015).

Invariância a tradução local pode ser uma propriedade muito útil se o objetivo for encontrar alguma característica ao invés de saber onde esta característica está. O uso de pooling impacta fortemente na elaboração da rede, onde deve-se considerar que a função aprendida pela camada precisa ser invariante a pequenas traduções. Quando esta afirmação é verdade, proporciona uma melhora significativa na eficiência estatística da rede (GOODFELLOW et al., 2016).

O processo pooling resume as respostas ao longo de uma região, o que permite usar menos unidades de agrupamento, relatando estatísticas de resumo para as regiões de agrupamento espaçadas com k pixeis de distância (ao invés de 1 pixel de distância). E isso consequentemente melhora a eficiência computacional da rede, pois a próxima camada tem k

vezes menos insumos para processar (GOODFELLOW et al., 2016).

Por fim, em muitas tarefas o agrupamento é essencial para o manuseio de entradas de tamanho variável. A entrada para a camada de classificação deve ter um tamanho fixo. E isso pode ser manipulado através da variação do tamanho de um deslocamento entre as regiões de agrupamento, fazendo com que a camada de classificação sempre receba o mesmo número de estatísticas resumidas independentemente do tamanho da entrada (GOODFELLOW et al., 2016).

4 MATERIAIS E MÉTODOS

Nesse capítulo serão descritos os materiais e métodos que foram utilizados para o desenvolvimento dos objetivos no qual visa identificar objetos em uma imagem, sendo estes latas e/ou garrafas.

4.1 MATERIAIS

O bom desempenho de uma rede depende de uma boa programação. Mas este desempenho também pode ser influenciado por uma boa ferramenta. Sendo assim, a busca por tecnologias que tragam um bom custo benefício deve ser levado em consideração, pois a base do treinamento de uma rede neural se resume em cálculo de larga escala o que por sua vez requer muito processamento.

A dependência do processamento em larga escala demanda o uso de uma unidade de processamento gráfico (GPU), isso porque a GPU possui dedicação exclusiva ao processamento gráfico tornando-a muito mais rápida e eficiente em comparação com a unidade central de processamento (CPU) que gerencia todo processo condicionado no computador.

Os *Frameworks* para treinamento de rede mais conhecidos como *TensorFlow*, *Caffe* e *Darknet* oferecem suporte tanto para o uso da CPU quanto para GPU, e todos integram a biblioteca *OpenCV*. Quando opta-se pelo uso da GPU é necessário um integrador entre o *Framework* e a GPU. Para os *Frameworks* citados, o mais utilizado é o *Compute Unified Device Architecture* (CUDA). Apesar do CUDA ser bastante utilizado e possuir suporte através dos *Frameworks* é de uso exclusivo de placas da marca NVIDIA.

Em resumo, para dar inicio a parte prática é necessário um computador, o banco de imagens, um *Framework*, e uma rede. Os quais serão descritos nas próximas sessões.

4.1.1 Computador

Em relação ao hardware não há muito o que se explorar, pois dentre as opções disponíveis a que melhor se encaixou possui as seguintes informações: Notebook DELL modelo Inspiron 15 7000 Series 7559 com memória de 16GB, processador Intel® Core™ i7-6700HQ CPU @ 2.60GHZ x 8, com uma GPU modelo GeForce GTX 960M/PCIe/SSE2, e armazenamento SSD de 250GB. O sistema operacional instalado é o Ubuntu 16.04 LTS de 64-bit.

4.1.2 Banco de Imagens

Base de Dados - O website ImageNet[1] é uma base de imagens online que possui mais de 14 milhões de imagens, e atendeu boa parte das imagens usadas no treinamento, porém a rede exigia certas variações nas imagens das quais não foram encontradas entre as imagens obtidas no ImageNet, logo foi necessário buscar algumas imagens através do Google[2].

4.1.3 Framework

Como já mencionado os *Frameworks* mais conhecidos são *TensorFlow*, *Caffe* e *Darknet*. Todos possuem seus prós e contras, porém são capazes de treinar as mesmas redes, como por exemplo, a LeNet, GoogleNet, YOLO, etc.

O *Tensorflow* possui uma interface de treinamento muito rica em informações, se utiliza de gráficos coloridos e logs que traduzem o passo-a-passo da rede. Já o *Caffe* é o mais antigo, dentre os citados, foi utilizado em inúmeros trabalhos, e existem diversos tutoriais sobre como organizar e treinar uma rede. Já o *Darknet* possui uma fase pós treinamento muito eficiente, com uma única linha é possível visualizar o resultado do treinamento em uma imagem passada por parâmetro.

O *Framework* é para ser um facilitador de treinamento de uma rede, sua escolha

[1] http://image-net.org/index
[2] https://www.google.com/imghp?hl=pt-pt

pode gerar impacto no tempo de treinamento da rede, mas não no resultado final, pois este está diretamente relacionado a estrutura da rede. Não existe um comparador entre eles pra saber qual é melhor, até porque há inúmeras variáveis a serem consideradas, como qualidade/quantidade/tamanho das imagens, preparação do *dataset*, variáveis de saída, tempo de iteração, tempo de convergência, quantidade de classes, etc. Logo o critério de escolha foi com base no aproveitamento geral para aquisição de resultados, contudo o *Framework* utilizado foi o *Darknet*.

Segundo Redmon (2013–2016) 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 necessário em casos que se deseja utilizar a GPU, e o *OpenCV* é necessário quando há uma variedade maior nos tipos de imagens suportadas, logo, ambos são necessários.

Para instalação do *CUDA* são necessários duas ferramentas, a *The NVIDIA CUDA Toolkit* e a biblioteca *The cuDNN library*.

The NVIDIA CUDA Toolkit[3] - além de bibliotecas matemáticas, possui também um compilador para GPUs, e ferramentas para depuração que otimizam o desempenho das aplicações. O ambiente fornecido por esta ferramenta permite programar em C e C++ para desenvolver aplicações que podem ser executadas através da GPU. É uma ferramenta gratuita porém exclusiva para placas de vídeo que suportam a tecnologia CUDA.

The NVIDIA CUDA Deep Neural Network library (cuDNN)[4] - é uma biblioteca desenvolvida para combinar a aceleração por GPU juntamente com os princípios primitivos de redes neurais e deep learning. Oferece uma alta sintonia entre implementações que possuem rotina padrão tais como as fases forward e backward do algoritmo back-propagation, convolução, pooling, normalização, camadas de ativação, etc. Esta biblioteca é de uso exclusivo da ferramenta *The NVIDIA CUDA*.

OpenCV[5][6] - foi projetado visando a eficiência em visão computacional e com um enfoque em aplicativos de processamento em tempo real. Desenvolvida em C e C++, permite o processamento *multi-core*, além de ser multiplataforma e *open source*.

[3] https://developer.nvidia.com/cuda-toolkit
[4] https://developer.nvidia.com/cudnn
[5] https://opencv.org/
[6] https://pt.wikipedia.org/wiki/OpenCV

4.1.4 Rede

Dentre as inúmeras redes neurais convolucionais existentes que visam a busca de objetos em uma cena, tais como *LeNet*[7], *Inception/GoogleNet*[8], *AlexNet*[9], *Single Shot MultiBox Detector*[10] (SSD), a utilizada para esse trabalho foi a *YOLO*[11].

Antes de explicar a estrutura da rede utilizada, vale dispor algumas informações. O termo YOLO é uma abreviação para *You Only Look Once*. De acordo com Redmon e Farhadi (2016) existem duas estruturas de rede YOLO, sendo elas, a *Tiny-YOLO*, e a *YOLO* padrão.

A *Tiny-YOLO* em resumo é baseada no *modelo de referência do Darknet*[12], é mais rápida porém com uma taxa de acerto menor que a *YOLO* padrão. Com GPU é capaz de alcançar uma faixa 155 quadros por segundo (FPS) o que a torna viável para detecção de objetos em vídeos e/ou tempo real.

Já a *YOLO* padrão é baseada no modelo de rede *Extraction*[13], é capaz de processar imagens a uma faixa de 45FPS. Por um lado, o quadrante de localização do objeto não é tão preciso, mas se mostrou menos propícia a encontrar falsos positivos dentre as redes que possuem a mesma finalidade que ela (REDMON et al., 2015).

Dentre as versões existentes do YOLO, a que foi usada é a *YOLO* padrão versão 2 lançada em 2016. Segundo Redmon e Farhadi (2016) o melhor desempenho de uma rede geralmente está associado a redes maiores ou da combinação de vários modelos, porém, a *YOLOv2* ao invés de crescer, diminuiu suas camadas em comparação com a primeira versão, o que resultou em uma melhora significativa em comparação com outras redes.

A arquitetura da rede *YOLOv2* é baseada na arquitetura de rede *Darknet-19* criada pelos mesmos autores. A *Darknet-19* foi criada com o objetivo de treinar com o *dataset Imagenet* composto por mil classes. Sua arquitetura é composta por 19 camadas convolucionais intercaladas com mais 5 camadas que aplicam o processo *max-pooling* exemplificado na Sessão 3.4.

Os dados apresentados na Tabela 1 são referentes ao funcionamento da rede *Darknet-19* onde a primeira coluna representa a ordem estrutural da rede intercalada entre as camadas e os modelos de agrupamentos, a segunda coluna representa o número de filtros que serão

[7] http://yann.lecun.com/exdb/lenet/
[8] https://www.cs.unc.edu/w̃liu/papers/GoogLeNet.pdf
[9] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[10] https://arxiv.org/abs/1512.02325
[11] https://pjreddie.com/darknet/yolov1/
[12] https://pjreddie.com/darknet/imagenet/#reference
[13] https://pjreddie.com/darknet/imagenet/#extraction

Tabela 1 – Estrutura da Rede Darknet-19 (REDMON; FARHADI, 2016)

Type	Filters	Size/Stride	Output
Convolutional 1	32	3 x 3	224 x 224
Maxpool		2 x 2 /2	112 x 112
Convolutional 2	64	3 x 3	112 x 112
Maxpool		2 x 2 / 2	56 x 56
Convolutional 3	128	3 x 3	56 x 56
Convolutional 4	64	1 x 1	56 x 56
Convolutional 5	128	3 x 3	56 x 56
Maxpool		2 x 2 / 2	28 x 28
Convolutional 6	256	3 x 3	28 x 28
Convolutional 7	128	1 x 1	28 x 28
Convolutional 8	256	3 x 3	28 x 28
Maxpool		2 x 2 / 2	14 x 14
Convolutional 9	512	3 x 3	14 x 14
Convolutional 10	256	1 x 1	14 x 14
Convolutional 11	512	3 x 3	14 x 14
Convolutional 12	256	1 x 1	14 x 14
Convolutional 13	512	3 x 3	14 x 14
Maxpool		2 x 2 / 2	7 x 7
Convolutional 14	1024	3 x 3	7 x 7
Convolutional 15	512	1 x 1	7 x 7
Convolutional 16	1024	3 x 3	7 x 7
Convolutional 17	512	1 x 1	7 x 7
Convolutional 18	1024	3 x 3	7 x 7
Convolutional 19	1000	1 x 1	7 x 7
Avgpool		Global	1000
Softmax			

Fonte: (REDMON; FARHADI, 2016)

aplicados naquela camada. Já a terceira coluna denota o tamanho dado em x e y da máscara e também o *Stride*. E a última coluna representa a dimensão do mapa de característica após ter sido submetido aos processos descritos nas colunas anteriores. O *Softmax* citado na última linha, é uma função de ativação para redes com multi-classes, na qual aplica uma função de probabilidade de classe, onde a soma de todos os valores dados as classes deve ser igual a 1[14]. Logo, a classe que possuir o maior valor estatisticamente é a melhor saída.

Como dito, a rede *YOLOv2* é similar a rede *Darknet-19*, onde possui a mesma estrutura até a Convolução 18 e a partir desta possui mais cinco camadas convolucionais descritas na Tabela 2.

Como pode ser observado na Tabela 2 a dimensão de saída é um valor ímpar, isso porque o objetivo é que a dimensão dos mapas de características seja ímpar para que não haja

[14]http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

Tabela 2 – Estrutura da Rede YOLOv2 (REDMON; FARHADI, 2016)

Type	Filters	Size/Stride	Output
Convolutional 19	1024	3 x 3	7 x 7
Convolutional 20	1024	3 x 3	7 x 7
route 13			
Convolutional 21	64	1 x 1	14 x 14
reorg re		/ 2	7 x 7 x 256
route re 20			
Convolutional 22	1024	3 x 3	7 x 7
Convolutional 23	AA	1 x 1	7 x 7 x AA

Fonte: (REDMON; FARHADI, 2016)

mais de um ponto central (REDMON; FARHADI, 2016), logo, é por esse motivo que na versão *YOLOv2* a dimensão de entrada da imagem passou a ser recomendada de 416x416 ao invés de 448x448 como é na *YOLOv1*.

Ainda na Tabela 2 o *route* indica qual camada buscar para próxima convolução, quando há mais de uma camada sendo indicada é necessário fazer uma concatenação dos filtros mantendo a dimensão do mapa. Já o *reorg* com *Stride = 2* reduz o tamanho do mapa na metade porém aumenta o filtro em quatro vezes [15]. Por fim, a última camada de convolução é responsável pela classificação e localização de objetos. Logo a dimensão da saída de *7x7* representa o número de grades em x e y que será posto sobre a imagem, já o valor do filtro representado por *AA* é dado pela Equação 21 onde *class* representa o número de classes que a rede reconhece.

$$AA = (class + 5) * 5 \tag{21}$$

Segundo Redmon e Farhadi (2016) o objetivo com a rede YOLO é propor um mecanismo para treinamento conjunto para que a rede possa fazer detecção e classificação de objetos. Esse mecanismo utiliza imagens rotuladas para aprender informações especificas com predição de coordenadas para caixas delimitadoras e também para classificar objetos comuns. Essa informação trás três termos importantes que são usados na rede, sendo eles:

1. Imagens rotuladas é o mesmo que *Ground Truth*;
2. Predição de coordenadas é o mesmo que *Anchor Boxes*;
3. Caixas delimitadoras é o mesmo que *Bounding Boxes*.

De acordo com (REDMON; FARHADI, 2016) embora os *Anchor Box* possuam a mesma finalidade na qual visa auxiliar na criação dos *Bounding Box*, sua lógica de criação sofreu mudanças. A primeira mudança foi armazenar *offsets* ao invés de coordenadas baseado

[15] https://github.com/AlexeyAB/darknet/issues/120#issuecomment-313371171

na rede de proposta de região (RPN). Já a mudança mais significativa foi usar o algoritmo K-Means para predizer as dimensões de cada *Anchor Box* ao invés de setar manualmente na rede as dimensões e posteriormente encarregar a rede de ajustar conforme seu aprendizado. Contudo, o termo *Anchor Box* passou a ser chamado também de *Dimension Cluster*. O algoritmo K-Means prevê cinco (K=5) *Anchor Box* na imagem, e aplicado a uma fórmula com os *Ground Truth* permite a predição dos *Bounding Box*.

A criação dos *Bounding Box* obedece aos valores de saída da última convolução da rede. No exemplo da Tabela 2 a saída é igual a *7x7xAA*. O valor de *AA* é dado pela Equação 21, o objetivo dessa equação é dizer a rede quantos parâmetros cada *Bounding Box* possui e também a quantidade de *Bounding Box* por célula de grade. A quantidade de células de grades em *x* e *y* para o exemplo é de *7x7*. A Figura 17 ilustra esse processo.

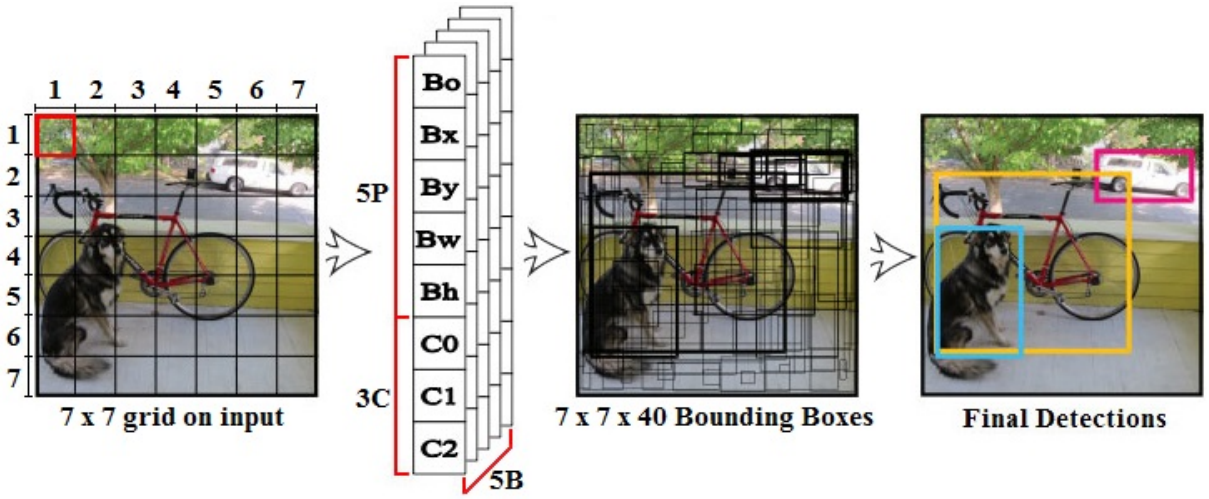


Figura 17 – O cenário considera que a rede possui três classes (3C), podendo ser C0 para cachorro, C1 para bicicleta e C2 para carro. Somados com cinco parâmetros padrões (5P) para cada Bounding Box totalizando oito parâmetros. Sendo cinco Bounding Box (5B), totaliza 40 dados para uma célula de grade. O processo é aplicado a todas as células.

Fonte: Adaptado de (REDMON et al., 2015)

Conforme ilustrado na Figura 17 os cinco parâmetros padrões de um *Bounding Box* representados por *Bo, Bx, By, Bw, Bh* são confiabilidade, coordenadas *x* e *y* do centro, largura e altura a partir do centro respectivamente. Segundo Redmon e Farhadi (2016) a célula de grade é responsável por saber se há um *Bounding Box* com um objeto pertencente a uma classe, logo, os valores das coordenadas dos *Bounding Box* são relativos a localização da célula de grade, isso limita o valor do *Ground Truth* entre 0 e 1, para isso é aplicado uma logística de ativação para restringir as previsões da rede nesse intervalo.

Os valores de *Ground Truth* auxiliam no cálculo de perda para os positivos verdadeiros, logo, dentre os inúmeros *Bounding Boxes* criados é necessário que apenas um seja responsável

pelo objeto. Para isso é selecionado o *Bounding Box* com a maior taxa de intersecção sobre a união (IoU) comparado ao *Ground Truth*. Essa estratégia leva a especialização entre as previsões dos *Bounding Box* onde cada previsão melhora a previsão de determinados tamanhos e proporções.

Diferentes dos *Bounding Box* e dos *Anchor Box* que são preditos pela rede, os *Ground Truth* devem acompanhar as imagens da faze de aprendizagem, ou seja, cada imagem deve ter um arquivo respectivo a ela, onde o conteúdo desse arquivo deve ser conforme ilustrado na Figura 18. Cada linha representa um objeto, e a primeira coluna representa a classe que esse objeto pertence, a segunda e a terceira coluna representam a posição x e y respectivamente do centro do objeto, e a quarta e quinta coluna representam a largura do objeto em x e y respectivamente.

```
1  0 0.7708333333333334 0.5858333333333333 0.22833333333333336 0.545
2  1 0.2275 0.4758333333333334 0.195 0.765
```

Figura 18 – Exemplo da estrutura do arquivo aceito pelo YOLO extraído da Figura 19(b).

Fonte: Modelado no BBox Label Tool versão disponibilizada por Valiati (2018)

Esses arquivos são gerados por uma ferramenta chamada *BBox Label Tool*, porém é necessário marcar na imagem os pontos de referência conforme ilustrado na Figura 19(b).

(a) Estrutura do segundo arquivo gerado pelo BBox Label Tool.
Linha 1 quantidade de objetos,
Linha 2 parâmetros da Lata,
Linha 3 parâmetros da Garrafa.

(b) Marcação do Objeto. Primeiro objeto marcado foi a Lata e o segundo a Garrafa.

Figura 19 – A Figura (a) representa as posições dada em pixel dos retângulos desenhado sobre os objetos na Figura (b).

Fonte: Modelado no BBox Label Tool versão disponibilizada por (VALIATI, 2018)

A versão da ferramenta utilizada foi disponibilizada por (VALIATI, 2018). A

ferramenta gera tanto o arquivo que o YOLO lê quanto um segundo arquivo que pode ser usado para calcular o IoU. A Figura 19(a) ilustra a estrutura do segundo arquivo onde a primeira linha representa a quantidade de objetos, e as demais a posição do objeto dado em pixel, sendo uma linha para cada objeto.

4.2 MÉTODOS

Os seguintes métodos foram seguidos para atingir o objetivo geral.

1. Montar a base de imagens para treinamento que consiste em 6 mil imagens de latas, e mais 6 mil imagens de garrafas;
2. Instalar e configurar:
   (a) The NVIDIA CUDA Toolkit;
   (b) The NVIDIA CUDA Deep Neural Network library (cuDNN);
   (c) OpenCV;
   (d) Darknet;
   (e) BBox-Label-Tool;
3. Configurar a rede YOLOv2 para atender ao objetivo desse trabalho;
4. Criar os arquivos *.txt* para o conjunto de imagens, nas quais a rede exige para treinamento usando a ferramenta BBox-Label-Tool;
5. Treinar a rede com a base de imagens;
6. Testar o desempenho da rede com imagens distintas das quais foram usadas no treinamento;
7. Treinar a rede com mais imagens;
8. Testar novamente o desempenho;
9. Treinar a rede com mais imagens;
10. Ultimo teste de desempenho.

Etapa 1: essa etapa consistiu em adquirir o volume de imagens mencionado pensando em montar um *dataset* extenso prevendo a ocorrência de imagens repetidas ou com resolução muito inferior. As imagens necessariamente precisam ter características de um ambiente real. Isso significa que o objeto deve sofrer influência da luz. As imagens não necessariamente precisam ter boa qualidade, e também o objeto ao qual se deseja localizar pode estar em diferentes ângulos, posições, distância, localização, e ambiente.

Foram selecionadas mais 300 imagens que não continham o objeto, mas sim objetos que poderiam induzir a rede a rotula-los em uma classe caso surgissem no contexto da imagem a ser analisada, são os chamados falsos positivos. Foram selecionados 25 imagens de seis classes diferentes, sendo elas:

1. Barris de metal, madeira ou plástico;
2. Recipientes como taças, xícaras, copos de vidro, plástico colorido;
3. Embalagens de Óleo sendo de lata ou plástico;
4. Latas de Leite em pó;
5. Enlatados de alimentos variados;
6. Embalagens de plástico de refrigerantes, água, sucos, com tamanhos e formas variadas.

As seis classes totalizam 150 imagens, e as demais 150 imagens são de contextos variados, contendo animais, pessoas, móveis, alimentos, automóveis, louças, eletrodomésticos, etc.

Etapa 2: Essa etapa consistiu em instalar as ferramentas citadas. O *The NVIDIA CUDA Toolkit* foi o mais complexo, pois o passo-a-passo no site não é suficiente, foi necessário executar os comandos fora do modo gráfico do sistema operacional, e desinstalar a placa de vídeo e reinstalar para a ferramenta funcionar. Já as demais ferramentas bastou seguir o tutorial nas páginas oficiais.

Etapa 3: Na rede YOLOv2, são necessários apontar a quantidade de classes e o valor do filtro da última camada de convolução, esse valor é dado pela Equação 21. Uma outra alteração importante são nos valores de *batch* e *subdivisions*. O valor do *batch* representa a quantidade de imagens no lote que será aplicado a normalização, essa quantidade impacta diretamente na memória da GPU. Já o *subdivisions* divide o valor do *batch* para diminuir a quantidade de imagens no lote, logo o *batch* deve ser múltiplo do valor no *subdivisions*.

Também é possível alterar a dimensão das imagens dentro da rede, através dos parâmetros *width* e *height*. Porém o ideal é manter em 416x416px. O número de iterações é dado pelo parâmetro *max_batches*.

Por fim outro parâmetro que pode ser alterado é o *Random*, este determina se a rede pode alterar a dimensão da imagem a cada dez iterações. As dimensões devem ser múltiplas de 32 e o valor mínimo é de 320x320, e o máximo de 608x608, Segundo Redmon e Farhadi (2016) o objetivo desta variação é que a rede aprenda a detectar objetos em tamanhos diferentes, e também em resoluções variadas.

Etapa 4: Nessa etapa foram criados os arquivos referente as imagens que a rede precisa para iniciar o treinamento. Esses arquivos foram criados conforme demanda, logo, o primeiro conjunto de imagens para treinamento consistiu em 3 mil imagens de latas e mais 3 mil imagens

de garrafas.

As imagens denominadas falsos positivo também necessitam do arquivo, porém o conteúdo deve estar em branco.

Etapa 5: Os parâmetros da rede podem ser divididos em dois tipos, onde um é referente as necessidades do usuário, e o outro é referente aos recursos disponíveis.

Os parâmetros de acordo com as necessidades do usuário definem a quantidade de classes e o número de iterações:

- *class* = 2 : 0 para lata, e 1 para garrafa;
- *filter* = 35 : filtro referente a última camada de convolução da rede dado pela Equação 21;
- *max_batches* = 10000 : quantidade de iterações;

Já os parâmetros de acordo com os recursos disponíveis são:

- *width* = 416 : largura da imagem;
- *height* = 416 : altura da imagem;
- *batch* = 64 : quantidade total de imagens usado para calcular a acurácia em uma iteração;
- *subdivisions* = 8 : valor que define a quantidade de imagens em um grupo de generalização no qual a rede calcula a média de IoU e a acurácia por grupo. A quantidade de grupos por iteração é o resultado da divisão entre o *batch* e o *subdivision*;
- *Random* = 0 : desligado;

As dimensões de largura e altura da imagem não puderam ultrapassar o valor de 480x480px para a imagem de entrada com os valores descritos no *batch* e *subdivisions*, pois ultrapassava a memória disponibilizada na placa de vídeo. E os valores do *batch* e *subdivisions* foram mantidos porque apresentaram um bom desempenho entre os demais testados.

Já o *Random* foi desligado porque novamente a dimensão escolhida poderia ser acima de 480x480px, e mesmo alterando o *framework* para limitar o valor em 480 ao invés de 608, seria contrário a proposta dos autores em relação ao uso desse recurso.

Como o modelo da rede é supervisionado, são necessários imagens para treinamento e imagens para teste, essa divisão foi feita de forma que 75% das imagens são destinadas ao treinamento, e os demais 25% para teste.

Essa mesma rede foi treinada com dois modelos de *dataset*, o treinamento de ambas durou dez mil iterações, levando aproximadamente dez segundos cada iteração.

o *Modelo 1* consiste em 3 mil imagens de latas, mais 3 mil imagens de garrafas totalizando 6 mil imagens. Já o *Modelo 2* consiste em 3 mil imagens de latas, mais 3 mil imagens de garrafas, e mais as 300 imagens denominadas falsos positivos totalizando 6300 imagens.

Etapa 6: Na fase de teste, foi possível notar a deficiência da rede na detecção de objetos

fisicamente próximos porém de classes diferentes.

Etapa 7: Como ficou perceptível a deficiência da rede em detectar objetos de classes diferentes na mesma imagem, foram inseridas mais 400 imagens que contêm ambas as classes, ou seja, cada imagem têm pelo menos uma lata e uma garrafa. Os dois modelos de *dataset* receberam essas imagens, totalizando 6400 imagens para o *Modelo 1*, e 6700 imagens para o *Modelo 2*.

Como a rede ainda estava em processo de convergência, cada uma iniciou de onde parou, treinando mais dez mil iterações cada, logo, o parâmetro que determina o número de iterações passou a ser: *max_batches = 20000*.

Etapa 8: Na segunda fase de teste foi possível notar três coisas, a primeira é que ambos os modelos sanaram a deficiência que tinham em encontrar classes diferentes na mesma imagem. A segunda constatação é que o *Modelo 2* localizou bem menos falsos positivos, principalmente em relação aos objetos citados nas seis classes da Etapa 1. E a terceira constatação é que há uma deficiência em encontrar objetos deitados.

Etapa 9: Como dito, foi possível perceber que a rede possui dificuldade em encontrar os objetos deitados/inclinados, porém, para resolver essa deficiência são logicamente necessário imagens com ângulos variados, o que não é tão simples de encontrar. Contudo, foram selecionadas mais 300 imagens para latas, e mais 300 imagens para garrafas, totalizando 600 imagens. Cada uma dessas imagens foram aleatoriamente rotacionadas para um dos seguintes ângulos: 90°, 180°, 270°, na tentativa de sanar esse problema.

A constatação de que o uso de imagens denominadas falso positivo estava melhorando o resultado da rede fez com que não houvesse a necessidade de continuar treinando o *Modelo 1*. Logo, foram adicionados mais 600 imagens ao *Modelo 2* totalizando 7300 imagens. E novamente a rede foi treinada a partir de onde parou, treinando mais dez mil iterações, o que resulta na alteração do parâmetro *max_batches* para 30000.

Etapa 10: Após a constatação dos resultados serem adequados, surgiu a necessidade de montar um banco de imagens apropriado para realizar testes de rotação, escala, iluminação, desfoque, mudança de ambiente, estado do objeto, quantidade de objetos e/ou classes, logo, foi montado um estúdio para simular essas condições.

Para a rotação foram considerados os eixos *x*, *y* e *z* conforme ilustrado na Figura 20.

Os resultados obtidos nessa fase serão submetidos a cálculos de *Precisão* e *Cobertura* que são dados extraídos de uma matriz chamada de *Matriz de Confusão*.

Uma Matriz de Confusão é um método de representação de dados que geralmente possui a estrutura ilustrada na Figura 21.

Em relação a Figura 21, *VP* é o número de classes reais preditas corretamente, já *FP* é o numero de classes reais não encontrada, *FN* é o número de vezes que a classe A foi dado

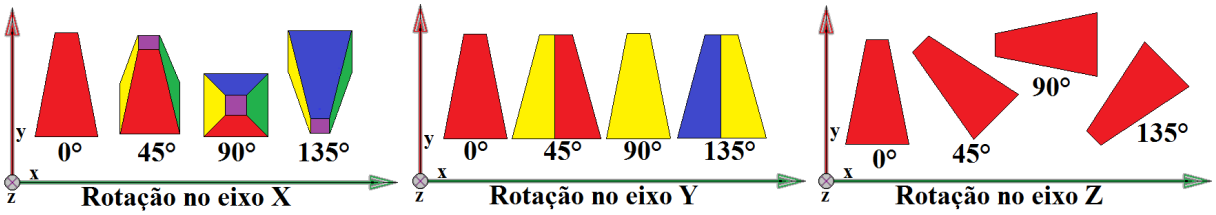


Figura 20 – Exemplo de rotação por eixo de objetos considerando um espaço tridimensional.

Fonte: Autoria Própria

		REAL	
		Classe A	Classe Ã
PREDITO	Classe A	VP	FP
	Classe Ã	FN	VN

Figura 21 – Matriz de Confusão para uma classe, onde considera a Classe A e o que não pertence a Classe A. As colunas representam o dado real e as linhas o que foi predito.

Fonte: Adaptado de (WITTEN et al., 2011)

a outro objeto, e por fim *VN* é o numero de objetos que não pertencem a Classe A e não foi predito como sendo desta.

Com a Matriz de confusão é possível calcular a *Precisão* da rede para saber se tudo o que foi recuperado é relevante, e a *Cobertura* para saber se tudo que é relevante foi recuperado. A Equação 22 representa a Precisão, e a Equação 23 representa a Cobertura (WITTEN et al., 2011).

$$Precisão = \frac{VP}{VP+FP} \tag{22}$$

$$Cobertura = \frac{VP}{VP+FN} \tag{23}$$

Por fim, existe uma equação conhecida como *Medida F* ou *Média Harmônica* representada pela Equação 24 que calcula uma média mais precisa com os valores obtidos na Precisão e Cobertura (WITTEN et al., 2011).

$$Média\ Harmônica = \frac{2\ x\ (Precisão\ x\ Cobertura)}{Precisão\ +\ Cobertura} \tag{24}$$

5 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os resultados para a conclusão do treinamento da rede de detecção de latas e garrafas.

A Figulra 22 ilustra a estrutura da rede usada no treinamento. Como pode ser visto, na primeira camada de convolução entra uma imagem RGB com dimensão de 416x416px e sai um mapa de características com dimensão de 416x416x32.

Camada	Filtro	Máscara / Stride	Entrada	Saída
0	conv 32	3 x 3 / 1	416 x 416 x 3	416 x 416 x 32
1	max	2 x 2 / 2	416 x 416 x 32	208 x 208 x 32
2	conv 64	3 x 3 / 1	208 x 208 x 32	208 x 208 x 64
3	max	2 x 2 / 2	208 x 208 x 64	104 x 104 x 64
4	conv 128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128
5	conv 64	1 x 1 / 1	104 x 104 x 128	104 x 104 x 64
6	conv 128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128
7	max	2 x 2 / 2	104 x 104 x 128	52 x 52 x 128
8	conv 256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256
9	conv 128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128
10	conv 256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256
11	max	2 x 2 / 2	52 x 52 x 256	26 x 26 x 256
12	conv 512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512
13	conv 256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256
14	conv 512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512
15	conv 256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256
16	conv 512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512
17	max	2 x 2 / 2	26 x 26 x 512	13 x 13 x 512
18	conv 1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024
19	conv 512	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 512
20	conv 1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024
21	conv 512	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 512
22	conv 1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024
23	conv 1024	3 x 3 / 1	13 x 13 x 1024	13 x 13 x 1024
24	conv 1024	3 x 3 / 1	13 x 13 x 1024	13 x 13 x 1024
25	route 16			
26	conv 64	1 x 1 / 1	26 x 26 x 512	26 x 26 x 64
27	reorg	/ 2	26 x 26 x 64	13 x 13 x 256
28	route 27 24			
29	conv 1024	3 x 3 / 1	13 x 13 x 1280	13 x 13 x 1024
30	conv 35	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 35

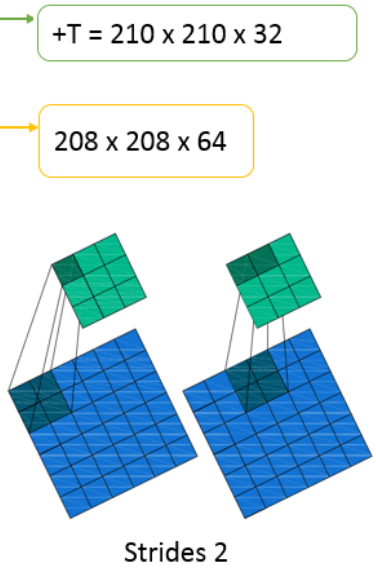


Figura 22 – Estrutura da Rede YOLOv2 Treinada no Darknet.

Fonte: Autoria Própria

A segunda camada é de *max-pooling* com *strides* de 2, logo a Figura chamada *Strides 2* é uma representação de como essa camada funciona, a mesma vai reduzir a dimensão da imagem na metade buscando o maior valor dentro da LRF de dimensão 2x2. Note que nas camadas de convoluções a dimensão do mapa é o mesmo na entrada e na saída, isso porque internamente a rede aplica o *transposed* e o *padding* para que a saída não perca dimensões durante a convolução.

A função do *reorg* é diminuir a dimensão dos mapas porém sem eliminar elementos, por isso é feito a multiplicação dos mapas por quatro, isso mostra que a rede só esta reorganizando suas informações pois ao multiplicar a entrada que é 26x26x64 obtêm-se a mesma quantidade de dados ao multiplicar o valor da saída 13x13x256.

Por fim, na última camada de convolução responsável pela classificação a entrada é de 13x13x1024 e a saída é de 13x13x35. Ao multiplicar os valores da última camada de aprendizagem (13x13x1024) resulta em 173056, no qual totaliza o número de pixels da imagem de entrada (416x416). Essa ocorrência sempre será verdade para valores de dimensões múltiplos de 32.

A rede final foi treinada com um total de 7300 imagens, sendo 3300 com latas, 3300 com garrafas, 400 com ambas, e 300 sendo falsos positivo. As 30 mil iterações resultaram em uma média de precisão *AP = 89.45%* para latas, e *AP = 85.65%* para garrafas, totalizando uma média final da média de precisão mAP = 87,55%, a um custo de aproximadamente 84 horas para atingir esse resultado. Porém os valores mais interessantes de se analisar são referentes a Precisão, Cobertura, Média Harmônica e média IoU, nas quais resultaram em 0.86, 0.89, 0.87 e 75.04% respectivamente, os valores da Matriz de confusão são de VP = 2652, FP = 430, e FN = 334, dados estes obtidos através do *framework Darknet*.

Foram obtidas 70 fotos no estúdio, cada foto foi redimensionada para o tamanho de 1200x900px. A soma de todos os objetos em cada imagem resulta em 154 latas, 156 garrafas, 28 possíveis falsos positivos. Esses objetos foram trabalhados em relação a rotação, escala, iluminação, desfoque, oclusão, mudança de ambiente, estado integro do objeto. Todas as fotos possuem mais de um objeto, seja ele pertencente a uma classe ou não. O estado integro do objeto refere-se somente as latas, onde quer dizer que algumas latas foram amassadas para analisar se a rede consegue identifica-las. A Tabela 3 exibe as características dos objetos encontrados pela rede, vale lembrar que um objeto pode ser submetido a mais de uma alteração, como por exemplo rotação mais oclusão, logo a soma do total não necessariamente será a mesma do total de objetos reais.

Tanto para a Tabela 3 quanto para a Tabela 4 foram trabalhadas imagens com iluminação normal, baixa iluminação e escala de cinza no ambiente controlado. Já para outros

ambientes foram testados 11 fundos ao total, onde variavam entre grama, piso, tijolos, madeira, concreto, pedra, céu, pano, e água, porém a iluminação e padrão de cor não foram trabalhados nos novos ambientes. Já as rotações foram testadas os três eixos (*x, y, z*) conforme descrito na Figura 20 no Capítulo 4. As rotações aplicadas nesses eixos são as descritas nas tabelas. Vale mencionar que alguns objetos foram submetidos a rotações em mais de um eixo.

Tabela 3 – Característica dos objetos Encontrados

		Ambiente Controlado			Novo Ambiente	**Total**
		Normal	Baixa Iluminação	Escala Cinza	Normal	
Lata	Normal	34	4	20	9	**67**
	Empilhamento	4	2	0	0	**6**
	Rotação Z 45°	2	1	0	3	**13**
	Rotação Z 90°	2	0	4	1	
	Rotação X 45°	0	0	0	2	**12**
	Rotação X 90° ou 270°	5	2	0	1	
	Rotação X 135° ou 225°	2	0	0	0	
	Rotação Y 45° ou 315°	2	0	0	2	**12**
	Rotação Y 90° ou 270°	3	0	0	3	
	Rotação Y 135° ou 225°	0	0	0	2	
	Escala	7	1	0	0	**8**
	Desfoque	7	1	0	0	**8**
	Oclusão	16	2	2	2	**22**
	Objeto Alterado	11	0	4	0	**15**
Garrafa	Normal	30	14	14	9	**67**
	Empilhamento	4	2	0	0	**6**
	Rotação Z 45°	1	0	0	1	**24**
	Rotação Z 90°	11	0	11	0	
	Rotação X 45°	0	0	0	2	**6**
	Rotação X 90° ou 270°	1	0	0	0	
	Rotação X 135° ou 225°	0	1	0	0	
	Rotação X 180°	2	0	0	0	
	Rotação Y 45° ou 315°	0	0	1	3	**4**
	Escala	7	2	0	0	**9**
	Desfoque	7	2	0	0	**9**
	Oclusão	14	6	5	5	**30**
Total		**172**	**40**	**61**	**45**	

Fonte: Autoria Própria

Sabendo que a Tabela 3 apresenta os objetos encontrados, e a Tabela 4 os objetos não encontrados, é possível tirar algumas informações em relação ao desempenho da rede. Dos 60 objetos oclusos somente 8 não foram encontrados o que representa 11.76% do total. Mas analisando os demais dados, fica perceptível a deficiência da rede em encontrar objetos em

Tabela 4 – Características dos Objetos Não Encontrados

		Ambiente Controlado			Novo Ambiente	**Total**
		Normal	Baixa Iluminação	Escala Cinza	Normal	
Lata	Normal	0	0	0	1	**1**
	Rotação Z 45°	1	0	0	0	**1**
	Rotação X 45° ou 315°	0	1	1	0	**4**
	Rotação X 90° ou 270°	0	1	1	0	
	Rotação Y 45° ou 315°	0	1	0	0	**3**
	Rotação Y 90° ou 270°	0	0	1	0	
	Rotação Y 135° ou 225°	0	0	1	0	
	Escala	0	1	0	0	**1**
	Desfoque	0	1	0	0	**1**
	Oclusão	1	2	0	0	**3**
	Objeto Alterado	0	0	5	0	**5**
Garrafa	Rotação Z 90°	0	0	1	0	**1**
	Rotação X 45° ou 315°	0	0	0	1	**8**
	Rotação X 90° ou 270°	3	0	2	2	
	Rotação Y 45° ou 315°	1	0	0	1	**6**
	Rotação Y 90° ou 270°	3	0	1	0	
	Oclusão	2	0	1	2	**5**
Total		**11**	**7**	**14**	**7**	

Fonte: Autoria Própria

determinados ângulos de determinados eixos conforme apresentado na Tabela 5.

Tabela 5 – Relação de objetos não encontrados de acordo com o tipo de Rotação

		Encontrados	Não Encontrados	Total	**% Total Não-Encontrado**
Lata	Rotação Z	13	1	14	**7,14%**
	Rotação X	12	4	16	**25,00%**
	Rotação Y	12	3	15	**20,00%**
Garrafa	Rotação Z	24	1	25	**4,00%**
	Rotação X	6	8	14	**57,14%**
	Rotação Y	4	6	10	**60,00%**

Fonte: Autoria Própria

Como pode ser visto na Tabela 5 o maior problema da rede está em encontrar garrafas com rotação nos eixos *X* e *Y* a uma taxa de 57,14% e 60,00% respectivamente.

Já com relação a variação de ambiente e cor da imagem, a taxa de acerto em objetos obtidos em um ambiente controlado foi melhor que as demais variações testadas, porém a média de acerto se manteve acima de 81% para todas as variações conforme a Tabela 6, ou seja,

a variação de cor da imagem ou iluminação sobre o objeto ou até mesmo de ambiente não interfere significativamente na detecção do objeto.

Tabela 6 – Relação de objetos não encontrados de acordo com a variação de ambiente

		Encontrados	Não Encontrados	Total	**% Total Não-Encontrado**
Ambiente Controlado	Normal	172	11	183	**6,01%**
	Baixa Iluminação	40	7	47	**14,89%**
	Escala Cinza	61	14	75	**18,67%**
Novo Ambiente	Normal	45	7	52	**13,46%**

Fonte: Autoria Própria

Com os resultados obtidos em cada imagem, foi possível montar a Matriz de Confusão apresentada na Tabela 7.

Tabela 7 – Matriz de Confusão

		Real	
		Lata	Outro
Preditos	Lata	143	8
	Outro	11	23(+153)

		Real	
		Garrafa	Outro
Preditos	Garrafa	142	2
	Outro	14	27(+153)

Na Tabela 7 consta que, 143 latas e 142 garrafas foram classificadas corretamente, 11 latas e 14 garrafas não foram encontradas pela rede, 8 latas e 2 garrafas foram encontradas sem existir, e dos 28 possíveis falsos positivos 5 foram classificados como latas, e apenas 1 como garrafa. Vale ressaltar que as classes estão sendo analisadas individualmente, logo, o número total de falsos positivos (184) para a classe lata é dado pela soma do total de garrafas (156) mais a soma do total de possíveis falsos positivos (28), o mesmo vale para a classe garrafa onde a soma do total de latas (154) mais a soma do total de possíveis falsos positivos (28) resulta no total de falsos positivos (182).

Para entender melhor o que os resultados da Matriz de Confusão representado pela Tabela 7 sugerem, serão calculados Precisão e Cobertura, para obter um resultado mais preciso calculando a Média Harmônica.

$$Lata: \ Precisão \ \frac{143}{143+8} = 0.9470 \quad Cobertura \ \frac{143}{143+11} = 0.9285 \tag{25}$$

$$Lata: \ Média\ Harmônica \ \frac{2\ x\ 0.9470\ x\ 0.9285}{0.9470\ +\ 0.9285} = 0.9376 \tag{26}$$

O cálculo de Precisão e Cobertura apresentados na Fórmula 25 para lata resultou em

94,7% e 92,85% respectivamente, logo a média harmônica apresentada na Fórmula 26 resulta em 93,76%, isso significa que para a classe latas no banco de imagens criado, a taxa de acerto está um pouco melhor que a taxa média da rede.

$$Garrafa: \ Precisão \ \frac{142}{142+2} = 0.9861 \quad Cobertura \ \frac{142}{142+14} = 0.9102 \tag{27}$$

$$Garrafa: \quad Média\ Harmônica \ \frac{2\ x\ 0.9861\ x\ 0.9102}{0.9861\ +\ 0.9102} = 0.9466 \tag{28}$$

Já o cálculo de Precisão e Cobertura apresentado na Fórmula 27 para garrafa resultou em 98,61% e 91,02% respectivamente, logo a média harmônica calculada na Fórmula 28 resulta em 94,66% o que demonstra que para a classe garrafa no banco de imagens criado para teste, a taxa de acerto é melhor para garrafas se considerar que encontra menos falsos positivos em comparação com o identificador de latas.

Para obter um resultado mais preciso será calculado também a Matriz de Confusão Espelhada representada pela Tabela 8. O cálculo dessa matriz sugere inverter a lógica, logo, agora serão analisados os objetos que não foram encontrados e que por sua natureza não deveriam ter sido encontrados.

Tabela 8 – Matriz de Confusão Espelhada

		Real	
		Outro	Lata
Preditos	Outro	176	11
	Lata	8	143

		Real	
		Outro	Garrafa
Preditos	Outro	180	14
	Garrafa	2	142

$$Lata\ Negativa: \ Precisão \ \frac{176}{176+11} = 0.9411 \quad Cobertura \ \frac{176}{176+8} = 0.9565 \tag{29}$$

$$Lata\ Negativa: \ Média\ Harmônica \ \frac{2\ x\ 0.9411\ x\ 0.9565}{0.9411\ +\ 0.9565} = 0.9487 \tag{30}$$

Os cálculos de Precisão e Cobertura apresentados na Fórmula 29 para outros objetos não sendo latas resultou em 94,11% e 95,65% respectivamente, o que resulta em uma média harmônica de 94,87% conforme a Fórmula 30. Essa afirmação sugere que a rede possui uma perda de eficiência com latas a uma pequena taxa de 5,13%.

$$Garrafa\ Negativa: \ Precisão \ \frac{180}{180+14} = 0.9278 \quad Cobertura \ \frac{180}{180+2} = 0.9890 \tag{31}$$

$$Garrafa\ Negativa: \quad Média\ Harmônica\ \frac{2\ x\ 0.9278\ x\ 0.9890}{0.9278\ +\ 0.9890}\ =\ 0.9574 \qquad (32)$$

Já com relação ao objetos não sendo garrafas, a Precisão e Cobertura apresentados na Fórmula 31 resultou em 92,78% e 98,90% respectivamente, o que resulta em uma média harmônica de 95,74% conforme a Fórmula 30. Isso sugere que a rede possui uma perda de eficiência com garrafas a uma taxa de 4,25% sendo uma ocorrência um pouco menor com relação as latas.

Por fim ao analisar o resultado nas imagens em si, foi possível notar alguns padrões que serão abordados a seguir.

Primeiramente a classe lata, onde há 8 falsos positivos para latas, 4 são latas porém não de bebida e sim de inseticida, 1 objeto foi classificado como lata e garrafa, sendo que é uma garrafa. Já 2 sugerem que a proporção do rótulo em relação a garrafa induz a rede a encontrar uma lata após ter encontrado uma garrafa na mesma região. E por fim o último falso positivo, apesar da rede ter sugerido lata para uma garrafa pet, vale ressaltar que o objeto está com pouca iluminação e a rede diz que há uma chance de somente 39,8% de ser esse objeto.

Com relação as latas de inseticidas uma forma de resolver é treinar a rede utilizando essas latas como sendo parte dos falsos positivos, porém essa solução induz a pensar que todo objeto semelhante deve entrar na classe de falsos positivos? e se for isso, quantos objetos existem no mundo?, o melhor a se fazer é analisar o contexto, geralmente objetos como latas e garrafas são encontrados próximos de copos, garrafas plásticas, barris, alimentos, pessoas, molho de chaves, celulares, cadeiras, mesas, e destes, copos e garrafas plásticas possuem características similares aos do objeto a ser encontrado, logo é interessante tomar esses objetos como classes negativas, mas sim, sempre há casos isolados onde na cena pode haver um objeto que será classificado mesmo não sendo o objeto de busca, como foi o caso do inseticida, mas o custo dessa ocorrência em relação ao geral é aceitável, pois uma rede bem treinada nem sempre vai apresentar esses erros.

Com Relação a classe garrafa, 10 das 14 garrafas não encontradas estão sofrendo variação na rotação. Existem latas que foram encontradas que sofrem o mesmo tipo de angulação das garrafas que não foram encontradas, o que sugere que uma ampliação no banco de treinamento englobando as rotações no eixo X e Y para garrafas pode resolver.

6 CONCLUSÕES

Devido a dimensão que uma rede pode alcançar em relação ao número de nós e camadas profundas o uso das placas de vídeo se torna indispensável. Além da placa de vídeo, também é necessário o uso da tecnologia CUDA que paraleliza os cálculos entre os núcleos da placa influenciando no tempo de treinamento. O CUDA não é a única ferramenta que faz isso, porém é bem desenvolvida e por isso é um recurso explorado nos *frameworks* mais conhecidos.

O uso do modelo de rede YOLOv2 proporcionou de forma simples o treinamento dos dois objetos propostos (latas e garrafas). A arquitetura da rede torna flexível para treinar desde uma única classe até 80 classes variando somente o número de iterações para atingirem a mesma acurácia, o que a torna viável não somente aos objetos propostos nesse trabalho, mas também a qualquer outro.

É possível constatar que mexer na estrutura de uma rede CNN é um processo delicado que exige conhecimento mais aprimorado, entretanto ainda é possível compreender a rede através dos resultados, e com isso buscar soluções conhecidas, seja aumentando o volume de imagens para treinamento, ou deixando a rede treinar mais tempo, ou mudando a proporção de treinamento e teste, ou ainda alterando os parâmetros conhecidos.

O problema de detecção de latas e garrafas em imagens demonstrou resultados satisfatórios considerando o desempenho e o tempo, já que as taxas de acerto sobre as classes treinadas atingiram uma média de 87%, e a precisão de localização do objeto atingiu uma taxa de 75,04% a um custo de tempo de aproximadamente quatro dias de treinamento. E isso torna as redes neurais artificiais, principalmente as convolucionais quando se trata de imagens, uma opção a ser considerada quando surge problemas de detecção e classificação de objetos.

Uma sugestão de trabalho futuro é treinar a rede a partir desse modelo com o objetivo de classificar os objetos, pois a rede encontra latas e garrafas, a classificação indicaria se essa lata/garrafa é de refrigerante, cerveja, suco, etc, ou ainda identificar a marca. Com a classificação certamente a precisão da rede melhora, pois a generalização reduz e a rede passa a buscar detalhes mais específicos nos objetos.

REFERÊNCIAS

ABDEL-QADER, I.; ABUDAYYEH, O.; KELLY, M. E. Analysis of edge-detection techniques for crack identification in bridges. **Journal of Computing in Civil Engineering**, v. 17, n. 4, p. 255–263, 2003. Disponível em: <http://dx.doi.org/10.1061/(ASCE)0887-3801(2003)17:4(255)>.

ADI, K.; PUJIYANTO, S.; NURHAYATI, O.; PAMUNGKAS, A. Beef quality identification using color analysis and k-nearest neighbor classification. In: . [s.n.], 2015. p. 180–184. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84963973930&partnerID=40&md5=3d9b2d9b154a61430cec3ebb03924426>.

AKINLOLU, A. Facial biometrics of yorubas of nigeria using akinlolu-raji image-processing algorithm. **Journal of Medical Sciences (Taiwan)**, v. 36, n. 2, p. 39–45, 2016. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964916915&partnerID=40&md5=7b4b55824a54f28ce0bf69ae0b50e84a>.

AREL, I.; ROSE, D.; KARNOWSKI, T. Deep machine learning-a new frontier in artificial intelligence research. **IEEE Computational Intelligence Magazine**, v. 5, n. 4, p. 13–18, 2010. Cited By 154. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-77958488310&partnerID=40&md5=621fd1749901fd579e898db77d159e18>.

BEALE, R.; JACKSON, T. **Neural Computing: An Introduction**. [S.l.]: © IOP Publishing Ltd, 1990.

BENALIA, S.; CUBERO, S.; PRATS-MONTALBáN, J. M.; BERNARDI, B.; ZIMBALATTI, G.; BLASCO, J. Computer vision for automatic quality inspection of dried figs (ficus carica l.) in real-time. **Computers and Electronics in Agriculture**, v. 120, p. 17 – 25, 2016. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169915003397>.

BIDHULT, S.; XANTHIS, C.; LILJEKVIST, L.; GREIL, G.; NAGEL, E.; ALETRAS, A.; HEIBERG, E.; HEDSTRöM, E. Validation of a new t2∗ algorithm and its uncertainty value for cardiac and liver iron load determination from mri magnitude images. **Magnetic Resonance in Medicine**, v. 75, n. 4, p. 1717–1729, 2016. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84930029502&partnerID=40&md5=1b372ffe4feedfb9045a4c6e72733dd4>.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDERMIR, T. B. **Redes Neurais Artificiais: Teoria e Aplicações**. [S.l.]: LTC - Livros técnicos e científicos editora S.A., 2000.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **ArXiv e-prints**, mar. 2016.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. de. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. [S.l.]: LTC - Livros Técnicos e Científicos Editora Ltda, 2011.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. [S.l.]: Prentice Hall, 2008.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. Book in preparation for MIT Press. 2016. Disponível em: <http://www.deeplearningbook.org>.

GUPTA, D.; ANAND, R. A hybrid edge-based segmentation approach for ultrasound medical images. **Biomedical Signal Processing and Control**, v. 31, p. 116–126, 2017. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84979941598&partnerID=40&md5=9b0a322697695e4e2322c6ceacc86dce>.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2nd. ed. Porto Alegre: Bookman, 2001.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

KOVACS, Z. L. **Redes Neurais Artificiais: Fundamentos e Aplicações**. 4nd. ed. [S.l.]: Editora Livraria da Física, 2006.

LIMA, I.; PINHEIRO, C. **Inteligência artificial**. Elsevier Brasil, 2014. ISBN 9788535278095. Disponível em: <https://books.google.com.br/books?id=qjJeBgAAQBAJ>.

MADABHUSHI, A.; LEE, G. Image analysis and machine learning in digital pathology: Challenges and opportunities. **Medical Image Analysis**, v. 33, p. 170 – 175, 2016. ISSN 1361-8415. 20th anniversary of the Medical Image Analysis journal (MedIA). Disponível em: <http://www.sciencedirect.com/science/article/pii/S1361841516301141>.

MIRANDA, J. I. **Processamento de Imagens Digitais: Prática Usando Javatexttrademark**. [S.l.]: Embrapa, 2006.

NAJAFABADI, M. M.; VILLANUSTRE, F.; KHOSHGOFTAAR, T. M.; SELIYA, N.; WALD, R.; MUHAREMAGIC, E. Deep learning applications and challenges in big data analytics. p. 1–21, 2015.

NIELSEN, M. A. Neural networks and deep learning. 2015. Disponível em: <http://neuralnetworksanddeeplearning.com>.

PATIL, R. V.; PETE, D. J. Image change detection using stereo imagery and digital surface mode. In: **2015 International Conference on Information Processing (ICIP)**. [S.l.: s.n.], 2015. p. 192–197.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de Imagens Digitais : Princípios, Algoritmos e Aplicações**. [S.l.]: Thomson Learning, 2008.

PITERI, M. A.; RODRIGUES, J. C. **Fundamentos de Visão Computacional**. [S.l.]: Presidente Prudente, 2011.

REDMON, J. **Darknet: Open Source Neural Networks in C**. 2013–2016. `http://pjreddie.com/darknet/`.

REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Disponível em: <http://arxiv.org/abs/1506.02640>.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. **arXiv preprint arXiv:1612.08242**, 2016.

RUSSELL, I. F. Neural networks in the undergraduate curriculum. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 6, n. 5, p. 92–97, abr. 1991. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=128000.904009>.

SILVA, R. M. **Redes Neurais Artificiais aplicadas à Detecção de Intrusão em Redes TCP / IP**. 144 p. Tese (Doutorado), 2005.

SITARAM, D.; PADMANABHA, N.; S, S.; S, S. Still image processing techniques for intelligent traffic monitoring. In: **2015 Third International Conference on Image Information Processing (ICIIP)**. [S.l.: s.n.], 2015. p. 252–255.

SOLOMON, C.; BRECKON, T. **Fundamentos de processamento digital de imagens**. [S.l.]: Ltc, 2013.

VALIATI, G. **BBox-Label-Tool**. 2018. `https://github.com/gustavovaliati/BBox-Label-Tool`.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining Practical Machine Learning Tools and Techniques Third Edition**. [S.l.]: ELSEVIER, 2011.

YADAV, N.; YADAV, A.; KUMAR, M. **An Introduction to Neural Network Methods for Differential Equations**. [s.n.], 2015. 13–15 p. ISSN 21915318. ISBN 978-94-017-9815-0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84924190922\&partnerID=tZOtx>.