UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM SISTEMAS EMBARCADOS
PARA A INDUSTRIA AUTOMOTIVA

RODRIGO MEIRA DE ANDRADE

MÓDULO AUXILIAR DE MONITORAMENTO VEICULAR ATRAVÉS DE CÂMERAS E DEEP LEARNING

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

RODRIGO MEIRA DE ANDRADE

MÓDULO AUXILIAR DE MONITORAMENTO VEICULAR ATRAVÉS DE CÂMERAS E DEEP LEARNING

Monografia de Especialização, apresentado ao Curso de Especialização em Sistemas Embarcados para a Industria Automotiva, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Max Mauro Dias Santos

CURITIBA
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Eletrônica
Curso de Especialização em Sistemas Embarcados para Indústria Automotiva

TERMO DE APROVAÇÃO

MÓDULO AUXILIAR DE MONITORAMENTO VEICULAR ATRAVÉS DE CÂMERAS E DEEP LEARNING

por

RODRIGO MEIRA DE ANDRADE

Esta Monografia foi apresentada em 11 de dezembro de 20178 como requisito arcial para a obtenção do título de Especialista em Sistemas Embarcados para a Indústria Automotiva. A candidata foi arguida pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Max Mauro Dias Santos
Orientador

_____
Prof. Dr. Edenilson José da Silva
Membro titular

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
Coordenador do Curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

O número de mortes de trânsito no Brasil vem aumentando gradativamente a cada ano, uma parte desse aumento é devido ao desrespeito e a distração do condutor no trânsito. Atualmente centros urbanos utilizam radares para o monitoramento de velocidade dos veículos, e os dados estatísticos provam que o uso desses equipamentos diminuiu drasticamente a quantidade de acidentes e consequentemente a quantidade de mortos, hoje com a tecnologia atual já é possível empregar radares diretamente nos veículos, porém pouco se houve a respeito da existência deles, o uso da tecnologia atual para o desenvolvimento desses radares podem servir como um auxílio ao motorista, e não um equipamento para penalizar, com o uso de câmeras no veículo pode-se criar um equipamento que auxilie o motorista, transformando informações visuais capturadas pelas câmeras em informações sonoras, enviadas para o motorista, com técnicas de aprendizado de máquina, mais especificamente na área de deep learning, já é possível desenvolver algoritmos que possa identificar determinada ação do motorista pelo seus gestos e assim executar outra ação para auxilia-lo, o estudo a ser apresentado tem como objetivo o desenvolvimento de um algoritmo que através de câmeras consiga detectar a direção para onde o motorista está olhando e assim com câmeras distribuídas pelo veículo possa monitorar automaticamente regiões em que o motorista não está observando, e caso haja determinada situação de risco, esse algoritmo deverá emitir uma notificação sonora para o condutor.

Palavras chave: Veículo. Monitoramento. Imagem. Câmeras. Deep Learning

ABSTRACT

MEIRA DE ANDRADE, Rodrigo. **AUXILIARY VEHICLE MONITORING MODULE THROUGH CAMERAS AND DEEP LEARNING**. 2017. 37 f. Monografia (Curso de Especialização em Automação Industrial), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

The number of traffic fatalities in Brazil is increasing gradually each year, a part of this increase is due to the disrespect and the distraction of the driver in traffic. Currently, urban centers use radars to monitor vehicle speed, and statistical data prove that the use of such equipment has dramatically reduced the number of accidents and consequently the number of deaths, today with the current technology it is already possible to employ radars directly in vehicles, but little if there were of them, the use of the current technology for the development of these radars can serve as an aid to the driver, not an equipment to penalize, with the use of cameras in the vehicle can create equipment that helps the driver , transforming visual information captured by the cameras into sound information, sent to the driver, with machine learning techniques, more specifically in the area of deep learning, it is already possible to develop algorithms that can identify a certain action of the driver by his gestures and thus execute another action to help him, the study to be presented aim is to develop an algorithm that through cameras can detect the direction to which the driver is looking and thus with cameras distributed by the vehicle can automatically monitor regions in which the driver is not observing and in case of a certain risk situation, this algorithm shall issue a sound notification to the driver.

Keywords: Vehicle. Monitoring. Image. Cameras. Deep Learning

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Conforme facilmente se constata através dos veículos de informação, o número de mortos e vítimas de trânsito apresentam crescimento gradativo a cada ano. Dentre as principais razões dos acidentes tem origem na falta de atenção por parte dos motoristas, associada a distração, sono e desobediência. Nesse sentido, uma solução viável para que tais situações sejam evitadas pode se dar através do uso de câmeras no veículo, permitindo a criação de mecanismo de assessoramento ao motorista, no qual informações visuais capturadas pelas câmeras são convertidas em informações sonoras a fim de emitir um alerta ao condutor quando diante de uma situação de risco.

Através de métodos de aprendizado de máquina, notadamente na área de deep learning, mostra-se possível a criação de algoritmos capazes de discernir certa ação do condutor estabelecida por meio gestos e de sua própria fisionomia, a fim de que, uma vez detectado o ato imprudente, seja possível emitir uma resposta sonora como um suporte ao condutor.

1.1 PROBLEMA

Os radares utilizados atualmente nos centros urbanos tem a finalidade de promover a fiscalização da velocidade dos veículos, sendo que essa ferramenta tem se mostrado eficiente quando constatada a diminuição significativa da quantidade de acidentes, implicando por consequência na redução das vítimas de trânsito. Através da tecnologia atual, mostra-se possível utilizar radares diretamente nos veículos. Muito embora tal ferramenta seja pouco difundida, é de se notar que o uso da tecnologia atual para o desenvolvimento desses radares revela-se adequada para promover um instrumento de assistência ao condutor, e não apenas um equipamento para penalizar.

Desta feita, o propósito do presente estudo tem respaldo na elaboração de um algoritmo que, através de câmeras instaladas pelo veículo, identifique a direção para onde o condutor está olhando, a fim de que seja possível monitorar de forma automática setores que se encontram fora da área de percepção do motorista, sendo que, diante da ocorrência de uma situação de risco, o algoritmo emitirá uma notificação sonora para o condutor.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Esse trabalho tem como objetivo apresentar uma solução de visão computacional que possa vir a ser embarcada nos veículos para poder fazer monitoramentos de pontos cegos através da classificação da atitude do condutor.

1.2.2 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Identificar direção da visão do condutor
- Identificar áreas de risco
- Localizar e classificar pedestre nas áreas de risco
- Validar desempenho através de testes e simulações

1.3 JUSTIFICATIVA

A correria do dia a dia e a necessidade de se manter sempre conectado faz com que os motoristas fiquem cada vez mais distraídos no trânsito. De acordo com um levantamento do Cesvi/Brasil distração é uma das principais causas de acidentes. No mundo, pelo menos 3 mil pessoas morrem diariamente em ocorrências no trânsito, segundo dados das Organizações das Nações Unidas (ONU). No Brasil, o cenário também é preocupante. O país aparece em quinto lugar no ranking mundial de mortes no trânsito. Uma boa parcela deste indicador é resultado de práticas ruins de motoristas e pedestres.

19 pessoas são atropeladas por dia em São Paulo. Foram 7007 no ano de 2010, nas quais 630 morreram. Nesta matéria da revista Veja o autor diz a respeito dos pedestres: “No dia em que os motoristas aprenderem a respeitá-lo, teremos subido de patamar no quesito da pacífica e respeitosa convivência entre os habitantes da cidade“. (VEJASP, 2013)

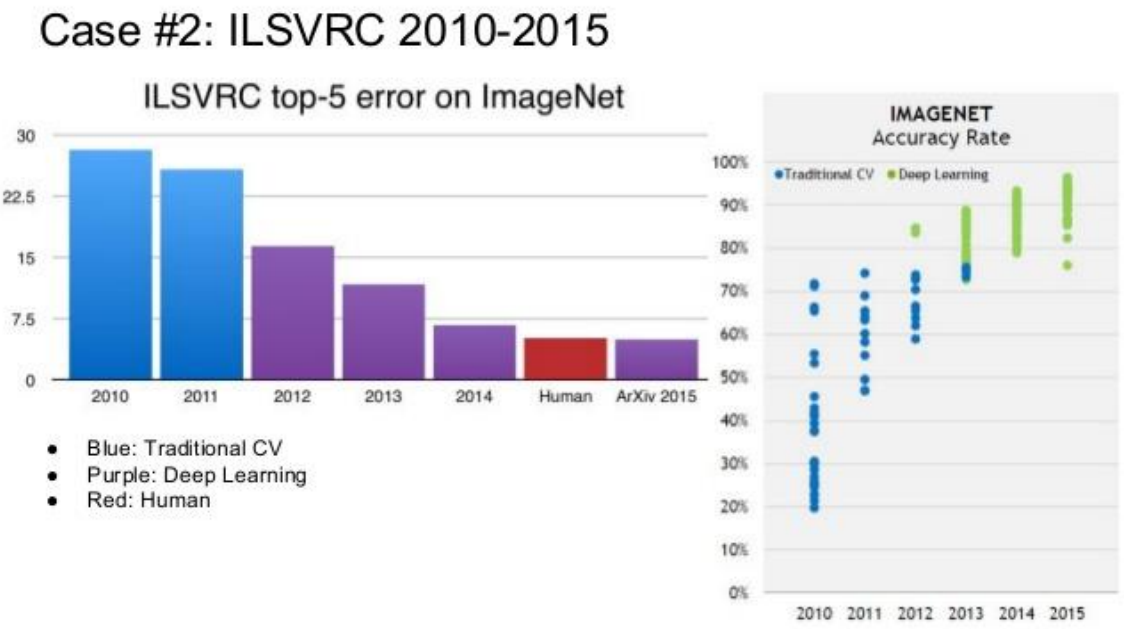
As câmeras em geral, conseguem produzir uma grande quantidade de informações através das imagens geradas por ela, atualmente os sensores utilizados pelas câmeras já possuem sensibilidade maior que o olho humano para alguns aspectos, como maior sensibilidade a luz infravermelha e maior sensibilidade a luminosidade são uma das características (STOLFI, 2008).

Desde 2010, anualmente ocorre uma competição (ILSVRC) para avaliar o desempenho dos algoritmos de visão computacional que vem sendo desenvolvidos, esta competição é feita pelo setor de visão computacional da universidade de Stanford, cujo um dos patrocinadores desta competição é o Google. Dentre os vários objetivos que os algoritmos devem cumprir nesta competição, um dos objetivos que ganhou fama é o “top-5 error”, neste objetivo o algoritmo deve classificar o objeto em uma imagem entre 1000 possíveis classes, o algoritmo pode dar como resultado as 5 classes com maior probabilidade que ele detectar, caso a classificação correta esteja entre uma destas 5 melhores predições, então é considerado acerto.

No início desta competição, os algoritmos utilizados eram majoritariamente baseados em técnicas de visão computacional para extrair informações das imagens, como histogramas, filtros de borda e gradientes. Até que em 2012 um algoritmo ganhou notoriedade, por ter um resultado aproximadamente de 85% de acerto, cerca de 10% acima dos resultados dos anos anteriores na categoria de “top-5 error”. Este algoritmo batizado de AlexNet mostrou novos horizontes a serem explorados, pois o AlexNet foi baseado em redes neurais convolucionais (CNN)(KRIZHEVSKY; HINTON, 2012). Basicamente, as redes neurais (ANN), em teoria, imitam o papel desempenhado por neurônios, através de modelos matemáticos baseados em estudos da medicina e biologia.(BERESFORD, 2000) Já as CNNs vão um pouco mais além, na topologia deste formato de rede existem determinados neurônios que desempenham papeis de filtros, assim como os neurônios do córtex cerebral, esses neurônios geram fortes sinapses ao serem excitados por determinadas características na imagem.(HUBEL; WIESEL, 1962)

Por fim, em 2015 um marco histórico foi alcançado nesta competição, uma CNN com o nome de ResNet obteve um erro de 3.57% (HE,), isto significa que um computador, pela primeira vez, teve desempenho superior ao um humano em classificar as imagens, já que o erro estimado para os humanos nesta tarefa é de aproximadamente 5%.

Figura 1 - Evolução de desempenho ao decorrer dos anos.



Fonte: www.image-net.org

A ILSVRC acaba sendo um indicador do avanço tecnológico na área de visão computacional, por conta da grande popularidade que esta competição ganhou. Com os dados fornecidos pela ILSVRC, é possível concluir que estes algoritmos podem auxiliar em grande peso o desenvolvimento de um software para um monitoramento autônomo da faixa de pedestre. A escolha do uso da câmera como principal sensor para esta tarefa, se dá por 4 características, que são elas:

- Baixo custo em comparação a outros sensores
- Fácil instalação
- Pode cobrir uma grande área para monitoramento
- O fator limitante de desempenho dificilmente é causado pela câmera.

1.4 ESTRUTURA DO TRABALHO

2 FUNDAMENTAÇÃO TEÓRICA

Na revisão da literatura foi feito um estudo das principais obras científicas relacionadas com conceitos que possam ser utilizados como ferramentas para que auxiliem o alcance do objetivo proposto. Para o alcance destes objetivos, a pesquisa foi dividida em 2 temas relacionados a visão computacional, nos quais juntos irão formar o alicerce do projeto desenvolvido. Estes temas são:

- Detecção e Classificação desses objetos.
- Rastreio de objetos entre frames de vídeos.

1.1 Classificação de Objetos

A classificação de imagem é a tarefa no qual um algoritmo possui uma imagem como entrada de dados e gera uma classe de saída baseada nas informações da imagem de entrada, esta classe de saída por ser tipos de objetos como carro, pedestre, avião, etc.

Para o computador uma imagem é uma matriz com 3 dimensões, cada posição desta matriz no eixo “x” e “y” refere-se ao valor de um pixel da imagem, para cada matriz alocada no eixo “z” refere-se a um canal de cor, comumente o eixo “z” possui 3 canais, cada canal refere-se as cores RGB, uma imagem com dimensões 640x480x3 significa que ela possui 640 pixels ao longo de seu eixo “x”, 480 pixels em seu eixo “y” e três canais de cores no eixo “z”, por padrão o valor de cor de cada pixel em cada canal, é um número inteiro de 8 bits, isto significa que este valor pode variar de 0 a 255, uma imagem RGB com uma resolução de 8 bits por canal contem 16.777.216 possíveis combinações de cores para cada pixel. Os valores destes pixels então são utilizados para extrair características da imagem parar gerar valores de entrada nos classificadores.

Dentro da área de aprendizado de máquina existem os métodos chamados classificadores, estes classificadores utilizam as variáveis independentes do problema proposto para reconhecer padrões e assim classificar a variável independente. Para que isto seja possível estes classificadores necessitam de uma porção de amostras, onde é conhecido os valores de entrada (variável independente) e os valores de saída (variável dependente) do problema, com estas amostras conhecidas, estes classificadores são ajustados a modo que com apenas utilizando as variáveis independente seja possível classificar a variável dependente.

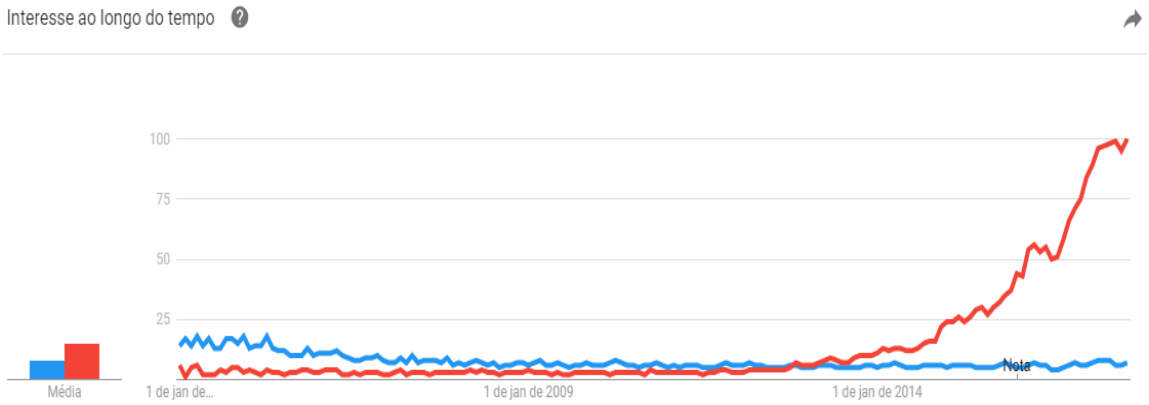
Os classificadores mais utilizados e conhecidos são:

- Knn

- Árvore de decisão
- *Random Forest*
- *Support Vector Machines (SVM)*
- *Xgboost*
- Redes neurais (ANN)
- Redes neurais convolutivas (CNN)
- Regressão logística
- Naive Bayes

Entre estes classificadores, dois deles possuem notoriedade por sua performance quando utilizados em imagens, estes classificadores são as SVM's e CNN's. Dentre estas duas, a SVM vem sendo explorada mais arduamente desde sua criação, contudo notou-se a diminuição dos estudos relacionados às SVM, e a partir do ano de 2012 os estudos relacionados a *deep learning,* no qual as CNN's fazem parte, vem crescendo exponencialmente.

Figura 2 - SVM (azul) vs. Deep Learning (vermelho)



Fonte: Google Trends

1.1.1 Deep Learning

Basicamente, as redes neurais (ANN), em teoria, imitam o papel desempenhado por neurônios, através de modelos matemáticos baseados em estudos da medicina e biologia.(BERESFORD, 2000) Já as CNNs vão um pouco mais além, na topologia deste formato de rede existem determinados neurônios que desempenham papeis de filtros, assim como os neurônios do córtex cerebral, esses neurônios geram fortes sinapses ao serem excitados por determinadas características na imagem.(HUBEL; WIESEL, 1962)

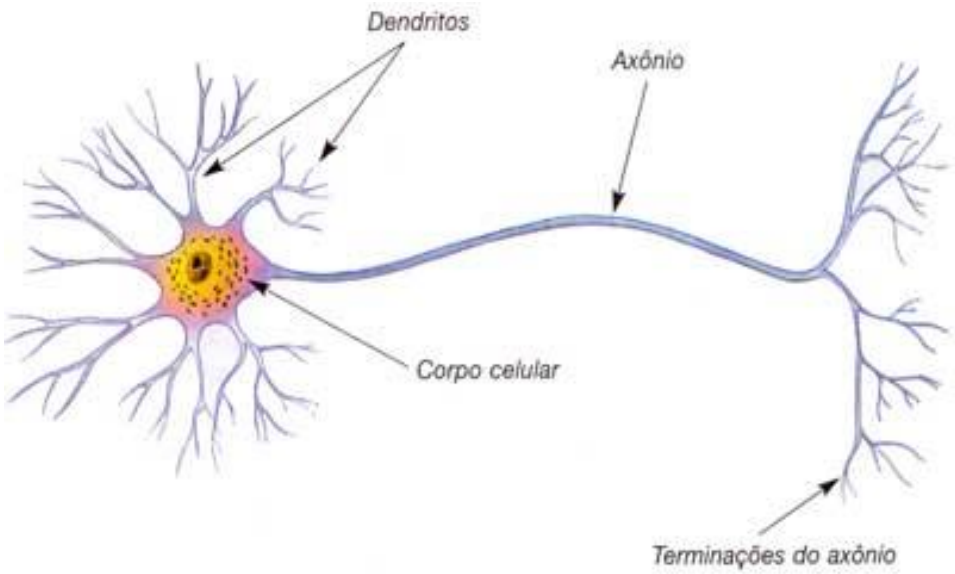
1.1.1.1 Redes Neurais

Os primeiros estudos relacionados as redes neurais foram feitos em 1943, neste estudo eles desenvolveram um método matemático com objetivo de descrever matematicamente o comportamento de um neurônico biológico, neste estudo mostrou-se que um neurônio utiliza as sinapses geradas pelos neurônios vizinhos para gerar sua própria sinapse e que a sinapse inibitória previne a excitação do neurônio naquele instante. (MCCULLOCH; PITTS, 1943) A primeira rede neural a obter sucesso em seu funcionamento surgiu na década de 50, esta rede tinha como objetivo o reconhecimento de padrões.(ROSENBLATT, 1957)

As redes neurais são modelos digitalizados que tem como inspiração o cérebro humano, estas redes tentam imitar a forma que o cérebro processa as informações, assim como o cérebro estas redes aprendem a reconhecer padrões através da experiencia adquirida ou então através de treinamentos.(BERESFORD, 2000)

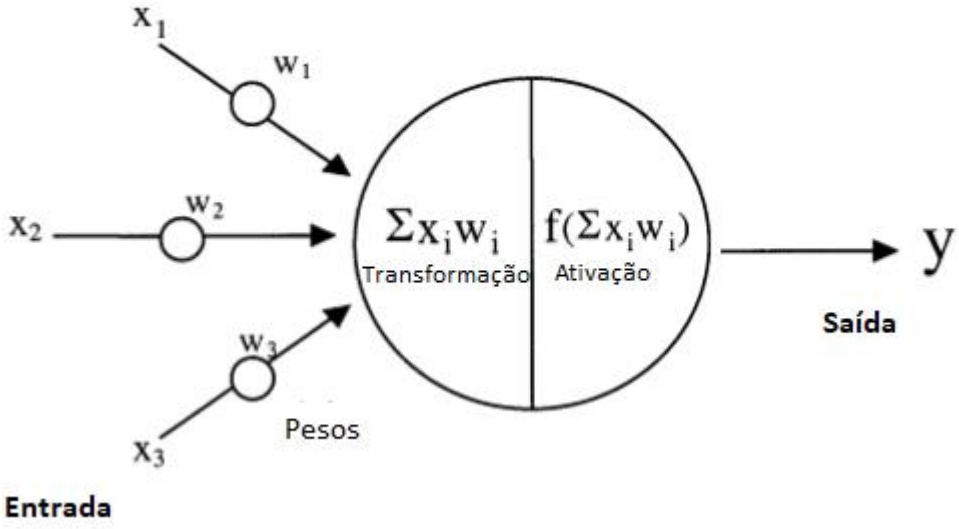
Um cérebro humano comum tem aproximadamente 100 bilhões de neurônios e cada um desses neurônios possuem entre 1000 a 10000 conexões com os neurônios vizinhos. O corpo do neurônio consiste em seu núcleo, Dendritos, Axônios e suas terminações nervosas. Os Dendritos são responsáveis por receber as sinapses de outros neurônios e carregar esta informação até o seu núcleo, o núcleo por sua vez processa esta sinapse e envia uma nova sinapse para outros neurônios através das terminações nervosas do Axônio.(BERESFORD, 2000)

Figura 3 – Inspiração biológica



Fonte: WEB

Figura 4 - Topologia matemática de um neurônio



Fonte: adaptado de (BERESFORD, 2000)

O modelo matemático que representa a atividade do núcleo ao processar as sinapses é dado pela formula:

$$y = f(\sum xi * w1) \qquad \textbf{(1)}$$

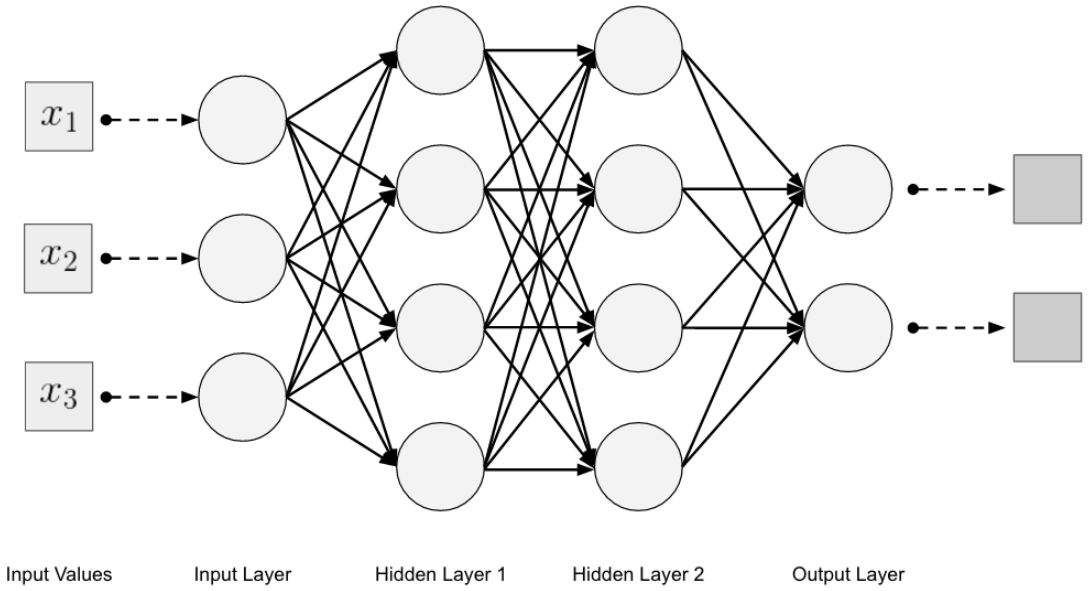
A variável “x” representa a sinapse de um neurônio vizinho “i”, essa sinapse possui um peso atrelado ao seu valor, isto representa o quão importante é esta sinapse para o neurônio que está processando, o valor deste peso é representado pela variável “w”. Com o valor do somatório destas sinapses e seus respectivos pesos é aplicada uma função conhecida como função de ativação, esta função pode variar entre vários modelos matemáticos e seu objetivo é gerar a sinapse do neurônio que será passada ao próximo neurônio.

Existem vários conceitos de topologias de redes neurais na literatura como a *Feedforward network* (TAHMASEBI; AMIRKABIR, 2011), *Feedback network* (ZAMARREN; GONZA, 2005)*, Recurrent neural network* (ROJAS, 1996) e *self-organizing maps.*(KOHONEN, 1990)*.*

O modelo de estudo de rede neural nesta revisão bibliográfica foi referente a *Feedforward network,* porque este modelo de rede é bastante utilizado nas redes neurais convolucionais, nas quais serão aplicadas para a classificação de objetos. (ROSENBLATT, 1957)

A rede *Feedforward* em sua topologia consiste em três camadas, a camada de entrada, as camadas ocultas e a camada de saída.

Figura 5 - Topologia de uma rede FeedFoward



Fonte: WEB

- Na camada de entrada desta rede a quantidade de neurônios utilizadas é igual a quantidade de variáveis independentes que é utilizada para classificar um problema, esta primeira camada fornece os dados para as camadas ocultas, neste modelo de rede, pode haver apenas uma ou várias camadas ocultas, a quantidade de neurônios utilizada nestas camadas geralmente é definida empiricamente.
- Na camada oculta é onde ocorre o processamento dos dados, após o treinamento da rede, os neurônios dessa camada geram determinadas sinapses ao detectar determinadas características provenientes da camada anterior ou da camada de entrada da rede, estas características são detectadas através da distribuição dos pesos entre os neurônios.
- Na camada de saída a quantidade de neurônios utilizada geralmente é a quantidade de classes em que a rede foi treinada para classificar, desse modo cada neurônio desta camada representa uma classe, e para cada neurônio da camada de saída, seu valor representa a probabilidade dos dados de entrada ser classificado como determinada classe

1.1.1.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (*Convolutional Neural Networks* – CNN) tem sido umas das inovações mais influentes no campo da visão computacional. Sua aplicação no campo de análise de imagens ganhou maior interesse a partir de 2012 devido ao seu desempenho alcançado na ILSVRC (KRIZHEVSKY; HINTON,

2012). Atualmente, grandes companhias do setor de tecnologia da informação vêm investindo fortemente nesta área. Empresas como o Facebook utiliza estes modelos de rede para seu mecanismo de localizar e marcar as pessoas nas fotos, o Google utiliza em seu mecanismo de busca por fotos, o Instagram assim como o Google também utiliza estes modelos de rede em seu mecanismo de busca, a Amazon utiliza no seu sistema de recomendação de produtos, o Pinterest criou a possibilidade de ter seu *feed* de imagens personalizado graças as CNNs.

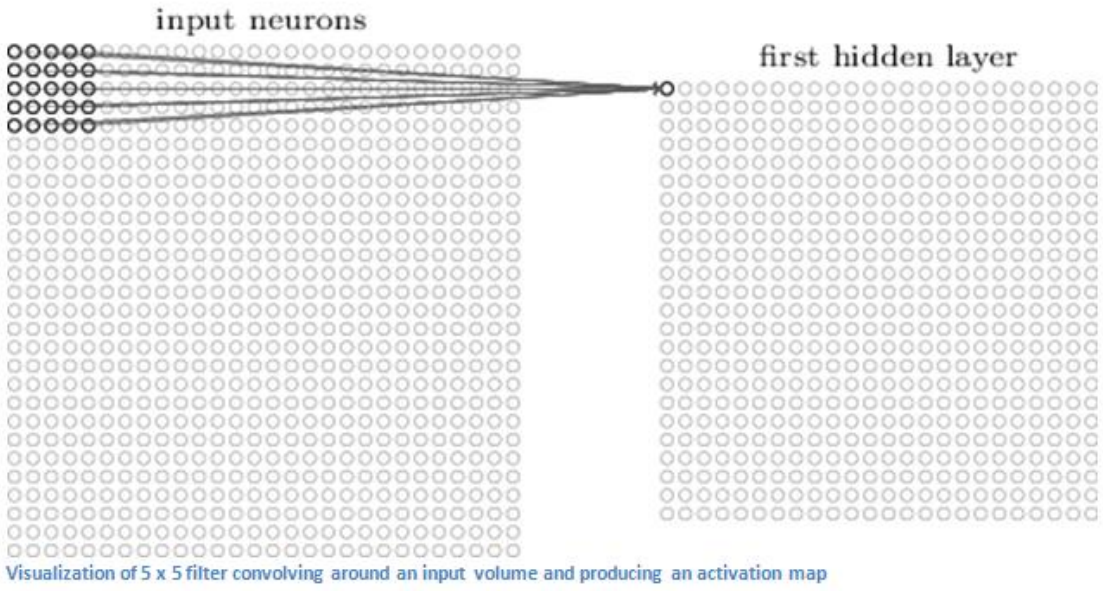
A inspiração das CNN's vem do córtex visual humano, neste córtex há uma pequena região com neurônios que são sensíveis as informações do nosso campo visual. Em 1962 Hubel e Wiesel realizaram um experimento onde provaram que em nosso córtex visual há neurônios que são excitados e geram fortes sinapses quando nossa visão é exposta a imagens contendo bordas orientadas em determinada direção. Neste experimento eles perceberam que estes neurônios formam uma arquitetura onde cada neurônio gera uma sinapse para determinada orientação de borda, juntos nesta arquitetura estes neurônios conseguem produzir a percepção visual. Isto demonstrou que cada neurônio possui determinada tarefa na busca de uma característica especifica, e que juntos com todas as características detectadas por estes neurônios é então formada nossa percepção visual. (HUBEL; WIESEL, 1962)

Baseados nestes fatos biológicos, as CNNs utilizam filtros (*kernels*) que são aplicados na imagem através de convoluções para a extração de características, assim replicando matematicamente o papel desempenhado pelos neurônios no córtex visual. A aplicação destas convoluções nas imagens geram novas imagens com determinadas características mais evidentes, e assim, sucessivamente estas imagens formam novas entradas das camadas mais profundas da rede, onde novamente são aplicadas convoluções com diferentes filtros. Desta forma as camadas mais superficiais da rede conseguem extrair características mais simples, como bordas verticais e horizontais, conforme a imagem vai passando para as camadas mais profunda da rede, características mais complexas são extraídas, como formas geométricas, até chegar em determinada camada em que características como rostos e objetos são extraídos, no geral quanto mais funda a camada da rede, mais abstrata é a característica extraída. As análises de todas estas características abstratas geram a classificação da imagem.

A primeira camada de uma CNN sempre será uma camada de convolução, nas camadas de convoluções os filtros utilizados geralmente são matrizes com pequenas dimensões (3x3,5x5,7x7, etc.), cada valor das posições desses filtros são chamados de pesos assim como nas ANN. A partir destas pequenas matrizes é feito um deslocamento desta matriz em relação aos dados de entrada da camada que são matrizes maiores onde representam a imagem de entrada. A resposta gerada pela convolução é produto de pequenas convoluções feitas através do filtro em relação a imagem durante seu deslocamento. O funcionamento de uma convolução ocorre de maneira simples, é feita um somatório das multiplicações ponto a ponto

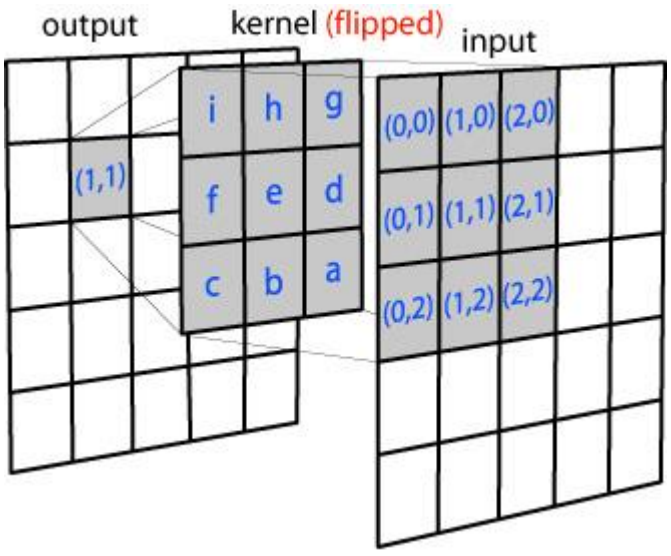
em relação ao filtro e parte da imagem onde este filtro está fazendo o deslocamento. A figura 9 e 10 ilustram de maneira bastante simples como é feito este deslocamento.

Figura 6 - Convolução de uma imagem (1)



Fonte: WEB

Figura 7 - Convolução de uma imagem (2)



Fonte: <http://www.songho.ca/dsp/convolution/convolution.html>

A fórmula de uma convolução discreta para matrizes 2D é:

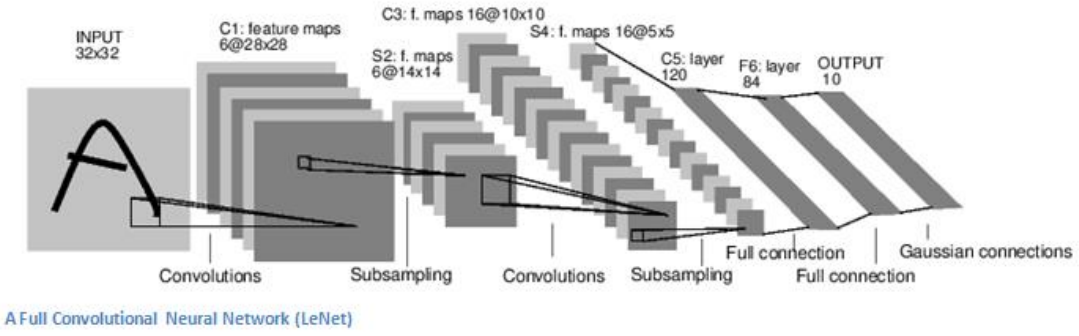
$$y(m,n) = x(m.n) * h(m,n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \times h[m-i, n-j] \quad \textbf{(2)}$$

Onde “y” é a matriz resposta, “x” a matriz kernel e “h” é a matriz de entrada, as variáveis “m”,”n”,”i” e “j” são iteradores utilizados para representar cada posição das matrizes, como a matriz “h” é a matriz de entrada contendo a imagem, é nela em que é feita o deslocamento da matriz filtro, os iteradores “m” e “n” representam o *offset* gerado a cada iteração do somatório.

Estas CNN não são formadas apenas por camadas de convoluções, em algum momento as características extraídas por estas camadas devem ser analisadas para gerar sua respectiva classificação. Após a última camada de convolução, a próxima camada geralmente utilizada é conhecida como *flatten,* sua função é bastante simples, nesta camada todas as matrizes geradas pelos filtros da última camada de convolução são agrupadas e transformadas em um único vetor, que servirá como entrada para a ANN que analisará este vetor, por exemplo, em uma determinada CNN sua última camada de convolução produz matrizes com tamanho de 32x32 e possui 128 filtros nesta camada, logo estas matrizes irão gerar um vetor com 131072 (32*32*128) posições.

A Figura 11 mostra a topologia sugeria do LeCun (1998) para uma rede com o propósito de detectar números escritos a mão.

Figura 8 - Topologia da LeNet



Fonte: (LECUN et al., 1998)

1.1.1.3 CNNs baseadas em regiões (R-CNN)

A criação das R-CNN foi um dos avanços mais importantes da visão computacional, o propósito deste tipo de modelo de rede é fazer a detecção de um objeto em uma imagem, até a criação das R-CNN era somente possível fazer a classificação da imagem inteira, agora além de ser possível a classificação de um objeto, também é possível detectar sua posição na imagem. Basicamente este

modelo de rede se divide em duas etapas, a detecção do objeto e a classificação desse objeto.

O primeiro modelo deste tipo de rede surgiu em 2014, esta rede consiste em 3 módulos, o primeiro é responsável for gerar regiões de interesse (ROI) onde possa conter algum objeto, o segundo módulo é uma CNN que faz a extração de características do ROI em questão, e o terceiro modulo utiliza classificadores SVM para cada classe com kernels lineares para classificar o ROI. (GIRSHICK et al., 2014)

Apesar da R-CNN ter sido bastante importante para comunidade de visão computacional e ser citada cerca de 1600 vezes, essa rede possui o problema de ser bastante lenta, testes realizados com a ResNet101 como o segundo modulo da R-CNN, a R-CNN demorou aproximadamente 50 segundos para processar apenas uma imagem. Nos anos seguintes grandes melhorias foram alcançadas com o desenvolvimento de novas redes, algumas delas baseadas no conceito principal da R-CNN, como a *Fast R-CNN*, *Faster R-CNN* e *MASK RCNN*, e outras utilizando novos conceitos como a Yolo, Yolo9000 e SSD.(GIRSHICK, 2015; IMPIOMBATO et al., 2015; LIU et al., 2016; REDMON; FARHADI, 2016; KAIMING HE GEORGIA GKIOXARI PIOTR DOLL´AR ROSS GIRSHICK, 2017; REN et al., 2017)

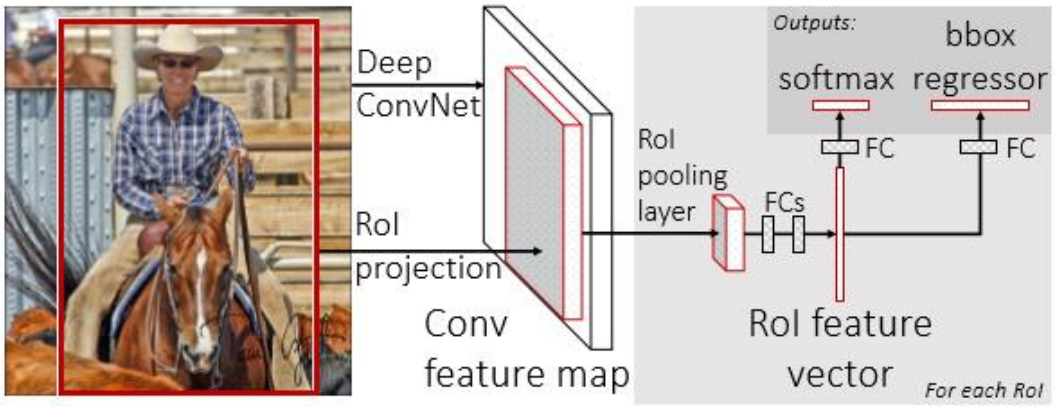
Foram selecionadas duas técnicas para serem estudas, uma proveniente da original R-CNN e outra utilizando um novo conceito, ambas as técnicas foram escolhidas analisando seu desempenho tanto em sua precisão quanto em sua velocidade de processamento, a técnica selecionada proveniente da R-CNN foi a *Faster R-CNN*, e a outra foi a Yolo9000.

1.1.1.3.1 Faster R-CNN

A *Faster R-CNN* é uma evolução da *Fast R-CNN*, cujo é uma evolução da R-CNN. Para melhor compreensão do funcionamento da *Faster R-CNN* é necessário entender o que houve de evolução da *Fast R-CNN*.

A grande diferença na *Fast R-CNN* é que este modelo de rede a imagem passa apenas uma vez pelas camadas de convolução, diferente da R-CNN a *Fast R-CNN* passa toda a imagem uma única vez nas camadas de convoluções, desse modo o vetor de características gerado na última camada desta rede é compartilhado para cada ROI gerado, outra característica desta rede é o uso da camada chamada *ROI Pooling Layer*, nesta camada os ROI's são transformados em um tamanho padrão, de forma que o vetor de características tenha o tamanho correto para se torna a entrada das camadas totalmente conectadas *(Fully Connected Layer FC*), nestas camadas cada neurônio é conectados com todos os neurônios da camada seguinte.

Figura 9 - Topologia da Fast R-CNN

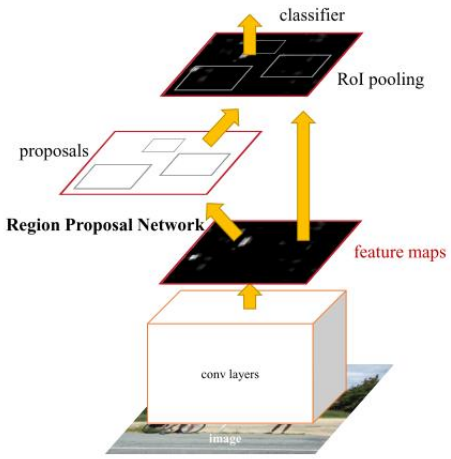


Fonte: (GIRSHICK, 2015)

De acordo com a figura 14 as posições da imagem a serem analisadas são geradas através de uma busca seletiva (UIJLINGS et al., 2012), para cada região proposta é gerado um ROI e logo para cada ROI é gerado um vetor de características que se torna a entrada das camadas totalmente conectadas e então a rede é dividida em duas FC, uma utiliza a ativação *softmax* e gera um vetor de probabilidade para cada classe treinada, este vetor tem o tamanho de N + 1 classes, a classe adicional abrange os objetos nos quais a rede não foi treinada, a outra FC é uma camada de regressão com um conjunto de 4 valores em sua saída, este valores representam os pontos de um quadro onde o objeto está localizado. (GIRSHICK, 2015)

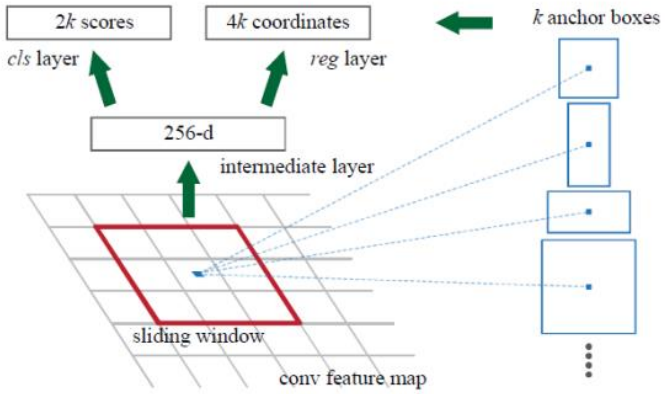
A diferença entre a *Fast R-CNN* e a *Faster R-CNN* é a forma em que as redes analisam as áreas da imagem com possíveis objetos, ao invés de utilizar uma busca seletiva, conforme a figura 16 a *Faster RCNN* utiliza uma quantidade pré determinada de molduras (*anchors*) para cada pixel do mapa de características gerado pela última camada de convolução, por padrão é utilizado 3 diferentes formas de moldura, variando sua altura e largura, e pra cada moldura é utilizado 3 fatores de escala, desta forma é gerado 9 molduras ao todo atrelada a cada pixel desse mapa de características.(REN et al., 2017) Cada moldura se torna a entrada de uma sub-rede (chamada de *Roi Proposal Networks (RPN)*) que analisa a possibilidade de haver um objeto dentro desta região, a resposta desta rede é gerada através de um FC com dois neurônios utilizando a função de ativação *softMax*, esta camada representa a possibilidade de haver um objeto dentro das possibilidades das classes treinadas, a outra camada de resposta desta rede é um FC que utiliza regressão logística nos quais os valores representam um quadro que contorna o possível objeto, na sequência, com o valores de posição e dimensão do quadro gerado pela RPN é aplicada a *ROI Pooling Layer,* onde novamente os valores servem de entrada para uma outra sub-rede que é responsável por classificar o objeto dentro das possíveis classes e também dar a posição e tamanho do quadro.

Figura 10 - Topologia da Faster R-CNN



Fonte: (REN et al., 2017)

Figura 11 - Funcionamento da sub-rede RPN



Fonte: <https://vincentweisen.wordpress.com/2016/05/04/ammai-lecture-10-deep-learning-methods-for-object-detection-i/>

A grande melhoria da Faster R-CNN além de uma predição mais acurada é sua velocidade, este modelo de rede é cerca de 250 vezes mais rápido que o modelo original, a R-CNN.(REN et al., 2017)

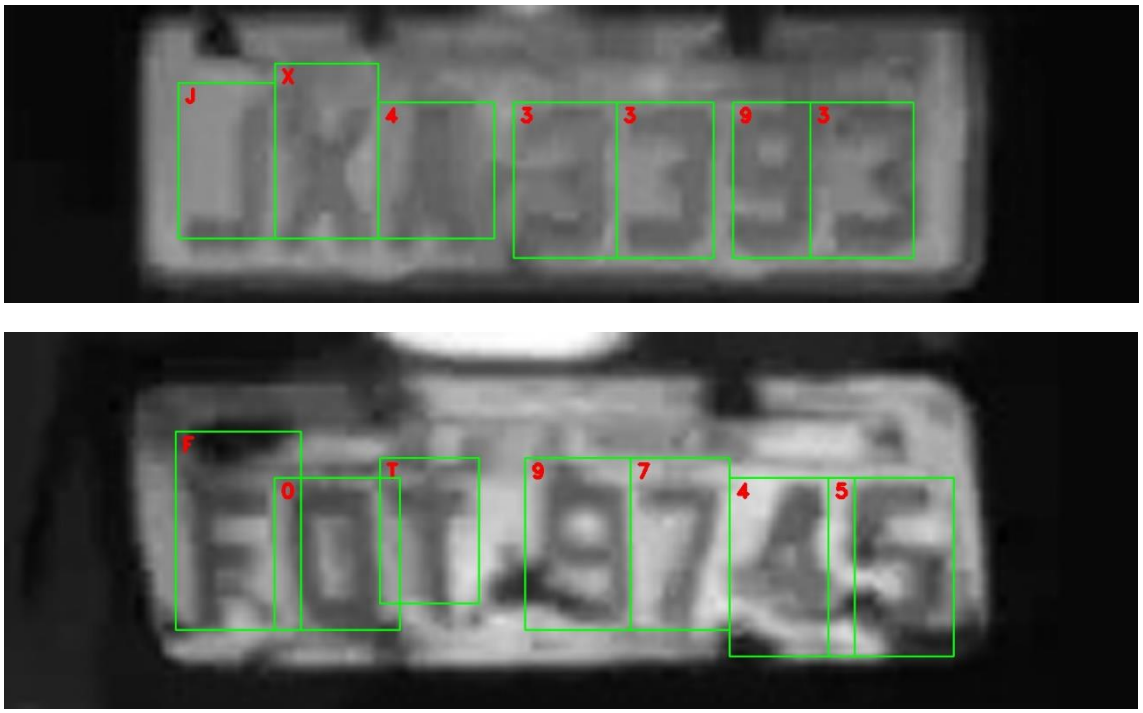
Tabela 1 - Comparação de desempenho

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Fonte: <https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html>

Na imagem 17 foi treinada uma *Faster R-CNN* utilizando como base uma rede derivada da VGG16, esta rede foi treinada com o propósito de fazer OCR em placas de veículos, mesmo com imagens de baixa qualidade a rede consegue uma taxa de acerto acima de 90% com um tempo relativamente baixo para predição, cerca de 600ms utilizando um CPU core I7, isto demonstra o quão versátil este modelo de rede pode ser.

Figura 12 - OCR em imagens de baixa qualidade



Fonte: O autor (2017)

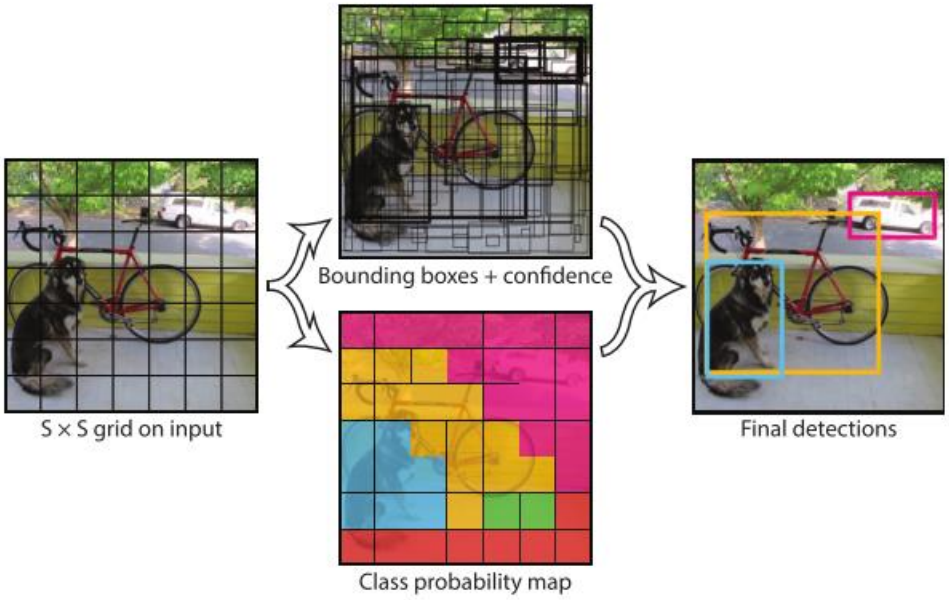
1.1.1.3.2 YOLO

YOLO é a abreviatura para *You Only Look Once*, este modelo de rede utiliza um tipo diferente de abordagem, a abordagem utilizada pela YOLO e por algumas outras redes é conhecida como *One-Shot*, este método de abordagem geralmente está atrelado a redes que conseguem um alto índice de desempenho no quesito quadros por segundo (*fps*) quando comparado a redes como *a Faster R-CNN*, o modelo de abordagem *One-Shot* é utilizado em redes onde a imagem passa uma única vez na rede, esta única passada na rede é o suficiente para localizar todos os objetos na imagem, isto que difere das demais redes, como no caso da *Faster R-CNN*, onde uma sub-rede é utilizada para cada pixel do mapa de característica da última camada de convolução.

A YOLO utiliza imagens de tamanho fixo de 448x448, ao invés de utilizar métodos como janelas deslizantes e técnicas para achar regiões propostas, a YOLO observa toda a imagem durante o treinamento e codifica informações sobre as classes, assim como sua aparência. Outra característica desta rede é que ela aprende a generalizar representações de objetos, após o seu treinamento, isto significa que a YOLO consegue generalizar características de objetos melhor que outras redes, pelo motivo de analisar uma área maior da imagem (IMPIOMBATO et al., 2015)

Esta rede divide a imagem ema uma grade de tamanho S x S. Se o centro do objeto estiver dentro de uma célula destas grades, esta célula é responsável pela detecção e regressão do quadro de contorno do objeto. (IMPIOMBATO et al., 2015) Cada quadro de contorno em que foi feita a predição consistem em 4 valores, os quais são: x, y, w, h e porcentagem de confiança, (x,y) representam o centro do quadro e (w,h) representam sua largura e altura respectivamente, e a porcentagem de confiança é a taxa para que dentro deste quadro haja um objeto de determinada classe onde a rede foi treinada.

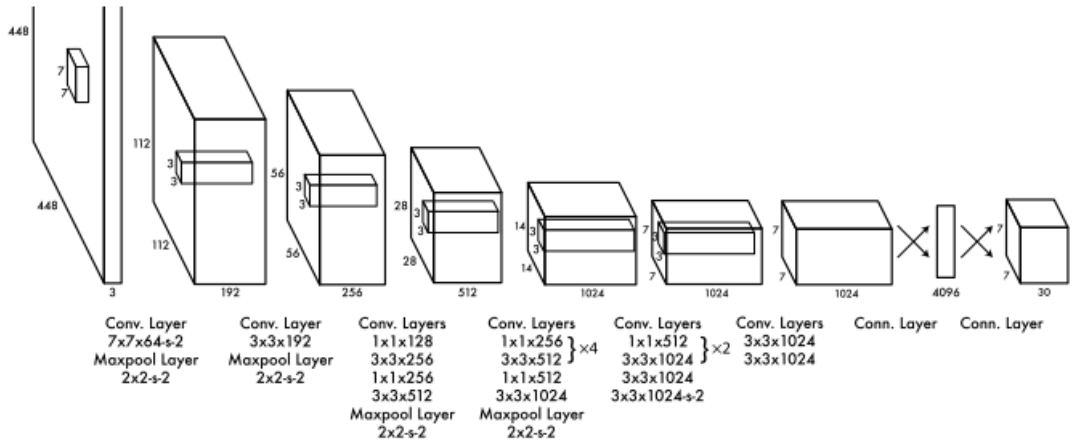
Figura 13 - Método de detecção da YOLO



Fonte: (IMPIOMBATO et al., 2015)

A arquitetura da rede YOLO foi inspirada pela rede GoogleNet (SZEGEDY et al., 2015), a YOLO possui 24 camadas de convoluções seguidas por duas camadas de FC, ao invés de utilizar os módulos de *inception,* é utilizada a pratica de *network in network* (LIN; CHEN; YAN, 2014) seguida por uma camada de convolução com kernel de tamanho 3 x 3. Também há a versão mais rápida da YOLO, no qual é utilizado menos camadas de convolução, apenas 9 ao invés de 24.

Figura 14 - Topologia da rede YOLO

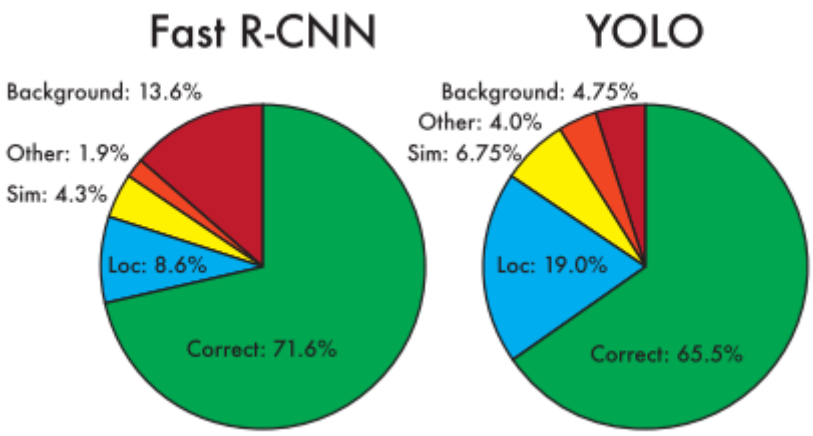


Fonte:(IMPIOMBATO et al., 2015)

A YOLO possui a limitação de poder apenar classificar um pequeno número de objetos em uma imagem, isto porque cada célula da grade onde a imagem foi dividida, pode apenas fazer a predição de uma classe, logo o número máximo de predições para a YOLO é de S^2. Outro ponto fraco deste modelo de rede é que devido ao fato dessa rede fazer a regressão do quadro de contorno através das informações que estão contidas na imagem, a rede tem maior dificuldade em generalizar informações de mesmos objetos com tamanhos e formas diferentes, isto acaba sendo resolvido com o método de força bruta, onde é utilizado um grande número de amostras para treino, fazendo com que a rede acabe aumentando seu poder de memorização.

Este primeiro modelo da YOLO criado, obteve um desempenho ligeiramente inferior comparado a FAST R-CNN no quesito de precisão.

Figura 15 - YOLO vs. FAST R-CNN

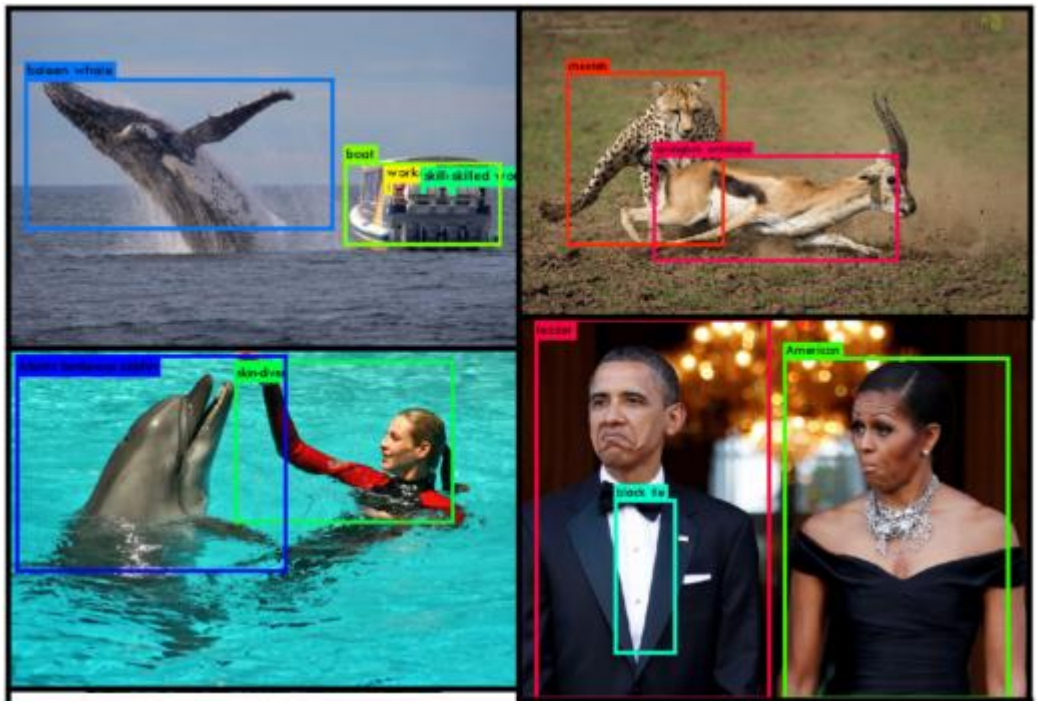


Fonte: (IMPIOMBATO et al., 2015)

A YOLO9000 foi criada a partir de melhorias da YOLO pelos mesmos autores, esta rede teve sua performance em velocidade e precisão melhoras. Uma das principais melhorias é a utilização de *anchors* para fazer a predição do quadro de contorno, assim como na *Faster R-CNN*, porque segundo os autores fazer a predição da largura e altura do quadro de contorno é mais fácil para rede aprender, do que fazer a predição das coordenadas desse mesmo quadro.(REDMON; FARHADI, 2016) Com o uso destes *anchors*, as camadas de FC da YOLO foram removidas e substituídas pelos *anchors*, sendo que também foi removida uma camada de *pooling*, para que o mapa de características permaneça com uma maior resolução, o tamanho da imagem também foi mudada, ao invés de utilizar o tamanho de 448x448 como anteriormente, agora é utilizado o tamanho de 416x416, este tamanho de imagem foi escolhido por conta que o tamanho da grade utilizada na imagem também foi aumentada, anteriormente por padrão era utilizada uma grade de 7x7, e agora este valor passou para 13x13, o valor de 416 foi escolhido porque 416 é divisível por 13 dando como resultado um número inteiro. No modelo de rede anterior o número máximo de objetos detectados por imagem era de 98, agora com o uso de *anchors*, é possível detectar mais de 1000 objetos na mesma imagem.

Com estas modificações a Yolo9000 passou ter precisão melhor ou bastante semelhante comparada a Faster R-CNN, contudo sua velocidade de processamento é cerca de 7 vezes mais rápida.

Figura 16 - Exemplo da utilização da YOLO9000



Fonte: Adaptado de (REDMON; FARHADI, 2016)

1.1.1.4 Treinamento de redes neurais

Para que as redes neurais funcionem adequadamente, é necessário ajustar os pesos da rede de modo com que gerem a resposta apropriada, para este ajuste são utilizadas informações de entrada nos quais seus valores de saída sejam conhecidos para servir de referência ao ajustar os pesos, este método para ajustar uma rede neural é conhecido como treinamento.(WU, 2017)

O método de treinamento utilizado para o ajuste desses pesos é chamado de *backpropagation,* este método é divido em 4 etapas distintas, as quais são:

- *Forward pass*
- *Loss function*
- *Backward pass*
- *Weight update*

A primeira etapa é o *forward pass*, nesta é utilizada uma imagem de treinamento, a qual passa por toda a rede, e a partir disto uma resposta na saída da rede é gerada, porém esta resposta gerada será aleatória, uma vez que os pesos da rede são inicializados com valores aleatórios próximos a zero. A partir deste ponto, a resposta gerada é utilizada na segunda etapa, durante a aplicação do *loss function.* O uso desta função gera um valor de referência do erro do valor de saída comparado ao valor de referência utilizado no treinamento, uma função bastante comum para se calcular o erro da rede é através do MSE (*mean squared error*)(WU, 2017), essa função calcula o erro através da soma do erro ao quadrado da rede, e é representada pela seguinte fórmula:

$$\boldsymbol{Erro} = \sum \frac{\mathbf{1}}{\mathbf{2}} \times (\boldsymbol{y} - \boldsymbol{y'})^{\mathbf{2}} \qquad \textbf{(3)}$$

Onde “y” é o valor resposta verdadeiro em relação a entrada e “y’” é o valor gerado pela rede.

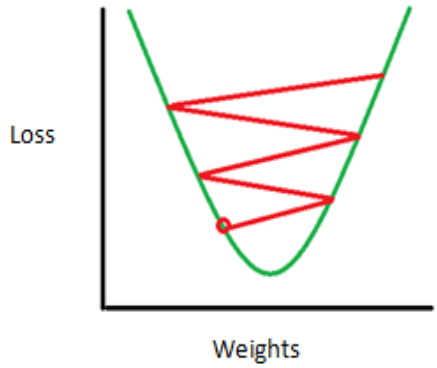
Com o valor do MSE obtido, então é aplicada a etapa de *backward pass*, nesta etapa os pesos são analisados para descobrir qual peso contribuiu mais para o aumento do valor de MSE, e assim procurar formas para que o valor do MSE seja menor na próxima iteração do treinamento. Matematicamente a fórmula que expressa esta etapa é:

$$\boldsymbol{w} = \boldsymbol{w_i} - \aleph \times \frac{\boldsymbol{dL}}{\boldsymbol{dW}} \qquad \textbf{(4)}$$

Sendo w o valor do peso, w_i o valor inicial do peso e $\aleph$ o fator de aprendizado.

Com a aplicação desta fórmula os pesos da rede são atualizados em sua última etapa do processo de treinamento, com os valores dos pesos atualizados o valor do erro calculado tenderá a minimizar aproximando de um ponto otimizado ao decorrer do treinamento.

Figura 17 - Decrescimento do MSE ao decorrer do treinamento



Fonte: WEB

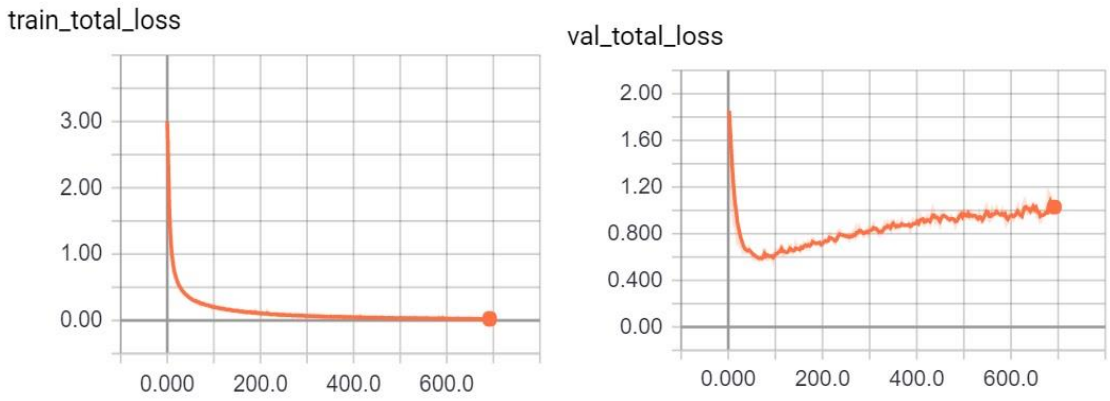
As 4 etapas do treinamento são repetidas para todas as imagens do conjunto de dados utilizado para o treinamento. Cada imagem ou um grupo de imagens do conjunto de dados de treinamento é conhecido como lote (batch), um período de treinamento (epoch) é a quantidade de iterações necessária para que a rede seja treinada para cada lote de treinamento. É comum um processo de treinamento precisar de vários períodos para convergir em uma boa precisão de resposta.(AP et al., 2017)

A validação da precisão de uma rede ocorre utilizando um conjunto de dados diferente do conjunto utilizado em seu treinamento, este conjunto de teste garante que a precisão da rede não estará viciada ao conjunto de treinamento. Este fenômeno é chamado de *overfitting* e ocorre quando um classificador em geral tem uma excelente performance quando aplicado ao conjunto de treinamento, e uma performance ruim quando aplicado a outros conjuntos que não seja o de treinamento. Isto ocorre quando os parâmetros do classificador se ajustaram tanto as características do conjunto de treinamento que não consegue mais classificar algo diferente deste conjunto. Geralmente este fenômeno acontece quando se utiliza valores muito alto para os períodos de treinamento.(DONGEN; GÜNTHER, 2010)

A figura 16 demonstra uma situação de *overfitting*, porque a função *loss* atingiu seu mínimo na *epoch* 63 com um valor de 0.57 no teste de validação, porém a função *loss* quando aplicada ao conjunto de dados de treinamento continua a diminuir ao decorrer das *epoch* de treinamento. Pelo fato da função *loss* ter um valor bastante próximo a zero ao aplicada no conjunto de treinamento e um valor alto quando aplicada ao conjunto de validação em relação ao conjunto de treinamento, isto demonstra que a rede projetada para esta situação tem a capacidade de uma boa classificação, porém o conjunto de dados de treinamento é insuficiente em

quantidade, não conseguindo generalizar a situação real e assim causando o *overfitting*.

Figura 18 - Classificador com característica de overfitting



Fonte: O autor (2017).

1.2 Rastreamento de objetos

A diferença entre detecção e rastreamento de objetos é que na detecção, de fato ocorre a detecção de um objeto, ou seja, há um determinado grau de certeza em que determinada posição da imagem há um objeto. Enquanto que para o rastreamento não funciona necessariamente com um objeto e sim com um aglomerado de pixels, com as características extraídas deste conjunto de pixel é então verificado sua posição na imagem entre um frame e outro, e este conjunto de pixel pode vir a ser parte de um objeto, e portando acabar realizando o rastreamento de um objeto. (ROTH, 2008)

1.2.1 Blob tracking

Na visão computacional *blob* compreende o conceito de detectar regiões na imagem em que possuem determinada característica em comum, como cor e brilho. (GREENSTED, 2009) A técnica de detecção do plano de fundo pode gerar uma máscara onde contém os *blobs* de objetos que não estão estaticamente parados. Os *blobs* geralmente são gerados através de uma imagem chamada de máscara, esta máscara é uma imagem binária com mesmo tamanho e formato que a original, as regiões detectadas em geral recebem o valor de 255, e para outras circunstâncias recebe o valor de 0.

Figura 19 - Exemplo de Blobs

Fonte: O autor (2017)

O rastreamento desses *blobs* podem ser feitas através de técnicas como *SIFT* e *SURF* (JUAN; GWUN, 2009; ZHOU; YUAN; SHI, 2009), *Camshift* e Mean-Shift (ALLEN; XU; JIN, 2006; LEICHTER; LINDENBAUM; RIVLIN, 2010) ou também com técnicas mais básicas, como a comparação da distância euclidiana entre *blobs* através de diferentes quadros do vídeo, considera-se o mesmo objeto determinado *blob* que teve sua centroide alterada dentro de uma distância máxima tolerada. (BOCHEM; KENT; HERPERS, 2011)

3 DESENVOLVIMENTO DO TEMA

Para o desenvolvimento, será utilizada duas câmeras, ambas as câmeras instaladas no interior do veículo, a primeira câmera será responsável pelo monitoramento do motorista, já a segunda câmeras será responsável na detecção de objetos em meio ao trânsito, para melhor posicionamento está câmera ficará localizada nas proximidades do retrovisor do meio.

Esse estudo foi dividido em dois módulos, o primeiro módulo é relacionado ao monitoramento do motorista, e o segundo módulo será responsável pelo monitoramento do trânsito, essa divisão facilitará alcançar os objetivos específicos, e também na possível falha do estudo, esta divisão aumenta a probabilidade de alcançar os objetivos específicos parcialmente, por fim, a divisão por módulos também irá facilitar a integração de novos módulos caso possa haver, ou na melhoria individual caso seja necessária ou haja a oportunidade.

O primeiro módulo responsável pela classificação da direção do olhar do condutor, irá fornecer dados ao segundo módulo que baseado na classificação do primeiro módulo irá determinar para qual área da imagem deve focar seu processamento e classificação de objetos.

A programação desses algoritmos será feita em Python, utilizando uma das principais bibliotecas de processamento de imagem, cujo nome é OpenCV, essa biblioteca terá como maior objetivo a manipulação e edição das imagens, como por exemplo, adicionar ruídos, mudar o tamanho da imagem, ajustar o brilho, etc. Outra biblioteca importante na qual será utilizada é o Keras, essa biblioteca foi criada a fim de abstrair funções mais complexas de outra biblioteca focada no estudo de deep learning, dessa maneira a criação e edição das redes criadas se torna mais fácil, uma das bibliotecas na qual o Keras abtrai é o TensorFlow, o TensorFlow é uma ferramenta poderosa para redes neurais, criada pelo google, essa ferramenta possui funções bastante uteis para acompanhar e comparar resultados, como o TensoBoard, onde essa função gera um WebService, com inúmeras variais de treinamento, por gerar um WebService, é possível acompanhar o treinamento a longa distância, como inserir esse treinamento em um servidor da Amazon.

3.1 Módulo 1

No desenvolvimento desse módulo será treinada uma rede neural capaz de classificar a direção em que o motorista esteja olhando, esta rede deverá ser capaz de classificar a direção do olhar do condutor em 3 classes, direita, esquerda e frontal. Com base nos estudos feitos, a rede neural selecionada para esta tarefa será baseada na rede darknet ou squeezeNet, ambas as redes demostraram ter um bom desempenho na predição da classes dos objetos existentes no dataset do ImageNet, e também possuem pequenas quantidades de parâmetros quando comparadas as redes mais tradicionais como a VGG16, essa menor quantidade de parâmetros irá auxiliar no desempenho de velocidade de processamento, assim como também facilitara o treinamento para convergir em um resultado com maior precisão, o desempenho de velocidade é particularmente importante porque este estudo visa o funcionamento do algoritmo em um sistema embarcado no veículo com a ausência de uma GPU.

O dataset será gerado a partir de um vídeo gravado no interior do veículo, cada frame desse vídeo de treinamento será classificado manualmente dentro de 3 classes, um segundo vídeo irá passar pelo menos procedimento, porém esse vídeo terá uma posição diferente de câmera, também como será gravado com outro condutor, esse vídeo será responsável pela validação do treinamento dessa rede.

3.2 Módulo 2

Nesse módulo será utilizada as redes Faster RCNN e YOLO, nesse módulo deverá ser utilizada especialmente este modelo de rede, porque essas redes são capazes de além de classificar objetos, também conseguem localiza-lo na imagem, a localização do objeto é especialmente útil, por conta que será possível saber se determinado objeto se encontra no ponto cego do motorista ou não. Em um primeiro momento será utilizada a rede YOLO, essa rede foi selecionada por primeiro pelo seu desempenho descrito em seu artigo. Nos primeiros testes também não serão feitos nenhum tipo de treinamento, esta rede será carregada com os pesos já treinados por seu autor, originalmente esta rede já possuiu todas as classes que possam ser de interesse para esse estudo, como a classificação de humanos, tipos de veículos e objetos comuns no trânsito, como o semáforo por exemplo. Caso essa rede com os pesos já treinados não obter o desempenho esperado, então será criado um dataset especifico para a situação, esse dataset será criado de forma semelhante ao dataset do módulo um, a diferença é que em cada frame será anotado com as coordenadas e classificação dos objetos existentes nesse frame, por fim, se mesmo assim o desempenho ainda não for satisfatório, será refeito o

mesmo procedimento com a rede Faster RCNN, contudo com esse modelo de rede, espera-se um desempenho de velocidade aproximadamente 7 vezes mais lento.

4 CONSIDERAÇÔES FINAIS

O Deep learning demonstrou-se ser uma ferramenta muito poderosa para trabalhar com imagens, contudo no desenvolvimento dessa pesquisa, as redes neurais convolucionais utilizadas obtiveram um desempenho de velocidade inferior ao esperado, isso gera a necessidade de utilizar um dispositivo provido de uma GPU para que os processamentos dos frames possam ocorrer em tempo real.

A rede YOLO obteve um resultado muito além do esperado para a classificação, para alcançar os objetivos específicos, esperava-se que a rede fosse capaz de detectar humanos com um erro satisfatório, porém além de conseguir fazer essa classificação, a YOLO conseguiu classificar diversos objetos com uma precisão bastante alta para objetos a curta e média distância, alcançando um resultado muito próximo ao ideal.

Já a rede darknet utilizada para classificar a direção do olhar do motorista, teve um resultado satisfatório, mas há espaços para melhorias, como adicionar novas classes que possam, por exemplo, detectar se o condutor está ou não distraído, ou até mesmo utilizar redes capazes de detectar o estado físico e emocional do motorista.

Devido ao fato da rede YOLO possuir um baixo desempenho em frames, não foi possível validar o algoritmo em uma situação real, apenas valida-lo em simulações.

Para trabalhos futuros essa pesquisa abriu a oportunidade de minerar a quantidade de informações geradas pelas redes neurais, como classificação dos objetos e suas respectivas posições.

REFERÊNCIAS

ALLEN, J. G.; XU, R. Y. D.; JIN, J. S. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. **Proceedings of the Pan-Sydney area workshop on Visual information processing**, v. 36, p. 3–7, 2006. Disponível em: <http://dl.acm.org/citation.cfm?id=1082122>.

AP, G. E. G.; INIMA, S. H. M.; NOCEDAL, J.; TAK, P.; TANG, P. ON LARGE - BATCH TRAINING FOR DEEP LEARNING : **ICLR**, p. 1–16, 2017.

BERESFORD, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. **Journal of Pharmaceutical and Biomedical Analysis**, v. 22, p. 717–727, 2000.

BOCHEM, A.; KENT, K.; HERPERS, R. **FPGA Based Real-Time Tracking Approach with Validation of Precision and Performance**, 2011. .

DONGEN, H. M. W. V. B. F. Van; GÜNTHER, E. K. C. W. Process mining : a two-step approach to balance between underfitting and overfitting. **Software & Systems Modeling**, p. 87–111, 2010.

GIRSHICK, R. Fast R-CNN. **International Conference on Computer Vision**, 2015. Disponível em: <http://arxiv.org/abs/1504.08083>.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 580–587, 2014.

GREENSTED, A. Blob Detection. **The Lab Book Pages**, 2009. Disponível em: <http://www.labbookpages.co.uk/software/imgProc/blobDetection.html>.

HUBEL, B. Y. D. H.; WIESEL, A. D. T. N. RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX. **J. Physiol**, p. 106–154, 1962.

IMPIOMBATO, D.; GIARRUSSO, S.; MINEO, T.; CATALANO, O.; GARGANO, C.; LA ROSA, G.; RUSSO, F.; SOTTILE, G.; BILLOTTA, S.; BONANNO, G.; GAROZZO, S.; GRILLO, A.; MARANO, D.; ROMEO, G. You Only Look Once: Unified, Real-Time Object Detection. **Computer Vision and Pattern Recognition**, v. 794, p. 185–192, 2015.

JUAN, L.; GWUN, O. A comparison of sift, pca-sift and surf. **International Journal of Image Processing (IJIP)**, v. 3, n. 4, p. 143–152, 2009.

KAIMING HE GEORGIA GKIOXARI PIOTR DOLL´AR ROSS GIRSHICK. Mask R-CNN. **Computer Vision and Pattern Recognition**, 2017.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, 1990.

KRIZHEVSKY, A.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems**, p. 1097-, 2012.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-Based Learning Applied to Document Recognition. **Proceedings of the IEEE**, 1998.

LEICHTER, I.; LINDENBAUM, M.; RIVLIN, E. Mean Shift tracking with multiple reference color histograms. **Computer Vision and Image Understanding**, v. 114, n. 3, p. 400–408, 2010. Disponível em: <http://dx.doi.org/10.1016/j.cviu.2009.12.006>.

LIN, M.; CHEN, Q.; YAN, S. Network In Network. **Neural and Evolutionary Computing**, p. 10, 2014. Disponível em: <http://arxiv.org/abs/1312.4400>.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.; BERG, A. C. SSD: Single Shot MultiBox Detector. **Computer Vision and Pattern Recognition**, 2016. Disponível em: <http://arxiv.org/pdf/1512.02325v2>.

MCCULLOCH, W.; PITTS, W. A Logical Calculus Of The Ideas Immanent In Nervous Activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.

REDMON, J.; FARHADI, A. YOLO9000: Better, Faster, Stronger. **Computer Vision and Pattern Recognition**, 2016. Disponível em: <http://arxiv.org/abs/1612.08242>.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 6, p. 1137–1149, 2017.

ROJAS, R. The Hopfield Model. **Neural Networks**, 1996.

ROSENBLATT. **The perceptron A perceiving and recognizing automaton**, 1957. .

ROTH, S. People-Tracking-by-Detection and People-Detection-by-Tracking. **Computer Vision and Pattern Recognition**, 2008.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 07–12–June, p. 1–9, 2015.

TAHMASEBI, P.; AMIRKABIR, A. H. Application of a Modular Feedforward Neural Network for Grade Estimation Application of a Modular Feedforward Neural Network for Grade Estimation. **Natural Resources Research**, n. April 2016, 2011.

UIJLINGS, J. R. R.; VAN DE SANDE, K. E. A.; GEVERS, T.; SMEULDERS, A. W. M. Selective Search for Object Recognition. **International Journal of Computer Vision**, 2012. Disponível em: <https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf>.

WU, J. **Introduction to Convolutional Neural Networks**, 2017. .

ZAMARREN, M.; GONZA, P. A. Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. **Energy and Buildings**, v. 37, p. 595–601, 2005.

ZHOU, H.; YUAN, Y.; SHI, C. Object tracking using SIFT features and mean shift. **Computer Vision and Image Understanding**, v. 113, n. 3, p. 345–352, 2009. Disponível em: <http://dx.doi.org/10.1016/j.cviu.2008.08.006>.