CENTRO UNIVERSITÁRIO FEI
ARTHUR AMARAL DE OLIVEIRA

INTERPRETAÇÃO DE SITUAÇÕES DE RISCO EM CENAS DE TRÁFEGO DE VEÍCULOS UTILIZANDO LÓGICA PROBABILÍSTICA

São Bernardo do Campo
2019

ARTHUR AMARAL DE OLIVEIRA

INTERPRETAÇÃO DE SITUAÇÕES DE RISCO EM CENAS DE TRÁFEGO DE VEÍCULOS UTILIZANDO LÓGICA PROBABILÍSTICA

Dissertação de Mestrado apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Paulo Eduardo Santos.

São Bernardo do Campo

2019

Oliveira, Arthur Amaral de.
INTERPRETAÇÃO DE SITUAÇÕES DE RISCO EM CENAS DE TRÁFEGO DE VEÍCULOS UTILIZANDO LÓGICA PROBABILÍSTICA / Arthur Amaral de Oliveira. São Bernardo do Campo, 2019.
83 f. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Paulo Eduardo Santos.

1. Raciocínio Probabilístico. 2. Redes Neurais Artificiais. 3. Sistemas de Assistência ao Motorista. I. Santos, Paulo Eduardo, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Arthur Amaral de Oliveira **Matrícula:** 116302-1

Título do Trabalho: Interpretação de situações de risco em cenas de tráfego de veículos utilizando lógica probabilística.

Área de Concentração: Inteligência Artificial Aplicada à Automação e Robótica

Orientador: Prof. Dr. Paulo Eduardo Santos

Data da realização da defesa: 24/09/2019

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Paulo Eduardo Santos	Ass.: _____
Prof. Dr. Flavio Tonidandel	Ass.: _____
Prof. Dr. Fábio Gagliardi Cozman	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Dedico este trabalho a minha família por ter me entendido minha ausência durante sua realização.

AGRADECIMENTOS

Agradeço em especial meu professor orientador Paulo Eduardo Santos que me auxiliou e me guiou durante esta longa e prazerosa jornada.

RESUMO

A desatenção ou erro de motoristas atrasam suas reações em situações de risco em que uma frenagem é necessária para se evitar uma colisão (Roadmap, 2017). Essa é uma razão do crescimento da utilização de sistemas de assistência ao motoristas disponíveis nos automóveis atualmente. De fato os motoristas desejam carros mais seguros, um outro fator que contribui para a ampliação da utilização desses sistemas é regulatório, uma vez que serão obrigatórios nos países da comunidade Europeia e nos Estados Unidos a utilização de no mínimo sistemas de freio de emergência (Roadmap, 2017). Este trabalho apresenta um sistema de assistência ao motorista, que se utiliza da fusão de dados, capaz de gerar um alerta preventivo de risco de colisão. O sistema é capaz de raciocinar sobre o contexto de tráfego de veículos, utilizando-se uma base de conhecimento sobre direção defensiva, definida pelo autor. Com o uso de uma rede neural convolucional, este trabalho segmenta e classifica veículos em um contexto de tráfego. Através de técnicas de visão computacional, em especial a transformada Hough, as faixas de rolagem são segmentadas para discretização da cena em relação à posição espacial. Combinando as informações obtidas pela rede neural convolucional e da segmentação das faixas de rolamento, é possível discretizar as posições de todos os veículos da cena. Essa junção de informações são evidências para que o sistema, codificado em lógica probabilística usando-se ProbLog, realize a inferência do risco. O sistema foi avaliado em duas diferentes sequências de cenas, onde foram avaliados 10 diferentes limiares de risco, comparando-os com o limiar de um especialista. O resultado mostrou valor de Precisão 1 e Revocação 0.906 para a cena 1, e para a cena 2 Precisão 0.579 e Revocação 0.879, o que mostra que este tipo de solução utilizando-se lógica probabilística é promissora.

Palavras-chave: Raciocínio probabilístico. Redes neurais convolucionais. Sistemas de Assistência ao motorista.

ABSTRACT

Drivers have a late reaction time in risky situation, due to inattention or by mistake, when a breaking is necessary to avoid a collision (Roadmap, 2017). This is one of reasons for the growth of the driver's advanced assistance systems available in the vehicles nowadays, moreover drivers want safer vehicles, another factor to the usage of these systems is regulatory, since it will be mandatory on European community countries and United States emergency breaking systems (Roadmap, 2017). This work presents an advanced driver assistance system, using data fusion, capable of generate a preventive collision alert. The system is capable of reasoning over a vehicle traffic context, using defensive driving as knowledge base built by the Autor. A convolutional neural network is used in this work to perform segmentation and classification of vehicles in traffic context. Through computational vision, specially Hough's transformation, the road lanes are segmented and then the scene is discretized by its spatial position. Using the information from the convolutional neural network with the road lane segmentation is possible to discretize the position of all vehicle in the scene. That information combined are evidences to the system, coded in ProbLog a probabilistic logic programing, to infer the scene risk. The system was evaluated in two different scenes chains, where10 risk limits were analyzed, comparing the risk limits against the limits provided by a specialist. The results showed Precision of 1 and Recall of 0.906 for scene 1 and for scene 2 Precision of 0.579 and Recall 0.879, what shows such solution using probabilistic logic is promising.

Keywords: Probabilistic reasoning. Convolutional neural network. Advanced driver assistance systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com o programa Europeu de avaliação carros novos, cuja sigla, EuroNCAP, do inglês "European New Car Assessment Programme", ROADMAP (2017), cita em seu estudo que 90% dos acidentes são causados por erros humanos. Em geral, são dois problemas que contribuem para que esses erros aconteçam. O primeiro, está ligado às violações das regras de trânsito, como velocidade acima do permitido, direção sobre influência de drogas ou álcool. O segundo, está relacionado à falta de atenção ao trânsito, à fadiga, à distração e à inexperiência do motorista.

Como resposta ao segundo problema, a indústria automotiva vem desenvolvendo diversos sistemas de auxílio à direção, em inglês "Advanced Driver Assistance Systems"(ADAS), para dar suporte aos motoristas com a finalidade de prevenir e mitigar os erros humanos na condução de veículos. Hoje, os sistemas de aviso de troca de faixa, a assistência para se manter dentro da faixa, os sistemas autônomos de emergência de mudança de direção, o piloto automático adaptativo, os sistemas de assistência de velocidade, entre outros, são os mais comuns.

Os sistemas ADAS de prevenção de colisão traseira são considerados atualmente efetivos segundo Nakamori et al. (2002), estes tipos de sistemas medem as distâncias a frente utilizando-se de radares laser e também de câmeras estéreo, que são utilizados por algoritmos para antecipar situações perigosas, que pode gerar de um alerta e até a execução de ação independente do motorista.

Segundo estudo da Organização Mundial de Saúde, conforme mostrado em Kukkalaet al. (2018), todo ano ocorrem 1,25 milhões de mortes por acidentes relacionados ao tráfego de veículos, o que significa um gasto anual de US$ 518 bilhões. Esses dados, além de demandarem de forma crescente medidas de segurança dos motoristas, fizeram com que os fabricantes de automóveis desenvolvessem sistemas avançados de auxílio à direção de veículos.

Para poder representar o contexto de tráfego de veículos, as técnicas de Aprendizado de Máquina, mais precisamente as Redes Neurais Artificiais (RNA) e, em especial, Redes Neurais Convolucionais (RNC), têm obtido, nos últimos anos, grandes resultados no campo da visão computacional no que se refere a detectar, classificar e segmentar objetos. Ano após ano, os resultados obtidos no desafio de detecção e classificação de objetos proposto pela *ImageNet*, uma base de dados com mais de 14 milhões de imagens (Denget al. (2009)), mostram os avanços conseguidos pelas RNCs.

Sistemas mais avançados de prevenção de acidentes utilizam-se tanto de radar a laser quanto de câmeras (NAKAMORI et al., 2002). A utilização dos dois tipos de sensores dá pois são complementares, câmeras conseguem interpretar as faixas de rolagem e limites de pista, enquanto os radares laser conseguem identificar obstáculos sólidos como outros veículos.

Apesar do aumento da quantidade de sistemas de aprendizado de máquina, as redes neurais não conseguem interpretar o contexto como um todo, mas sim mostrar todos os objetos que compõem aquele contexto. A interpretação do contexto e de suas incertezas requer regras e relações espaço-temporais. Demanda o uso do raciocínio probabilístico, utilizando-se de regras de lógica escritas; a lógica probabilística é uma técnica apropriada para modelar esse contexto. De acordo com Shterionov (2015), aplicações que necessitam de um maior formalismo devido ao raciocínio sobre incertezas precisam mais do que lógica. Por essa razão, produziu-se uma vasta quantidade de desenvolvimentos na área de Programação em lógica probabilística. ProbLog é uma ferramenta de programação de lógica probabilística, uma extensão do ProbLog, que além de lógica booleana, pode receber também sentenças em lógica de primeira ordem (SHTERIONOV, 2015).

Este trabalho propõe um sistema ADAS que utiliza redes neurais convolucionais, empregando uma sequência de quadros de uma cena de tráfego de veículos como entrada, para obter, na sua saída, uma lista classificada com as probabilidades de os objetos detectados estarem na cena. Esta lista será utilizada como evidências para um sistema de raciocínio probabilístico, o qual será escrito em ProbLog. O programa contém regras de direção segura no trânsito, que também serão codificadas em ProbLog. Utilizando-se tanto as regras de direção segura no trânsito quanto as evidências provenientes da RNC foi possível inferir a probabilidade de uma cena conter uma situação de risco de colisão entre veículos.

2 REVISÃO TEÓRICA

Neste capítulo discorre-se sobre as teorias básicas que são utilizadas na proposta do trabalho. Em Raciocínio Probabilístico, apresenta-se a programação em lógica com o objetivo de modelar as regras do contexto proposto para a inferência probabilística. Em visão computacional, descrevem-se técnicas de detecção e classificação de objetos utilizando-se as Redes Neurais Artificiais.

2.1 PROGRAMAÇÃO EM LÓGICA

A modelagem e o raciocínio sobre dados relacionais, como explicado em Dries et al. (2015), pode ser feito através de um paradigma de programação, a programação em lógica (PL). Programas lógicos são uma coletânea de afirmações verdadeiras sobre um domínio. Apesar de muitas aplicações de aprendizado de máquina utilizarem-se de ferramentas de PL, a necessidade de raciocínio mais eficiente para tratar incertezas levou ao desenvolvimento de um formalismo que envolve representações relacionais atreladas a asserções probabilísticas, como a programação em lógica probabilística (PLP). A PLP vai além da lógica pura, na qual uma afirmação deve ser verdadeira ou falsa, e considera que uma afirmação pode ser incerta e determinar uma probabilidade a ela.

2.1.1 ProbLog

De acordo com Dries et al. (2015), ProbLog é o estado da arte da implementação de uma linguagem de programação em lógica probabilística. Seu objetivo comum é disponibilizar uma ferramenta poderosa para modelagem e raciocínio sobre domínios estruturados e domínios incertos que aparecem em aplicações, como no reconhecimento de atividades, no processamento de linguagem natural entre outros. O desenvolvimento do ProbLog foi focado especialmente para ser empregado em técnicas de aprendizado de máquina e sua implementação, além disso, é uma das únicas linguagens de programação que possibilita, por exemplo, múltiplas consultas durante o cálculo das probabilidades marginais, além de poder incorporar novas evidências. Dries et al. (2015) explica que ProbLog é uma ferramenta que consiste na linguagem de programação ProbLog e na inferência e aprendizagem em ProbLog. Inicialmente a ProbLog foi proposta como uma ferramenta de PLP, cujo objetivo principal calcular a probabilidade de sucesso de uma pesquisa. Comumente este sistema foi chamado de

ProbLog1. A evolução do ProbLog para uma linguagem capaz de representar modelos gráficos através de lógica e representações relacionais, como por exemplo, os modelos probabilísticos relacionais e programas lógicos bayesianos, é chamado de ProbLog2.

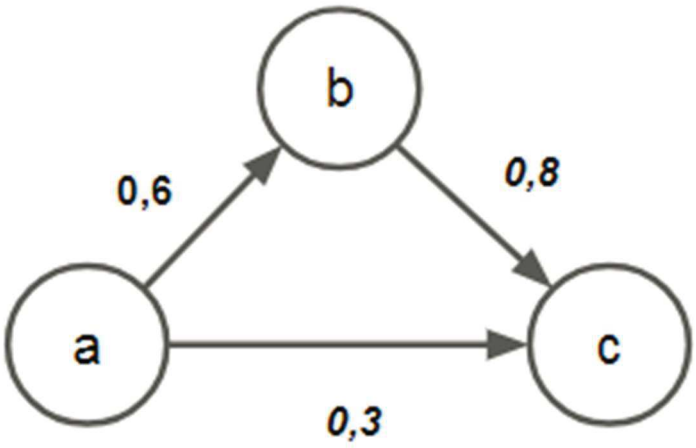
2.1.1.1 Sintaxe

ProbLog, como demonstrado em Dries et al. (2015), é uma linguagem de programação em lógica probabilística (PLP) de propósito geral. Os fatos probabilísticos podem ser simples ou compostos por mais de uma variável. Os fatos probabilísticos são as estruturas mais básicas para representar a incerteza dos dados. A representação de um fato probabilístico tem a forma $p_i :: f_i$ e afirma que que o fato f_i é verdade com probabilidade p_i ou falso com probabilidade $(1-p_i)$. As regras em ProbLog definem consequências lógicas de fatos probabilísticos. No exemplo da figura 1, um grafo modela a incerteza entre três entidades, representadas por nós, e cada dependência é uma ligação com sua respectiva probabilidade. O grafo da figura 1 é expresso através da seguinte sintaxe, levando-se em conta a forma apresentada acima para um fato probabilístico:

a) $p(X, Y) : -e(X, Y)$;

b) $p(X, Y) : -e(X, X1), p(X1, Y)$.

Figura 1-Grafo acíclico de 3 nós



Fonte: Autor

Em ProbLog, expressa-se o grafo da figura 1 da seguinte maneira:

a) 0.6:: e(a,b);
b) 0.3:: e(a,c);
c) 0.8:: e(b,c).

O fato *0.6:: e(a,b)* expressa uma ligação existente ente os nós *a* e *b*, com probabilidade de *p* de *0.6.* Todos os possíveis nós de um grafo são representados dessa maneira. Nós que não existirem podem ser omitidos ou serem declarados com probabilidade zero. Por exemplo: um nó de *b* para *a* seria expresso do seguinte modo: *0.0:: e(b,a).*

Um átomo, é uma sentença que enuncia fatos através de relações ou predicados, no exemplo do grafo na figura 1 a relação entre os nós se dá por uma aresta *e*. Sendo assim, um átomo que se unifica com fato probabilístico é chamado de átomo probabilístico e um átomo que se unifica com a cabeça de uma regra, de átomo derivado. Um conjunto de átomos derivados e de átomos probabilísticos de um programa em ProbLog é uma disjunção. Tomando a figura 1 como exemplo tem-se: *{e(a,b),e(a,c),e(b,c)}* e *{p(a,b),p(a,c),p(b,c),P(a,c)}*.

A linguagem utilizada pelo sistema ProbLog2, também suporta fatos probabilísticos intencionais, que são empregados para compactar um conjunto de fatos probabilísticos, aplica-se a seguinte forma: $P::f(X_1,...,X_n):-Corpo$, onde o corpo é a conjunção de literais e não incluem outros fatos probabilísticos. A probabilidade *P* pode ser um número ou ainda uma variável que unifica com um número quando o corpo é provado. As variáveis X_1 até X_n são instanciadas quando o corpo é provado.

No seguinte fato probabilístico intencional: *0.3::(A,B):−membro(A,[a,b]), membro(B,[c,d]),* estão compactados a representação dos seguintes fatos:

a) 0.3:: nó(a,c);
b) 0.3:: nó(a,d);
c) 0.3:: nó(b,c);
d) 0.3:: nó(a,d).

2.1.1.2 Semântica

Todo fato probabilístico de um programa de ProbLog é uma variável aleatória binária, que pode ser verdadeira ou falsa. Além disso, a escolha do valor verdade de um átomo probabilístico é chamada de escolha atômica. A escolha atômica de todas as probabilidades é a escolha total. Para n fatos probabilísticos, existem duas escolhas totais, e cada escolha forma um único modelo de programa em ProbLog, chamado de mundos possíveis. Dado que átomos

probabilísticos são considerados variáveis independentes e randômicas, é possível determinar a probabilidade de um possível mundo como o produto das probabilidades associadas com as escolhas atômicas.

Sendo $\Omega = \omega_{1,..,}\omega_N$ um conjunto de mundos possíveis em um programa escrito em ProbLog, onde $N = 2^n$, e n é o número de átomos probabilísticos, um único possível mundo pode ser mostrado como uma tupla (w^+_i, w^-_i), onde w^+_i é o conjunto de átomos probabilísticos em w_i que são verdadeiros e w^-_i são falsos.

Formalmente, um programa em ProbLog define a distribuição sobre todos os mundos possíveis como mostrado na equação 1, onde p_i são as probabilidades dos átomos a_i.

$$P(w_i) = \prod_{a_j \in w_i^+} p_j \prod_{a_j \in w_i} (1 - p_j) \quad (1)$$

A somatória sobre as probabilidades de todos os mundos possíveis em um programa ProbLog é igual a um, conforme a equação 2:

$$\sum_{w_i \in \Omega} P(w_i) = 1 \quad (2)$$

Uma query de um átomo q é verdadeira em todos os mundos possíveis w_q se o modelo expresso pelo mundo implicar em q $(w^q \models q)$ (DRIES et al., 2015).

A query $p(a,c)$ realizada para o grafo da figura 1 é verdadeira se existir, pelo menos, um caminho entre os nós a e c. Esta query é verdadeira em 5 dos possíveis 2^3=8 mundos possíveis, como pode ser visto tabela 1 abaixo:

Tabela 1-Mundos possíveis entre a e c

Mundos possíveis	nó(a,b)	nó(a,c)	nó(b,c)	Probabilidades
w_1	T 0.6	T 0.3	T 0.8	0.144
w_2	T 0.6	T 0.3	F 0.2	0.036
w_3	T 0.6	F 0.7	T 0.8	0.336
w_4	T 0.6	F 0.7	F 0.2	0.084
w_5	F 0.4	T 0.3	T 0.8	0.096
w_6	F 0.4	T 0.3	F 0.2	0.024
w_7	F 0.4	F 0.7	T 08	0.224
w_8	F 0.4	F 0.7	F 0.2	0.056
			$\sum$ =	1.000

Fonte: Autor

2.1.1.3 Inferência e aprendizado

A linguagem ProbLog realiza os seguintes tipos de inferência:

a) Probabilidade marginal: sendo L um programa em ProbLog e um conjunto de átomos probabilísticos At, uma query $Q \subseteq At$ é soma de todas as probabilidades, onde a pergunta $q \in Q$ é verdade para um programa em ProbLog $L : P(q)$. No exemplo da tabela 1, é a soma das probabilidades que estão em negrito, $p(a,c)=0,636$.

b) Probabilidade condicional: é o cálculo da probabilidade, dadas as evidências observadas. Evidências são um conjunto de tuplas de átomos e seus respectivos valores verdades observados. Formalmente uma evidência é $E=e$, onde $E \subset At$ é o conjunto de átomos e e é o conjunto de valores verdade observados. A probabilidade condicional de uma pesquisa $q \in Q$ é o cálculo da relação $P(q|E=e)=P(q \wedge E=e)/P(E=e)$. No exemplo do grafo acíclico da tabela 1 para a pesquisa p(a,c) dado como evidência que o *nó(a,c)=falso*, temos: $P(p(a,c)|e(a,c)= false=0.48$.

c) Máximo a posteriori: além do máximo a posteriori, calcula-se também seu caso especial: explicação mais provável. Dado um programa L em ProbLog e um conjunto de átomos At, uma pesquisa Q e evidências $E=e$ em que $\{Q \cup E\} \subset At$, o cálculo de máximo a posteriori é encontrar os átomos probabilísticos mais prováveis para cada átomo probabilístico da pesquisa, em que as evidências estão no estado determinado na pesquisa. O resultado da explicação mais provável é o valor máximo da query $argmax_p\ P(Q = q|E = e)$, onde q é um conjunto dos valores verdade para cada $q \in Q$. No exemplo do grafo acíclico da tabela 1, dada a evidência *nó(a,b)=falso* o estado que demostra a explicação mais provável é w_7 com $P = 0,224$, pois tem a maior probabilidade dentre todos os possíveis mundo dado que *nó(a,b)= falso*.

Como mostrado em Fierens et al. (2013), a primeira etapa para a inferência de um PLP é a conversão do programa em ProbLog para sentenças sem variáveis de acordo com as evidências disponíveis. Isso significa que apenas parte do programa que tem relação, ou com a query que está sendo realizada, ou com as evidências, são convertidas para a forma proposicional.

No algoritmo proposto por Fierens et al. (2013), a conversão de um programa em ProbLog em sentenças sem variáveis é explicado a seguir. Um conjunto de dependência de um átomo At é o conjunto de todos os átomos que ocorrem em alguma evidência de *At*. A dependência de múltiplos átomos é a união dos conjuntos de dependência. Um átomo é

relevante quando tal átomo ocorrer no conjunto de dependência $Q \subseteq At$, e estiver presente tanto na query e como nas evidências.

Uma regra é relevante, explica Fierens et al. (2013), se a regra contém apenas átomos relevantes. Para encontrar tanto os átomos quanto as regras relevantes, usa-se o algoritmo Resolução SLD. SLD é um termo em inglês para *Selection-driven, Linear resolution for definite clauses* que se refere a estratégia de controle usado em PLP's para resolver problemas não determinísticos. O termo resolução refere-se em lógica, como uma ferramenta matemática para provar-se sentenças em lógica de primeira ordem.

De acordo com Fierens et al. (2013), a resolução SLD não é ótima e por isso não utiliza todas as informações disponíveis do programa em ProbLog, entretanto uma alternativa utilizada para otimização é podar as regras inativas da árvore de busca. Regras inativas são as regras que contém átomos que são falsos no conjunto de evidências ou a negação de um átomo que é verdadeiro no conjunto de evidências. Regras inativas não contribuem com a semântica do programa.

Fierens et al. (2013) mostra que o resultado da conversão do programa em PLP é apenas a parte relevante do problema inicial, e que contém apenas as informações necessárias para resolvê-lo, a desvantagem é que todo processo precisa ser refeito caso exista alguma modificação ou nas evidências ou na pesquisa.

A próxima etapa do processo de inferência de acordo com Fierens et al. (2013) é a conversão da parte relevante do programa em fórmula booleana. Uma fórmula booleana é uma forma puramente lógica e não probabilística. Esta conversão não é meramente sintática, as regras devem ser interpretadas de acordo com diferentes semânticas, as regras para PLP sob semântica de Mundo Fechado deve ser a equivalente a semântica utilizada em lógica de primeira ordem sem Mundo Fechado. Para regras acíclicas a conversão é direta, aplica-se a negação por falha seguindo as regras de completamento de Clark. Para exemplificar, é utilizado o exemplo do Alarme:

a) %fatos probabilísticos
   - 0.1:: roubo;
   - 0.2:: terremoto;
   - 0.7:: ouvealarme(X):−pessoa(X).

b) %regras;
   - pessoa(mary);
   - pessoa(jonh);
   - alarme :−roubo;
   - alarme :−terremoto;
   - liga(X):−alarme, ouvealarme(X).

Existem duas regras para "alarme":

a) alarme :−roubo;
b) alarme :−terremoto.

Segundo a regra de completamento de Clark, a forma proposicional para as duas regras acima é:

a) alarme ↔ rouboVterremoto.

A partir da forma proposicional, rescrevesse a fórmula em na forma normal conjuntiva (FNC), abaixo têm-se a conclusão para alarme:

a) alarmeV¬roubo;
b) alarmeV¬terremoto;
c) ¬alarmeVrouboVterremoto.

De acordo com Fierens et al. (2013), para regras cíclicas, negação por falha em conjunto com a completamento de Clark não produz uma fórmula equivalente as regras. Por essa razão, soluções mais sofisticadas são necessárias. Dois algoritmos são utilizados em ProbLog. Ambos algoritmos de conversão utilizam um conjunto de regras e geram uma fórmula equivalente. As fórmulas geradas, serão sintaticamente diferentes porque os algoritmos introduzem átomos auxiliares e regras auxiliares.

A primeira solução utiliza o algoritmo *Answer Set Programming* (ASP) descrito em Janhunen (2004). Este método rescreve as regras até não existir mais ciclos positivos, ao final todos os ciclos estarão negados. Essa técnica utiliza-se de tanto de regras auxiliares quanto de átomos auxiliares. Uma vez que não há mais ciclos positivos, pode-se aplicar a completamento de Clarke por fim rescrever em FNC. Está solução é chamada de baseada em regras.

A segunda solução foi proposta em Mantadelis e Janssens (2010) e é chamada de baseada em provas. Dada uma pesquisa Q e evidências E em que $Q \cup E$, são construídas provas para todos os átomos que são de interesse para $Q \cup E$. O processo de prova é uma estrutura recursiva que terá ciclos enquanto as regras tiverem ciclos. Quando todas os ciclos forem finalizados, então a recursão é finalizada e todas as fórmulas já são fórmulas booleanas. Por fim as fórmulas booleanas já podem ser descritas em FNC.

Para exemplificar é utilizado o modelo do fumante:

a) %fatos probabilísticos;
   - 0.2:: stress(P):−pessoa(P);
   - influencia(P1,P2):−amigo(P1,P2);
   - pessoa(p1);
   - pessoa(p2);
   - pessoa(p3);
   - amigo(p1,p2);
   - amigo(p1,p3);
   - amigo(p2,p1);
   - amigo(p3,p1).

b) %regras;
   - fumante(X):−stress(X) ;
   - fumante(X):−fumante(Y),influencia(Y,X).

Considerando apenas uma parte simplificado do exemplo do fumante acima, a conversão pelo algoritmo baseado em prova gera regras equivalente a conjunções conforme abaixo:

c) fumante(p1)↔ aux1∨stress(p1);

d) fumante(p2)↔ aux2∨stress(p2);

e) aux1↔ fumante(p2)∨influencia(p2,p1);

f) aux2↔ fumante(p1)∨influencia(p1,p2).

No exemplo acima aux_1 e aux_2 são átomos auxiliares utilizados para a conversão. aux_1 diz que a pessoa p_1 começou a fumar uma vez que foi influenciado pela pessoa p_2, que de fato é fumante. Enquanto o exemplo do modelo do fumante é cíclico, no exemplo simplificado após

a aplicação da conversão por prova o ciclo é quebrado. Isso pode ser visto na última sub fórmula que utiliza *stress(p_1)* ao invés de *fumante(p_1)*.

Partindo-se agora de fórmulas booleanas φconstroem-se as fórmulas booleanas ponderadas, que segundo Fierens et al. (2013) é a última etapa de conversão antes da inferência. Seja uma fórmula φ, definida pelas conjunções φ_r e uma fórmula φe que contém as evidências $E=e$. Para as evidências φ_e existe uma sentença a para cada átomo verdadeiro e uma sentença $\neg a$. A ponderação para toda função é derivada do literal probabilístico de cada fato probabilístico. Para todo fato probabilístico $P:: f$, um peso p é adicionado para sua respectiva f, e um peso $1-p$ para $\neg p$. A ponderação de literais não probabilísticos é sempre 1.

Tanto a conversão por regra quanto conversão por prova, retornam um conjunto de fórmulas que são equivalentes às regras de um PL segundo Fierens et al. (2013), e estão de acordo com o lema 1. *SAT(φ_r)* refere-se a um conjunto de modelos da fórmula φ_r, enquanto *MOD(L_g)* denota um conjunto de modelos de um programa L_g em ProbLog.

Lema 1 Sejam L_g um programa em ProbLog e φ_r um conjunto de fórmulas derivadas das regras L_g então *SAT(φ_r)= MOD(L_g)*.

O teorema 1 proposto por Fierens et al. (2013) afirma que fórmulas ponderadas são equivalentes com a conversão das partes relevantes do programa.

Teorema 1 Sejam L_g a parte relevante de um programa em ProbLog de acordo com um pesquisa Q e evidências $E=e$. Seja *MODE=e(L_g)* os modelos em *MOD(l_g)* que são consistentes com $E=e$. Seja φ_r as fórmulas e $w(\cdot)$ os pesos das funções já ponderadas derivadas de L_g. Então:

a) Modelo equivalente: *SAT(ϕ)=MODE=e(Lg)*.
b) Peso equivalente: $\forall\omega \in SAT(\varphi): w(\omega)=P_{Lg}(\omega)$. Por exemplo, o peso ω segundo $w(\cdot)$ é igual a probabilidade de ω segundo L_g.

Uma vez que um programa em lógica probabilística tenha sido convertido em fórmulas booleanas ponderadas, podem ser reformuladas e, portanto, pode-se aplicar os algoritmos do estado-da-arte já existentes para o cálculo de inferência probabilística.

2.2 REDES NEURAIS ARTIFICIAIS

Neste trabalho as RNA serão utilizadas para detectar e segmentar objetos, que pertencentes a uma sequência de quadros como entrada.

A seguir, apresentam-se modelos matemáticos simplificados de neurônios artificiais, como formalizado por Hagan et al. (2014) e como eles podem ser interconectados, formando uma arquitetura de rede.

A notação matemática a ser aplicada é a seguinte:

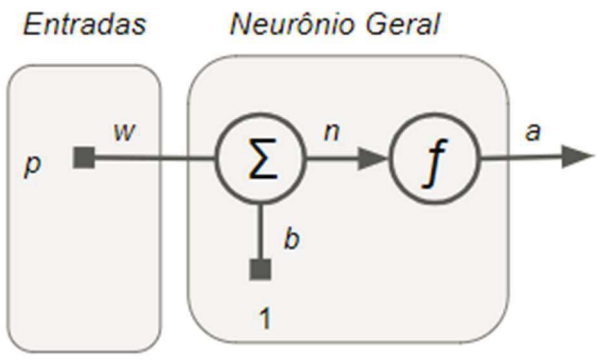
a) Escalar - letras minúsculas em *itálico*: *a,b,c;*
b) Vetores - letras minúsculas em **negrito: a,b,c;**
c) Matrizes - letras maiúsculas em **NEGRITO: A,B,C.**

2.2.1 Neurônios de entrada única

Um neurônio de entrada única pode ser visto na figura 2, onde uma entrada escalar p é multiplicada por um fator de ponderação w formando assim wp e enviado para a somatória. A outra entrada, de valor 1, é multiplicada por bias b e enviado a somatória. O resultado da somatória é denominado entrada da rede e vai para uma função de transferência f que produz um escalar a como saída do neurônio, representado pela equação 3

$$a = f(wp + b) \tag{3}$$

Figura 2-Neurônio de entrada única



Fonte: Autor

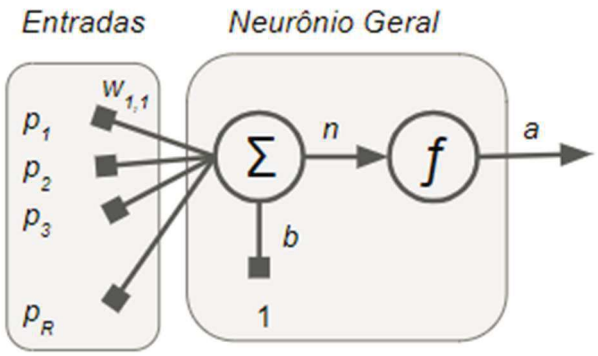
Os escalares w e b são parâmetros ajustáveis pela função de transferência escolhida pelo responsável pela rede neural. Com isso, são ajustados por alguma regra de aprendizagem para

que as relações de entrada e saída do neurônio atinjam alguma meta específica. Escolhem-se as funções de transferência para satisfazer alguma especificação do problema que o neurônio está tentando resolver. A importância das funções de ativação é que elas trazem um componente não linear para as RNA, o que leva a pensar na possibilidade de que aprendam mais do que relações lineares das variáveis, sejam elas dependentes ou independentes.

2.2.1 Neurônios de múltiplas entradas

Normalmente um neurônio tem mais de uma entrada, portanto um neurônio com *R* entradas, como na figura 3, tem suas entradas individuais $p_1,p_2,...,p_R$ ponderadas pelos respectivos elementos $w_1,w_2,...,w_R$ da matriz do fator de ponderação **W**. Soma-se o bias com as entradas já ponderadas para formar a entrada da rede n, como visto na equação 4.

Figura 3-Neurônio de múltiplas entradas



Fonte: Autor

$$n = w1,1p1 + w1,2p2 + \ldots + w1,RpR + b \quad (4)$$

A equação 3.4 pode ser descrita na forma matricial, de acordo com a equação 5:

$$n = \boldsymbol{Wp} + b \quad (5)$$

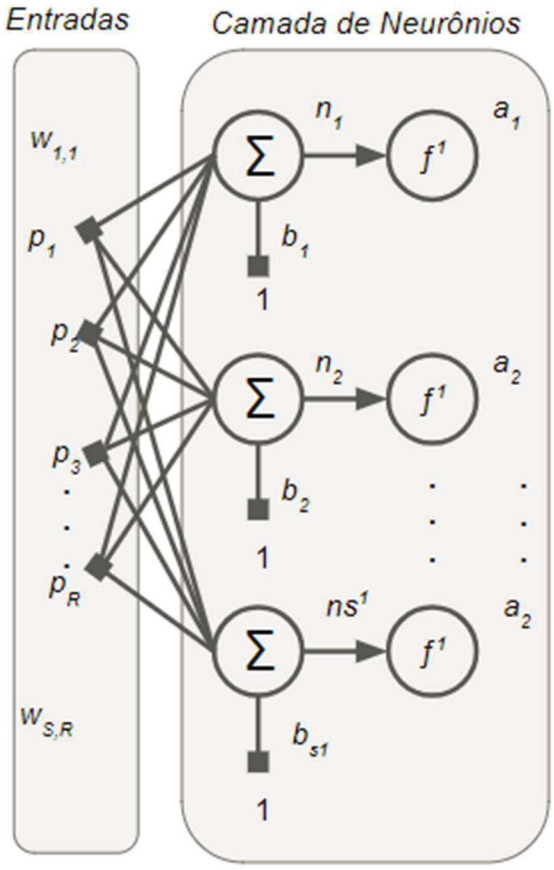
Dessa maneira um neurônio pode ser definido de acordo com a equação 6:

$$a = f(\boldsymbol{Wp} + b) \quad (6)$$

2.2.3 Neurônios de uma camada

Uma rede com apenas uma camada de neurônios S, como mostrado na figura 4, recebe cada entrada *p* que está conectada com cada neurônio através da matriz do fator de ponderação e recebe um bias na somatória, que passa seu resultado para a função de transferência para gerar a saída.

Figura 4 - Rede com uma camada de neurônios



Fonte: Autor

A matriz do fator de ponderação W é representada pela equação 7:

$$a = f(Wp + b) \tag{7}$$

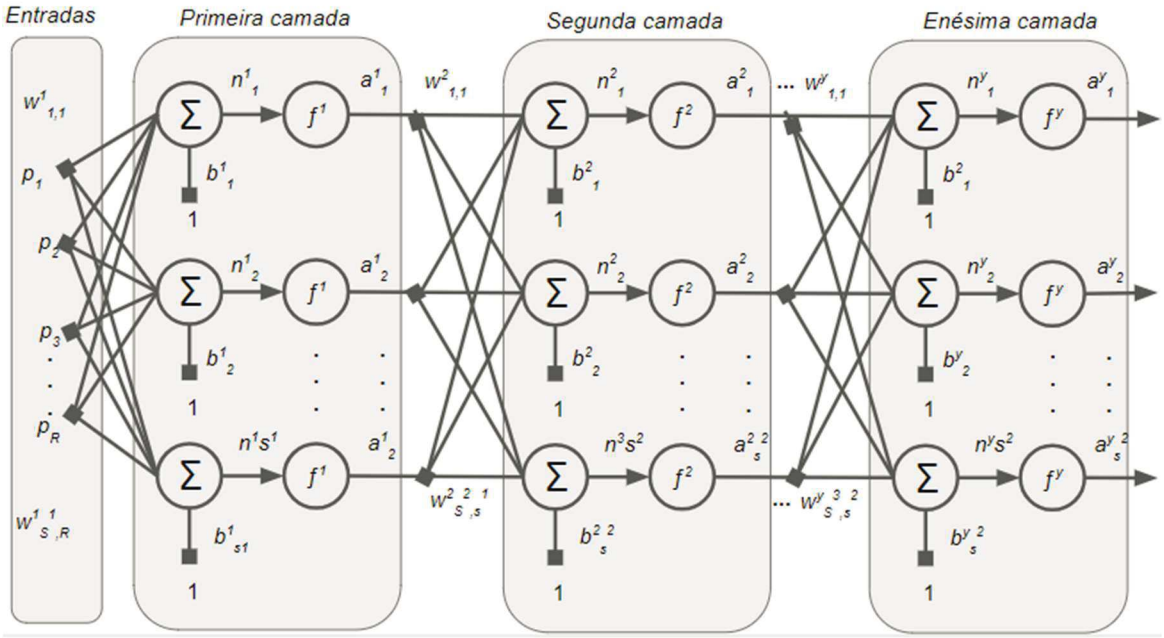
Cada linha da matriz **W**, de acordo com a equação 8 indica o neurônio destino ao qual os fatores de ponderação estão associados, enquanto as colunas indicam a fonte da entrada de cada fator de ponderação.

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{8}$$

2.2.4 Neurônios de múltiplas camadas

Em uma rede com vários neurônios, como visto na figura 5, cada camada tem sua própria matriz de ponderação **W**, seus próprios vetores de erro **b**, de entrada *n* e uma saída *a*. Para definir todos esses elementos de cada camada, inclui-se um índice sobrescrito em cada termo. A camada de saída é a própria saída da rede. Todas as outras camadas são denominadas de camadas ocultas. Redes neurais multicamadas tem mais capacidade de resolver problemas que as redes simples. Por exemplo, uma rede de duas camadas tendo uma log-sigmoide na primeira camada e uma linear na segunda pode ser treinada para aproximar funções, porém as de camada única não.

Figura 5-Rede com múltiplas camadas de neurônios

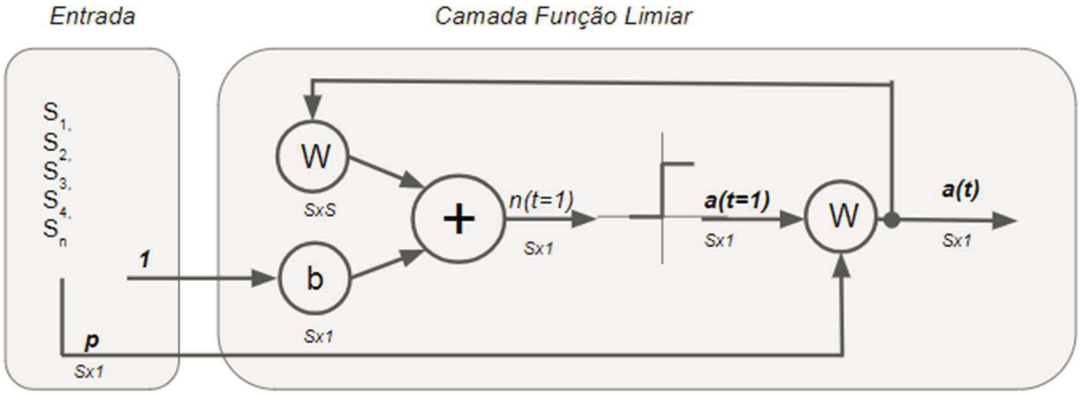


Fonte: Autor

2.2.5 Neurônios de recorrentes

As redes neurais recorrentes são redes com realimentação de informações da saída diretamente para a matriz **W**, ou seja, suas saídas estão conectadas às entradas. Pode se visualizar na figura 6 um exemplo de rede recorrente discreta em relação ao tempo. Redes recorrentes são poderosas se comparadas com as não recorrentes, especialmente pelo motivo de as saídas futuras serem computadas, levando-se em conta as saídas de períodos anteriores.

Figura 6-Rede recorrente



Fonte: Autor

3.2.6 Redes neurais de alimentação direta

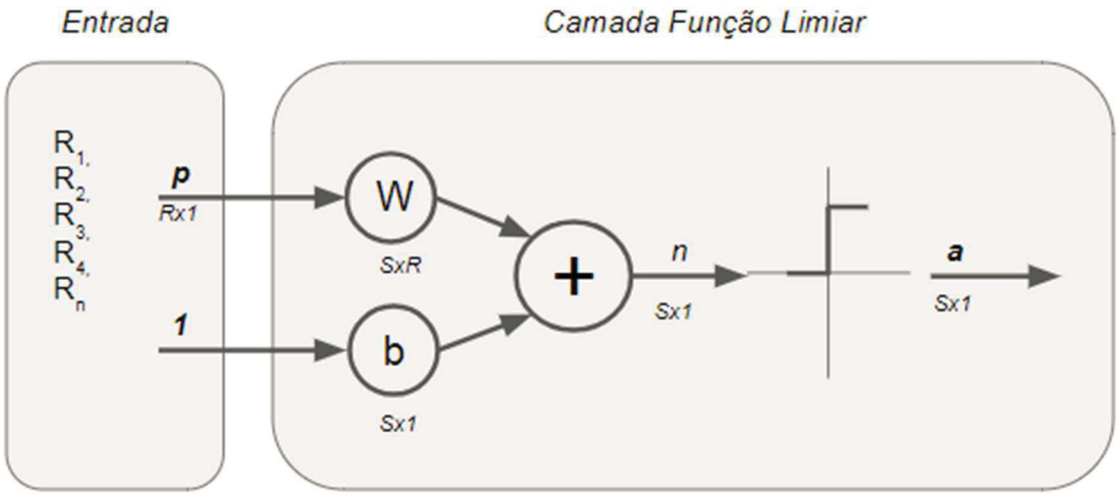
As técnicas modernas utilizadas em aprendizado de máquina são uma estrutura poderosa para o aprendizado supervisionado de acordo com Goodfellow, Bengio e Courville (2016). Uma RNA pode aumentar sua complexidade adicionando-se mais camadas ou mesmo mais entradas em cada camada.

Nesta seção, mostra-se o modelo básico de uma RNA, segundo Hagan et al. (2014). Este modelo básico é chamado de Perceptron.

Perceptron ou Redes Neurais de alimentação direta, como mostrado na figura 7, tem sua saída definida pela equação 9.

$$a = Limiar(Wp + b) \tag{9}$$

Figura 7-Perceptron



Fonte: Autor

Tendo-se em vista a seguinte matriz de ponderação da equação 10:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{10}$$

Considere um vetor composto pelo enésimo elemento da coluna de **W**, conforme equação 11:

$${}_i\mathbf{W} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \tag{11}$$

Então, pode-se particionar o vetor de ponderação conforme a equação 12:

$$ {}_i\mathbf{W} = \begin{bmatrix} {}_1\mathbf{W}^T \\ {}_2\mathbf{W}^T \\ \vdots \\ {}_S\mathbf{W}^T \end{bmatrix} \tag{12} $$

A partir dessas equações, pode-se escrever o enésimo elemento do vetor de saída da rede, como mostrado na equação 13:

$$ \mathrm{a}_1 = Limiar(\ {}_{\mathrm{i}}\mathbf{W}^{\mathrm{T}}\mathbf{p} + \mathrm{b}_{\mathrm{i}}) \tag{13} $$

As redes neurais de alimentação direta têm sua saída computada diretamente e as informações passam apenas uma vez por cada neurônio da rede; não existe retroalimentação. Esse tipo de estrutura é comumente utilizado para reconhecimento de padrões, para aproximação de funções que serão usadas em filtros adaptativos e em controles automáticos. Na próxima seção será apresentado um caso especial de redes neurais de alimentação direta, chamadas de redes neurais convolucionais. As redes neurais convolucionais atualmente, como mostrado mais a frente no capítulo 3, tem se mostrado eficiente para o processamento de imagens quando é necessário detectar e classificar objetos em imagens.

2.2.7 Redes Neurais Convolucionais

As RNCs são um caso especial das redes neurais de alimentação direta. De acordo com Goodfellow, Bengio e Courville (2016), são simplesmente redes neurais que utilizam convolução ao invés de multiplicação de matrizes em, pelo menos, uma de suas camadas. São utilizadas, principalmente, para processamento de dados com topologia do tipo grade, como imagens, por exemplo, que têm duas dimensões. As redes convolucionais obtiveram bastante êxito em aplicações práticas, principalmente com imagens. O nome convolução refere-se à operação matemática linear entre duas funções com valores reais e definidos na equação 14:

$$ s(t) = (x\ *w\ )(t) \tag{14} $$

O argumento w da função 14 deve ser uma função válida de probabilidade densa, senão a saída não será uma média ponderada. Deve também ser 0 na soma para todos argumentos negativos. A função x da equação 14 é a entrada e *w*, o kernel. Sua saída é denominada de mapa de característica. No caso de imagens, o tempo é discreto portanto, pode-se assumir que *t* é um valor inteiro e que tanto *x* quanto *w* estão restritos ao inteiro *t.* Com isso é possível definir uma convolução discreta na equação 15.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} (a)w(t-a) \tag{15}$$

Para imagens, utiliza-se convolução em mais de um eixo por vez. Neste caso emprega-se um kernel de duas dimensões, como mostrado na equação 16:

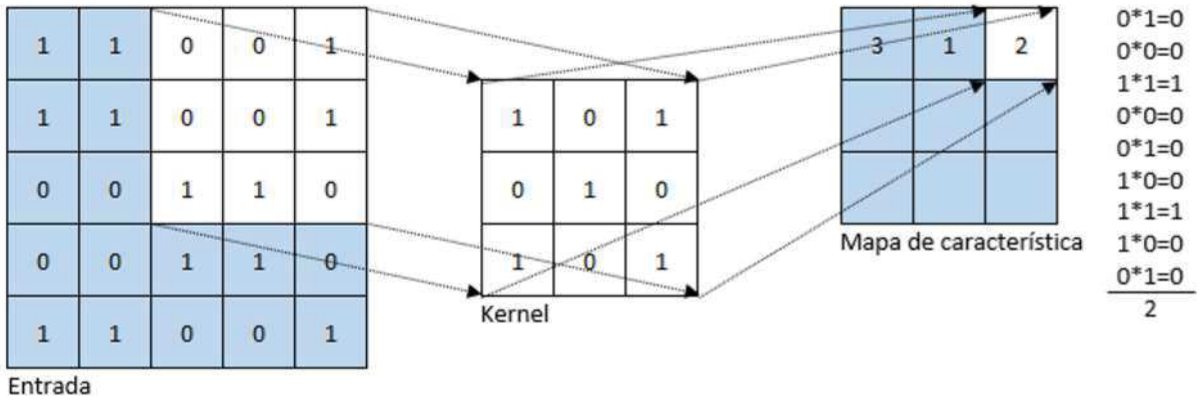
$$s(i,j) = (K * I)\big((i,j)\big) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n) \tag{16}$$

Uma convolução é comutativa, por isso pode ser definida também como visto na equação 17:

$$s(i,j) = (I * K)\big((i,j)\big) = \sum_{m}\sum_{n} I(i-m, j-n)K(m,n) \tag{17}$$

No exemplo da figura 8, há uma imagem binária de 5x5 utilizada como entrada I para a convolução, e um kernel K 3x3. O resultado no um mapa de característica é o valor 2, como pode ser visto na soma dos produtos I por K.

Figura 8-Exemplo criação de um mapa de característica



Fonte: Autor

A convolução traz três importantes ideias que ajudam a melhorar o sistema de aprendizado de máquina:

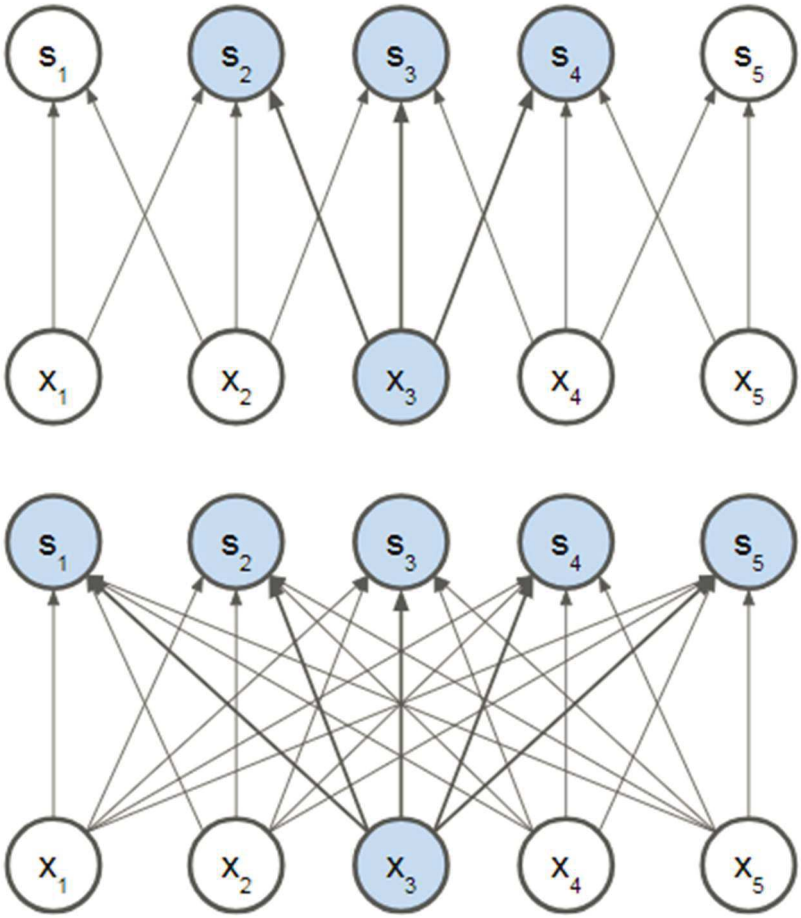
a) Interações esparsas: em RNAs tradicionais, multiplica-se uma matriz por outra matriz, ou seja, toda saída interage com toda entrada. Nas RNCs, utiliza-se um kernel muito menor do que o tamanho da entrada, o que significa uma grande redução de memória e que a computação da saída requer menos operações. No exemplo da figura 9, uma entrada x3 e as saídas s, que estão em destaque, mostram que utilizando um kernel de tamanho 3, apenas três saídas são impactadas por x, como mostrado na parte superior da figura 9. Por outro lado, quando a saída s é resultado de uma multiplicação de matrizes, a conectividade não é mais esparsa e, portanto, todas as saídas são afetadas por uma única entrada x3, como mostrado na parte inferior da figura 9.

b) Compartilhamento de parâmetros: utiliza cada membro do kernel em todas as posições dos dados de entrada e faz com que, em vez de aprender parâmetros separados para cada posição, apenas aprenda um conjunto. Isso não afeta o tempo de processamento da propagação direta, porém reduz muito a necessidade de memória para um modelo que utiliza, por exemplo, valores de kernel k de 3x3, 5x5 e 7x7, se comparado com as dimensões de uma imagem inteira. A figura 10 mostra as conexões com setas em negrito. Na parte superior, indica o uso de kernel de três elementos em um modelo convolucional. Na parte de baixo, indica o uso de uma matriz ponderada totalmente conectada. Nas RNAs que utilizam esse método, cada entrada tem sua própria matriz de ponderação, ou seja, neste caso não há

compartilhamento de parâmetros como o kernel compartilha em redes convolucionais.

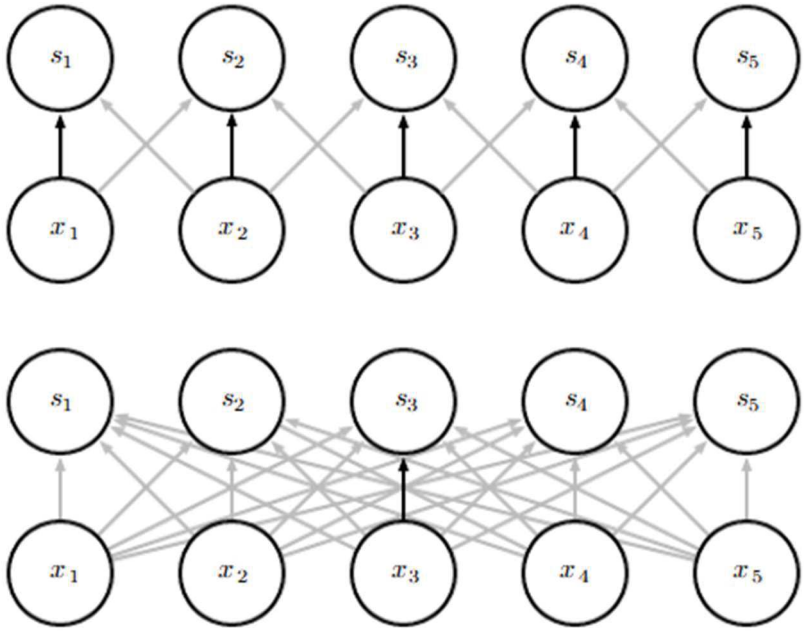
c) Representações equivariantes: a forma particular com que as convoluções compartilham parâmetros faz com que a camada tenha uma propriedade equivariante em relação à translação da entrada. Independentemente de onde o objeto esteja em uma imagem, a capacidade de identificar suas bordas será a mesma.

Figura 9-Conectividade esparsa



Fonte: Autor

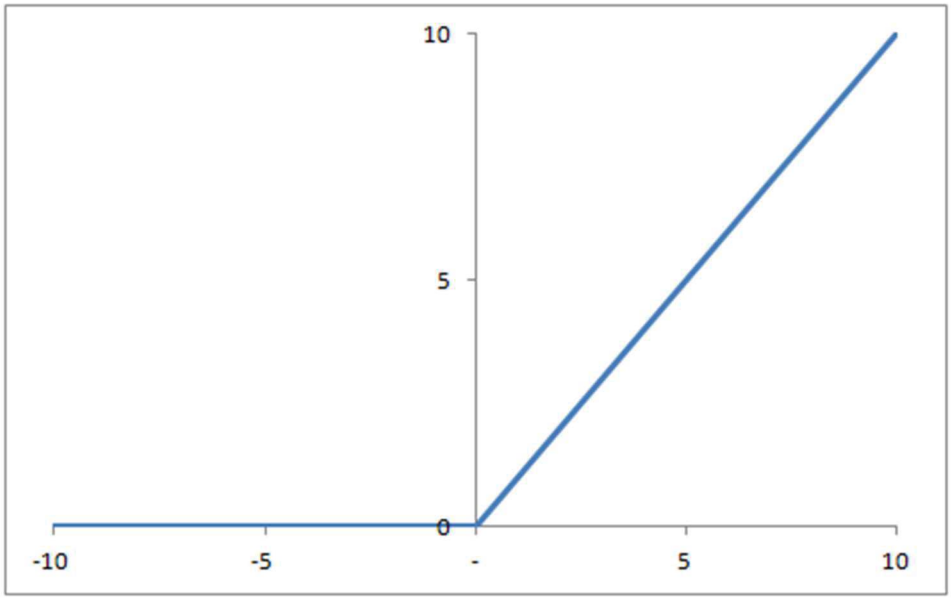
Figura 10-Compartilhamento de parâmetros



Fonte: Autor

Uma típica camada de uma RNC tem três estágios (GOODFELLOW; BENGIO; COURVILLE,2016). No primeiro estágio, as camadas realizam várias convoluções em paralelo para produzir um conjunto de ativações lineares. No segundo estágio, cada ativação linear é processada por uma função de ativação não linear, para RNCs a mais utilizada é a função de ativação linear retificada (ReLu). A figura 11 mostra o gráfico da função f(x)= max(0,x), que tem saída 0 sempre que a entrada for menor que zero, e tem saída igual à entrada para todos os valores maiores que 0.

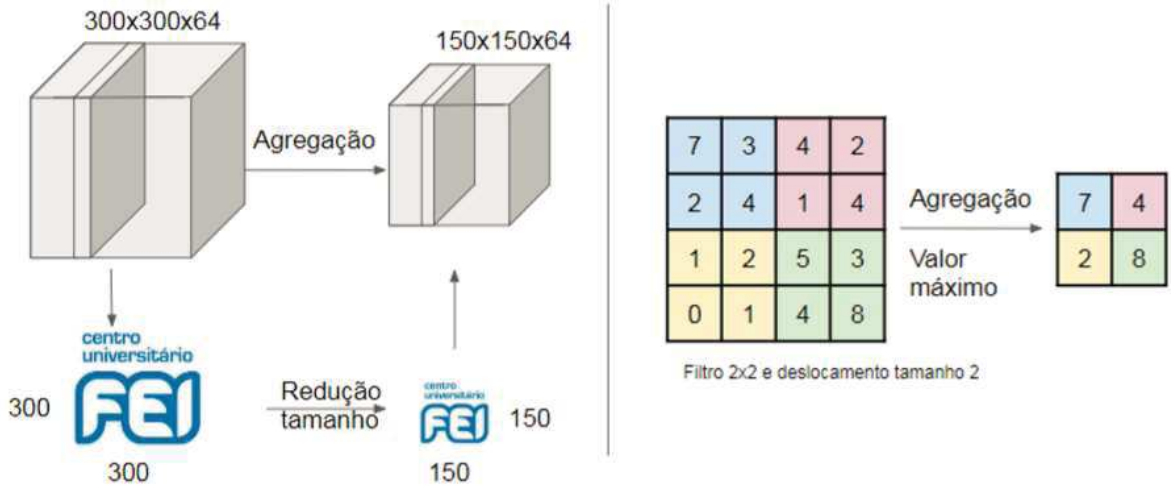
Figura 11-Função ReLu



Fonte: Autor

No terceiro estágio, tem-se uma camada com a função de agregação que modifica a saída da camada. Esta função substitui a saída da rede em certas regiões com uma síntese estatística das saídas próximas. É comum inserir camadas de agregação entre camadas sucessivas de convolução, pois elas reduzem o tamanho espacial da representação do problema por reduzir a quantidade de parâmetros e computação na rede. A forma mais comum de agregação é uma filtragem de tamanho 2x2 com passo 2. Esse processo reduz em 75% o tamanho do problema. Na figura 12, pode-se checar que a função de agregação reduz o tamanho espacial do problema, o tamanho da entrada é 300x300x64. Aplicando-se uma função de agregação com filtro tamanho 2x2, e com passo 2, o tamanho é reduzido para 150x150x64. A operação mais comum para função de agregação é o máximo valor, que copia o máximo valor da dentro do filtro de 2x2 como mostrado no exemplo da figura 12.

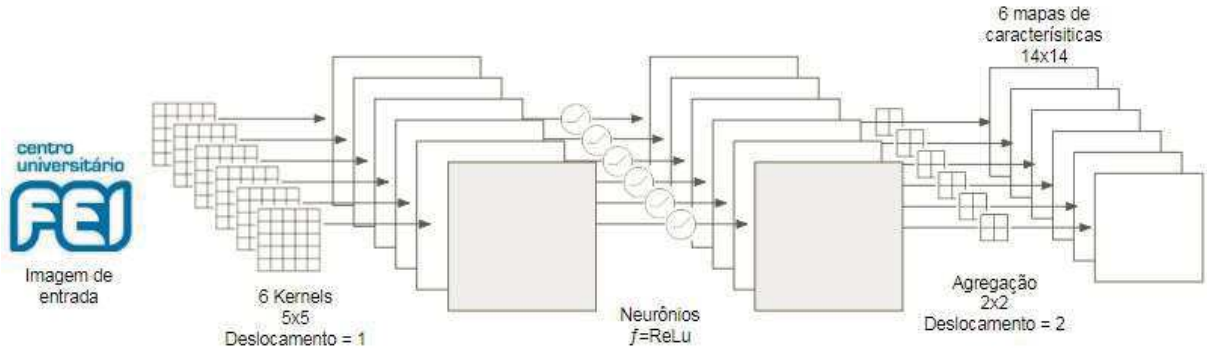
Figura 12-Função de agregação



Fonte: Autor

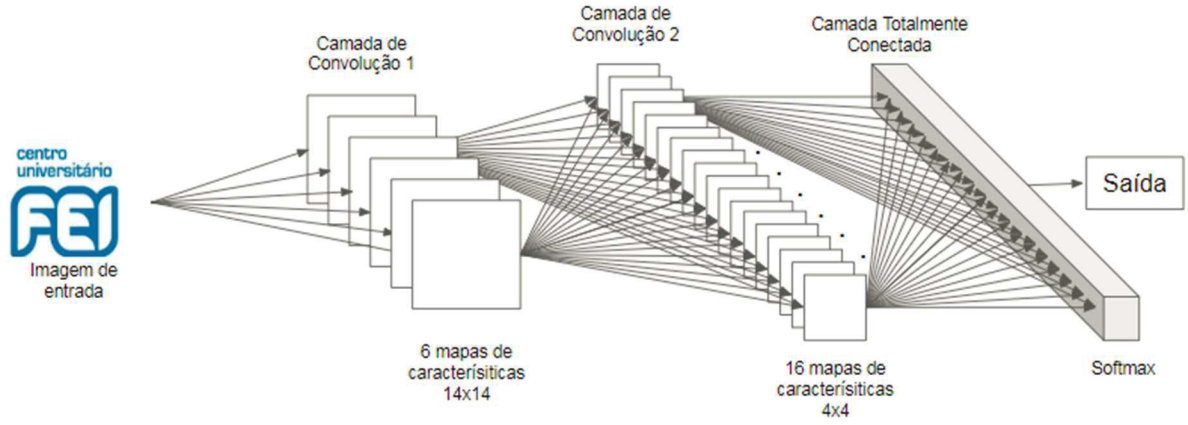
A forma mais comum de arquitetura de uma RNA é combinação de algumas camadas de convolução com função de ativação ReLu, seguidas por camadas de agregação, como pode ser visto no exemplo da figura 13, e repetem esse padrão até que a imagem possa ser fundida espacialmente a um pequeno tamanho. A última camada é totalmente conectada e corresponde à saída da RNC, representado como exemplo pela figura 14.

Figura 13-Camada de convolução



Fonte: Autor

Figura 14-Rede Neural Convolucional



Fonte: Autor

Uma RNC pode ser usada para gerar na sua saída um objeto estruturado de grande dimensão ao invés de apenas fazer a predição de uma classe para tarefas de classificação, ou mesmo um valor real para uma atividade de regressão. Normalmente, a saída é um tensor S, onde $S_{i,j,k}$ é a probabilidade que cada pixel (j,k) de uma entrada pertencer a uma classe i. Isso permite que o modelo classifique cada pixel em uma imagem e crie uma máscara que permite acompanhar os contornos dos objetos.

Neste capítulo foram apresentadas as teorias de redes neurais e lógica probabilística que serão necessárias para a base teórica da proposta deste trabalho. No próximo capítulo é apresentada a revisão bibliográfica deste trabalho, que mostra o estado da arte das principais técnicas que serão aplicadas: redes neurais e a lógica probabilística. No início do capítulo está a revisão dos sistemas ADAS, que tem como objetivo entender quais técnicas e métodos o estado da arte está utilizando.

3 REVISÃO BIBLIOGRÁFICA

Nas áreas de raciocínio lógico e de aprendizado de máquina, existem diversos trabalhos que mostram técnicas utilizadas para inferência, localização, classificação e reconhecimento de padrões em imagens e vídeos. Essas técnicas serão abordadas nas próximas seções.

3.1 LÓGICA PROBABILÍSTICA

Com o objetivo de interpretar faixas de trânsito, Souza e Santos (2011) desenvolveram um modelo de lógica probabilística que analisa os quadros de um vídeo captado do ponto de vista do veículo por meio de um sistema de visão de baixo nível baseando-se na transformada de Hough. Para realizar a inferência, uma Rede Lógica de Markov (MLN) disponibiliza a interpretação gerada pela saída da classificação, empregando uma base de conhecimento em lógica de primeira ordem, que contém as regras de trânsito e a representação do domínio.

O reconhecimento de eventos a partir de vídeos é importante para detecção de anormalidades no seu contexto. Técnicas que utilizam programação em lógica indutiva (ILP) conseguem aprender modelos, a partir de eventos devido à sua relação de espaço-tempo. No trabalho proposto por Dubba et al. (2015), é mostrado que para base de dados pequenas, ILP consegue ter sucesso na tarefa de aprender o modelo dos eventos. Para base de dados maiores é proposta uma abordagem diferente, é apresentado um método de aprendizado relacional supervisionado dos eventos para gerar o modelo.

Cohn et al. (2006) propuseram um sistema de visão cognitiva que combina visão computacional e representação simbólica qualitativa. A representação simbólica dos dados foi feita utilizando-se ILP e permite que a generalização de conceitos como equivalência e reordenamento. A aprendizagem do modelo, neste trabalho, tem como objetivo capturar o comportamento de eventos contidos em uma sequência de quadros, através da sua relação espacial qualitativa entre os objetos que fazem parte da cena.

Neste trabalho, a representação do domínio de tráfego de veículos e suas regras, necessitam de um método que consiga representar, através de um modelo probabilístico, as incertezas e correlações existentes no domínio. Neste sentido, a lógica probabilística utilizando-se da programação em lógica através do ProbLog, é apropriada para modelar e interpretar as incertezas presentes no domínio de tráfego de veículos e inferir prováveis situações de risco.

3.2 SISTEMAS DE ASSISTÊNCIA AO MOTORISTA

Classificam-se os sistemas de segurança veicular, segundo Kukkala et al. (2018), em dois tipos: passivos e ativos. Os sistemas passivos de segurança têm como objetivo proteger os ocupantes dos veículos depois de uma colisão. Alguns exemplos são os cintos de segurança, os air bags, o painel de instrumentos acolchoados, entre outros. Os sistemas de segurança ativos, por outro lado, têm como principal objetivo prevenir as colisões. Denominam-se esses sistemas de Sistemas Avançados de Assistência ao Motorista (ADAS).

A predição automática de risco de trânsito é um dos principais interesses dos ADAS. Em Inoue, Kuriya e Kobayashi Sosuke (2015) propõem-se um novo modelo de predição de risco utilizando raciocínio com inferência pela melhor explicação em lógica de primeira ordem. Para melhorar a inferência, empregou-se um algoritmo de aprendizado de máquina visto em Wang, Zhao e Hoi (2012) cujo objetivo é definir um peso para cada hipótese e, com isso, melhorar a robustez da predição do modelo. Fazendo-se uso de potenciais riscos já estabelecidos pelo centro de educação da Toyota e de outros materiais coletados de lições de trânsito, propuseram-se 93 riscos para a base de conhecimento.

Uma revisão feita sobre o estágio atual do ADAS, Bengler et al. (2014) avaliaram que, atualmente, os sensores presentes nos carros são capazes de receber informações detalhadas sobre o meio onde está o veículo. Entretanto, a avaliação de percepção e os alertas de situação realizados pelos controladores dos carros ainda estão iniciando sua aplicação. Por essa razão, afirmam que é necessário haver progresso nas áreas de classificação de cenas como, por exemplo: reconhecimento de objetos em condições dinâmicas, compreensão contextual de cenas, inferência sobre a relação entre objetos dinâmicos e a infraestrutura de trânsito. Com o desenvolvimento da área de classificação de cenas, espera-se que se estabeleça no mercado automotivo a utilização de soluções com Inteligência Artificial. A predição probabilística de comportamentos futuros, tanto dos veículos como de todos os objetos que fazem parte da cena, deve ser levada em conta por meio de inferência da intenção dos objetos nela presentes e da modelagem do comportamento desses objetos.

Sistemas de alerta de colisão são uma das funções do ADAS e um ponto crítico para segurança veicular, Pyo, Bang e Jeong (2016) propuseram um sistema de alerta de colisão que se utiliza de uma câmera monocular para capturar uma sequência de quadros da cena de tráfego em estradas. Uma RNC é alimentada com os quadros capturados pela câmera e utiliza sua saída como um classificador e detector de veículos da cena capturada. Propõe-se um sistema adaptativo de determinação da região de interesse dos quadros da cena em questão, utilizando-

se da detecção das faixas de rolamento através da transformada de Hough (DUDA; HART, 1972). Que usa a convergência das linhas detectadas como região de interesse válida. Para o cálculo do tempo de colisão, utiliza-se apenas o veículo, identificado pela RNC, e que está dentro da região de interesse do quadro da cena. Calcula-se a distância através da relação geométrica da largura do veículo detectado, comparada com a largura entre as faixas de rolagem identificadas.

Os sistemas ADAS baseados em câmera, como explica Kukkala et al. (2018), podem ser utilizados para diversos tipos de detecção como, por exemplo: automóveis, pedestres, faixas de rolamento e sinais de trânsito. A câmera, como um sensor de aquisição de informações, consegue também capturar informações como: cor, contraste, textura etc. Com o uso de câmeras para os sistemas ADAS, foi possível implementar as mais avançadas técnicas de aprendizado de máquina e de computação visual. Atualmente, os tipos de detecção já mencionadas acima utilizam-se de RNC como ferramenta.

Uma variedade muito grande de sistemas ADAS, sejam eles para prevenir colisões, piloto automático entre outras atividades, pode-se utilizar do rastreamento de veículos e essa tarefa é muito importante em termos de dados, para principalmente estimar a posição dos outros veículos como abordado por (Schubert; Richter; Wanielik, 2008).

Modelos de movimentos que estimam a posição dos veículos utilizam-se comumente de filtros de Kalman e suas variantes de acordo com Schubert, Richter e Wanielik (2008), o mesmo trabalho mostra que adicionando dados de GPS e do odômetro do veículo que está sendo testado (ego veículo) no modelo foi possível melhorar o resultado do rastreamento dos veículos ao redor. Schubert, Richter e Wanielik (2008) demonstra também que variações no modelo matemático de predição de movimento também influenciam no resultado. Foram testados modelos com aceleração constante, velocidade constante, velocidade de esterçamento constante, aceleração de esterçamento constante e suas combinações e derivações.

Utilizando-se de duas câmeras e o algoritmo *Semi global Matching*, Feth et al. (2018) criou uma imagem estéreo e com isso pode calcular distâncias na imagem estéreo. Sistemas de percepção de risco, que tem como objetivo identificar e mitigar situações de colisão a segundo norma ISO para a indústria automotiva, utilizam métricas que fazem uso das distâncias para cálculo de um tempo estimado de colisão. Através dessas métricas em conjunto com as imagens estéreas foi criado um mapa de risco, que simula a cena. Diversos mapas com diferentes situações de risco foram anotados por um especialista e uma RNC foi treinada para identificar os mapas com risco. Feth et al. (2018) conclui que este tipo de avaliação de risco pode ser utilizado com um sistema de redundância em sistema ADAS.

Com o aumento das tecnologias de comunicação, de acordo com Prabu (2015) já existem solução como tecnologias que trocam informações entre os automóveis e com os sistemas de informação de tráfego. Essas tecnologias tornam possíveis sistemas ADAS com o objetivo de alerta de colisão utilizando-se informações disponibilizadas pelos veículos e dos sistemas de tráfego, como por exemplo em cruzamentos onde não há contato visual entre os veículos e obstáculos físicos como construções que impedem o uso de sensores laser. Prabu (2015) cita que uma nova proposta de sistemas ADAS tem se tornado uma tendência nas pesquisas atuais que levam em consideração as trocas de informação entre veículos e com o ambiente.

Um sistema de prevenção de colisão traseira em estradas é proposto em Nakamori et al. (2002), em que uma câmera foi acoplada no painel do veículo. A imagem adquirida é transposta para uma projeção superior e, com isso foi possível determinar a velocidade do ego veículo através da variação das distâncias das linhas de tracejadas que limitam as faixas de rolagem, e as distâncias para os veículos a frente. A geração do alerta de colisão e do nível do alerta é determinada por uma curva distância por velocidade proposta pelo autor.

Utilizando-se a comunicação entre veículos, Hafner et al. (2013) desenvolveu um sistema ADAS que utiliza um algoritmo com filtro de Kalman para evitar colisão em cruzamentos. O modelo que é a entrada do filtro de Kalman leva em conta os parâmetros do carro e do condutor, mas principalmente adiciona a comunicação entre os veículos. A adição desta comunicação permite que se avalie a incerteza entre a comunicação e o atraso na comunicação para estimar o risco de colisão.

Como os fabricantes automotivos têm cada vez mais incluído sistemas de auxílio ao motorista, novos sistemas ativos que mitiguem os riscos devem ser desenvolvidos e melhorados. Como visto no estado da arte, os sistemas ADAS utilizam fusão de dados, como por exemplo a posição e velocidade, imagens das câmeras e outras informações dos diversos sensores atualmente já embarcados nos veículos para a análise do risco. Esta fusão de dados, como mostrado na revisão do estado da arte anteriormente neste capítulo, mostra que a fusão de informações é atualmente muito pesquisada, em especial porque é possível ter várias informações do mesmo contexto, e analisar os dados de acordo com o contexto para avaliar se algum desses dados podem ser ruidosos e atrapalhar na análise do risco. Pode-se utilizar também a fusão para que seja criado sistemas alternativos de mitigação de riscos caso o sistema principal falhe ou não identifique um risco real.

Levando-se em conta que cada vez mais os veículos vêm equipados com câmeras, a proposta deste trabalho é um sistema ADAS ativo de auxílio ao motorista que o alerte em caso

de um possível risco de colisão, e está em linha com o que se vem pesquisando em ADAS. Este trabalho utiliza fusão de dados entre a câmera embarcada no veículo em conjunto com uma base de dados que descreve regras de boa conduta no trânsito. Como mostrado na revisão feita por Bengler et al. (2014), ainda é necessário que a os sistemas ADAS incluam comportamentos futuros dos objetos de uma cena, e que sua análise seja realizada por predição probabilística. Por essa razão este trabalho implementa um sistema capaz de realizar predição probabilística através da fusão de informação tanto câmera quanto das regras que compõe a base de dados de boa conduta no trânsito. Diferentemente de Inoue, Kuriya e Kobayashi Sosuke (2015), que utilizou a lógica de primeira ordem para fazer a predição do risco, este trabalho representou o domínio em ProbLog principalmente por permitir a declaração e codificação de regras em fatos probabilísticos, e com isso conseguir representar sentenças complexas e atribuir a elas probabilidades. Neste contexto de fusão de dados, os quadros capturados pela câmera serão analisados por uma RNC, a próxima seção apresenta o estado da arte das RNC's.

3.3 REDES NEURAIS CONVOLUCIONAIS

Krizhevsky, Sutskever e Hinton (2012) propuseram uma RNC que consiste em cinco camadas convolucionais. Dentre elas, existem algumas camadas de agrupamento máximo e três camadas totalmente conectadas. Optou-se pela RNC em vez de redes neurais de alimentação direta, sem retroalimentação, com camadas de tamanhos similares porque as RNCs podem utilizar menos conexões e parâmetros, além de serem mais facilmente treinadas com pequena perda de performance. A contribuição dessa proposta se dá no sentido de indicar uma maneira eficiente e otimizada de realizar a convolução das imagens em 2D, utilizando-se de duas unidades de processamento gráfico (UPG). Além disso, a comunicação entre as UPG não é realizada em todas as camadas. Outra contribuição foi o emprego de modelagem das saídas dos neurônios (saturação não linear) de forma não tradicional. Utilizaram-se as funções retificadas lineares (FRL), que têm a propriedade de prevenir que o processo de aprendizado continue quando um exemplo já gerou uma entrada positiva, sem necessidade de normalização. Jarret et. al (2009) também mencionaram alternativas aos modelos tradicionais.

No trabalho proposto por Szegedy et al. (2014) para participar do *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) de 2014, melhorou-se, sem que houvesse acréscimo computacional, uma RNC chamada de *Inception*. Ainda que o estado da arte atual, na visão computacional, faça uso de camadas totalmente conectadas, introduziram-se camadas esparsas, inclusive dentro das camadas de convolução, para manter a computação inalterada. O ganho

introduzido pelas camadas esparsas acontece se a distribuição de probabilidades do conjunto de dados for representada por rede neural esparsa. Dessa maneira, pode-se construir uma topologia de rede ótima, camada a camada, analisando-se as correlações estatísticas das ativações dos neurônios em camadas anteriores e agregando-se os neurônios com alta correlação na saída. Segundo Szegedy et al. (2014) esta afirmação se aproxima do Princípio de Hebbian: “Neurônios que disparam juntos permanecem conectados”.

No trabalho apresentado por Simonyan e Zisserman (2014b) para a competição de ILSVRC de 2015, propôs-se uma RNC com 19 camadas, o que segue o mesmo princípio de aumentar a profundidade da RNC assim como em Szegedy et. al (2015). Entretanto, o resultado foi melhor no que se refere à localização de objetos. Diferentemente de Szegedy et al. (2014), Simonyan e Zisserman (2014b) usaram uma estrutura de rede mais simples.

O gargalo computacional atual da detecção de objetos através de RNC, segundo Ren et al. (2015) é a região, dentro da imagem, onde a RNC irá analisar a detecção de objetos. Por esta razão, Ren et al. (2015) propuseram uma RNC chamada de "Faster R-CNN". Para a geração de regiões, treina-se uma RNC totalmente conectada para detectar os limites dos objetos e uma pontuação do objeto para cada posição. A saída da camada convolucional gera um mapa de características usadas tanto para gerar as regiões propostas de localização dos objetos, quanto como entrada para uma outra RNC para a atividade de detecção. Por essa razão, as propostas de regiões têm custo computacional quase zero. Obtém-se esse custo baixo de computação pelo treinamento das redes. Primeiro, treina-se a RNC com a ImageNet para a tarefa de gerar as propostas de regiões e, depois, utilizando-se das propostas geradas pelo treinamento, treina-se novamente a RNC. Após essa etapa, as camadas de convolução ainda não estão conectadas; o último passo consiste em fazer a conexão entre as redes. Essa proposta de detecção de objetos consegue ser eficiente com uma taxa de 5-17 quadros por segundo.

Taylor et al. (2010) desenvolveram um método capaz de aprender representações de uma sequência de pares de imagens sucessivas usando fluxo ótico como entrada para a RNC, o que possibilita extrair características de movimento, segmentação e detecção de bordas. Um diferencial em relação a outros métodos propostos durante seu estudo foi a introdução de probabilidades na camada que realiza o agrupamento pelo valor máximo.

Combinando RNC com o algoritmo de análise independente do subespaço (ISA) Le et al. (2011) mostraram que é possível identificar variações temporais em vídeos. Nas primeiras camadas da RNC, o modelo aprende características que detectam movimentos de borda. Por meio da variação da velocidade dos objetos na imagem com a aplicação do ISA, obtêm-se

neurônios altamente sensíveis à variação da velocidade. Essa propriedade faz com que a referida abordagem seja capaz de reconhecer movimentos.

Como descrito anteriormente, o uso da RNC apresentou grandes resultados na classificação de imagens. Isso motivou Karpathy et al. (2014) a avaliar o emprego de RNCs para classificar vídeos e compará-las com as melhores práticas de classificação de algoritmos baseados em características locais. Os resultados mostraram melhoria na performance do uso de RNCs. Karpathy et al. (2014) apresentaram também um método que aplica uma arquitetura com duas RNCs. Elas processam separadamente a entrada em duas diferentes resoluções espaciais, as quais são fundidas na primeira camada totalmente conectada. Para inserir a informação de tempo, agregaram-se informações de 10 quadros submetidos a todos os níveis convolucionais. Reduziram-se pela metade os quadros da entrada em sua resolução espacial. Para a primeira RNC, essa foi a imagem de entrada; já para a segunda RNC, valendo-se da premissa de que o objeto de interesse ocupa a região central da imagem, ela somente recebeu a porção central do objeto. A redução de resolução não prejudicou o resultado do experimento e, ainda, houve redução de 50% no tempo de treinamento.

Simonyan e Zisserman (2014a) propõem um novo método para reconhecimento de ações em vídeos, que consiste em analisar separadamente as partes espacial e temporal de uma sequência de quadros. A parte espacial traz informações sobre a cena e seus objetos a cada quadro; a sequência de quadros traz informações sobre o movimento da cena, da câmera e do movimento dos objetos presentes na cena. As análises espacial e temporal são implementadas separadamente, usando-se RNCs e classificação normalizada entre 0 e 1. O processo de análise da parte espacial não contempla nenhuma novidade, entretanto, na análise temporal, a entrada para a RNC é a variação do fluxo ótico ou a variação de trajetória em uma sequência consecutiva de quadros. Dessa maneira, os dados de entrada já representam um movimento, não sendo necessário que a RNC avalie se ele existe ou não. Para agregar os resultados das duas RNCs, aplicaram-se dois métodos: a média e a máquina de vetores de suporte (SVM) sobre o resultado normalizado da segunda camada de classificação.

Além das RNC's serem usadas em uma ampla lista de tarefas, utilizando-se imagens como entrada, Ravanbakhsh et al. (2015) empregaram vídeos, em vez de apenas imagens, com o objetivo de reconhecer ações. Utilizando a última camada de RNC antes da camada de classificação, Ravanbakhsh et al. (2015) introduziram uma estrutura hierárquica capaz de capturar a variação temporal em vídeos. O referido modelo foi capaz de capturar pequenas ações dentro de ações complexas. Essas pequenas ações representaram um grau diferente de granularidade dentro da ação completa. Propôs-se a estrutura capturando-se as características

espaciais. Para isso, utilizou-se a arquitetura *AlexNet* (Krizhevsky, Sutskever e Hinton (2012)), treinadas com a base de dados ImageNet. As características temporais, diferentemente dos trabalhos anteriores que propunham fluxo óptico, foram adquiridas a partir de pequenos intervalos do vídeo. Nesses intervalos, computaram-se informações grosseiras das características dos quadros iniciais e finais, então, dividiu-se recursivamente o intervalo e computaram-se novamente as características até que não fosse mais possível dividir o intervalo do vídeo. Tais informações formaram uma estrutura de árvore binária, na qual os níveis que se encontram mais perto da raiz representam as informações mais grosseiras e as que se encontram mais perto das folhas apresentam as características mais finas do movimento. Para a análise da estrutura de árvore binária, empregaram-se abordagens clássicas da visão computacional. O método de classificação proposto foi SVM não linear.

Dai et al. (2017) apresentaram uma nova abordagem para o reconhecimento de ações em vídeos. Os autores definem ações por atividades de curto espaço de tempo, enquanto uma atividade é um conjunto de ações e esse conjunto de ações é o foco do seu estudo. Para uma precisa localização de atividades humanas, propôs-se uma Rede de Contexto Temporal (TCN). Essa rede é similar a uma Rede Neural Convolucional, baseada em Regiões Rápidas (Fast-R CNN), porém apresenta um novo método para classificar múltiplas escalas temporais. Amostraram-se, no mesmo momento, características defasadas no tempo e no tamanho da amostra. Tais amostragens são a entrada para a RNC que classifica as amostras.

Com o objetivo de reconhecimento de ações em vídeos, Dai et al. (2017) propuseram uma abordagem, *trajectory-pooled deep-convolutional descriptor* (TDD), que utiliza, além de características aprendidas através de RNC, uma técnica construída manualmente, chamada de *Improved Trajectories* (IT). A IT faz uso de fluxo ótico para o reconhecimento de trajetórias, como proposto por (Wang et al. apud Wang and Schmid). Apesar de TDD ser independente de IT, ela foi escolhida por apresentar um bom desempenho na detecção de trajetórias. A RNC utilizada para a TDD foi proposta por Simonyan e Zisserman (2014b), a qual é empregada para aprender um mapa de características espaciais e temporais discriminativas que, em conjunto com a IT, retornaram os descritores da TDD.

Para melhorar a demanda computacional que o fluxo ótico exige, Zhang et al. (2016) propõem utilizar vetores de movimento. Com isso, demonstram que sua proposta pode realizar reconhecimento de ações em tempo real, pois as ações podem ser obtidas diretamente durante o processo de decodificação do vídeo. Além da melhoria computacional, para melhorar a performance do reconhecimento das ações, treinam-se as RNCs com os vetores de movimentos e o fluxo ótico.

Utilizando-se de redes de memória de longo e curto prazo (LSTM), alimentadas pela última camada totalmente conectada por um sistema de duas RNC proposto por Simonyan e Zisserman (2014a), Li et al. (2018) propuseram uma abordagem capaz de reconhecer a ordenação espacial em uma sequência de quadros e, com isso, ser capaz de reconhecer movimento, além de determinar o local relevante dentro do domínio do espaço tempo. Em conjunto com a LSTM, utilizou-se um mecanismo de atenção, representado por uma rede neural.

As RNAs, mais especificamente as RNCs, são fundamentais na proposta deste trabalho, uma vez que seu papel é identificar automóveis nos quadros captados pela câmera acoplada no veículo. Além da detecção dos veículos, a RNC também segmenta o objeto em relação a sua posição espacial no quadro, o que é muito importante para o sistema, uma vez que as regras de boa conduta no trânsito têm relação espacial e precisa receber essa informação como premissa. A RNC foi escolhida para esta etapa do sistema por ser capaz de ser implementadas facilmente através de diversas bibliotecas e por serem disponibilizadas já treinadas, o que reduz o tempo de implementação do sistema, além disso desde 2012 os melhores resultados na ILSVRC foram obtidos por RNC's. Em relação ao treinamento, as bases de dados, como mostrada anteriormente nesta seção são extensas em se tratando de veículos e apresentam ótimo resultado na detecção e segmentação como mostrado nas competições da ILSVRC.

Neste capítulo foram relacionadas pesquisas que formam a base técnica deste trabalho e por representarem o estado da arte das suas respectivas áreas. A seção 3.1 apresentou lógica probabilística que neste trabalho foi utilizada para inferência e classificação de situações de risco em cenas de tráfego de automóveis, utilizando-se de uma base de conhecimento que contém regras de boa conduta no trânsito. Na seção 3,2 apresentou-se as soluções e conceitos de sistemas ADAS, que serviram de base para o autor planejar o limite aceitável de risco, e obter informações sobre quais são os parâmetros utilizados para determinar risco, em situações de tráfego de veículos. Por último na seção 3.3, o estado da arte de RNC foi revisado devido a este trabalho fazer uso de uma RNC para identificar veículos em de quadros de um contexto de tráfego de veículos. No próximo capítulo estão apresentados os fundamentos teóricos referentes à RNA e lógica probabilística.

4 SISTEMA DE INTERPRETAÇÃO LÓGICO PROBABILÍSTICO

4.1 OBJETIVO

O objetivo deste trabalho é desenvolver um sistema de assistência ao motorista, capaz de interpretar e classificar automaticamente o risco de um veículo de invadir a área segura entre o ego veículo e um veículo imediatamente à sua frente, na mesma faixa de rolagem em uma sequência de cenas de tráfego de veículos. Uma RNC é utilizada para segmentar e classificar carros, ônibus, caminhões e motos dentro de um contexto de tráfego de veículos com a finalidade de gerar na sua saída um vetor com esses dados. O vetor utilizado como evidência para um sistema de inferência lógico probabilístico, escrito na linguagem probabilística ProbLog. Esse sistema de inferência lógico probabilístico contém informações de uma base de conhecimento com regras de direção defensiva. O sistema realizará inferências de um outro veículo na cena, invadir uma área determinada logo à frente do ego veículo, com o objetivo de gerar um alerta quando houver risco.

4.1 JUSTIFICATIVA

Os sistemas avançados de assistência ao motorista (ADAS) vêm crescendo nos últimos anos por duas razões. A primeira é regulatória; nos Estados Unidos e nos países da União Europeia, já será obrigatório que em 2020 existam sistemas de freio de emergência e sistemas de alerta de colisão. A segunda é mercadológica; os motoristas estão cada vez mais interessados em carros com esses sistemas de assistência (ROADMAP, 2017).

Atualmente, a Euro NCAP, entidade criada pelo Reino Unido e financiada pela União Europeia, realiza testes destrutivos em automóveis com o objetivo de avaliar a segurança dos ocupantes. Emprega um protocolo bem rigoroso de testes e somente dá nota máxima para veículos que já possuem sistemas de freio automático que previnam colisão ou que as amenizem em caso extremos.

A EuroNCAP identificou em Roadmap (2017) que motoristas, por desatenção ou erro, têm o tempo de reação de frear atrasados em relação ao tempo necessário para evitar a colisão. Para atender a essa demanda, este trabalho pretende desenvolver um sistema de assistência ao motorista que se utiliza de RNCs em conjunto com um sistema lógico probabilístico para classificação de situações de risco em contextos de tráfego de veículos.

4.1 IMPLEMENTAÇÃO

Esta seção tem como objetivo explicar como o sistema funciona e foi implementado. Conta com uma explicação genérica dos passos do sistema, terminado com um diagrama de blocos. A partir da introdução do diagrama de blocos, uma explicação mais profunda de cada etapa enumerada no diagrama é dada.

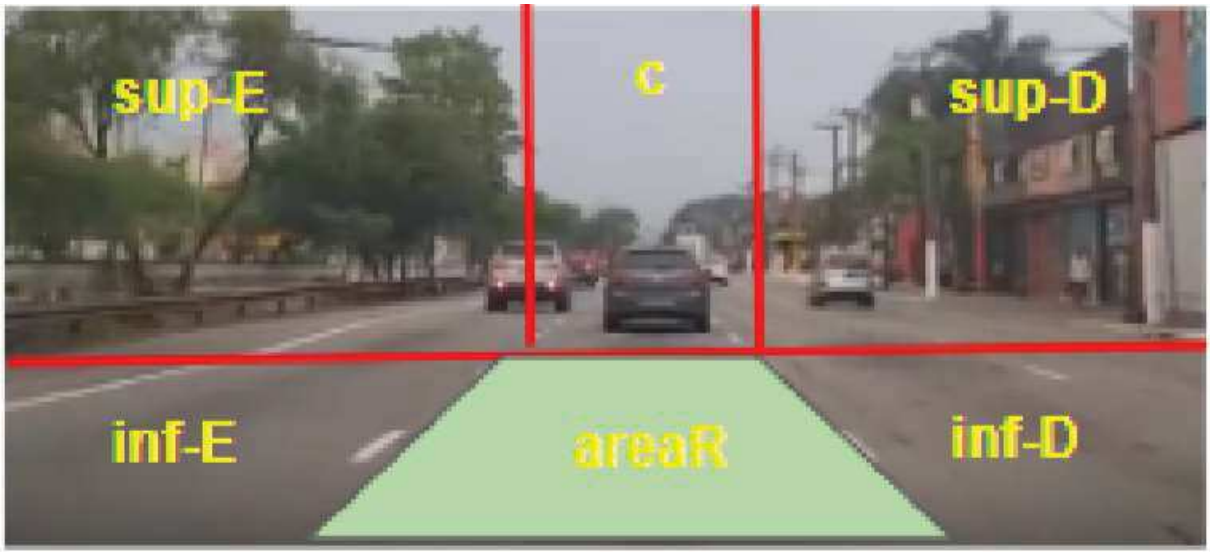
O sistema proposto foi implementado na linguagem Python com auxílio principalmente da biblioteca OpenCV, para alguns algoritmos de processamento de imagem que serão referenciados mais adiante neste capítulo. Embora exista uma implementação do ProbLog em uma biblioteca para Python, ela não foi utilizada por problemas de compatibilidade causados pela versão do Python. Devido a esse problema utiliza-se a chamada do programa em lógica probabilística via linha de comando dentro programa principal em Python. Devido a essa escolha de implementação, a troca de informação entre o Python e o ProbLog é feita através da leitura de arquivo de texto.

A determinação da área segura foi implementada utilizando-se técnicas de visão computacional, que detectam a faixa de rolagem, e geram uma região à frente do ego veículo, que será chamada de região segura. Essa região segura é a discretização para "mantenha uma distância segura do veículo a frente". Adicionalmente a região segura, as cenas foram discretizadas em outras 5 regiões: direita inferior, direita superior, central superior, esquerda inferior e esquerda superior.

A figura 15 demonstra as regiões de discretização conforme os itens abaixo:

a) Superior esquerda (sup-E).
b) Central (c).
c) Superior direita (sup-D).
d) Inferior esquerda (inf-E)
e) Região risco (areaR)
f) Inferior direita (Inf-D)

Figura 15-Regiões de discretização



Fonte: Autor

A RNC do sistema deverá segmentar e classificar veículos, com a finalidade de gerar na sua saída um vetor com todos os veículos da cena e suas respectivas posições. O vetor será utilizado como evidência para um sistema de inferência lógico probabilístico, escrito na linguagem probabilística ProbLog.

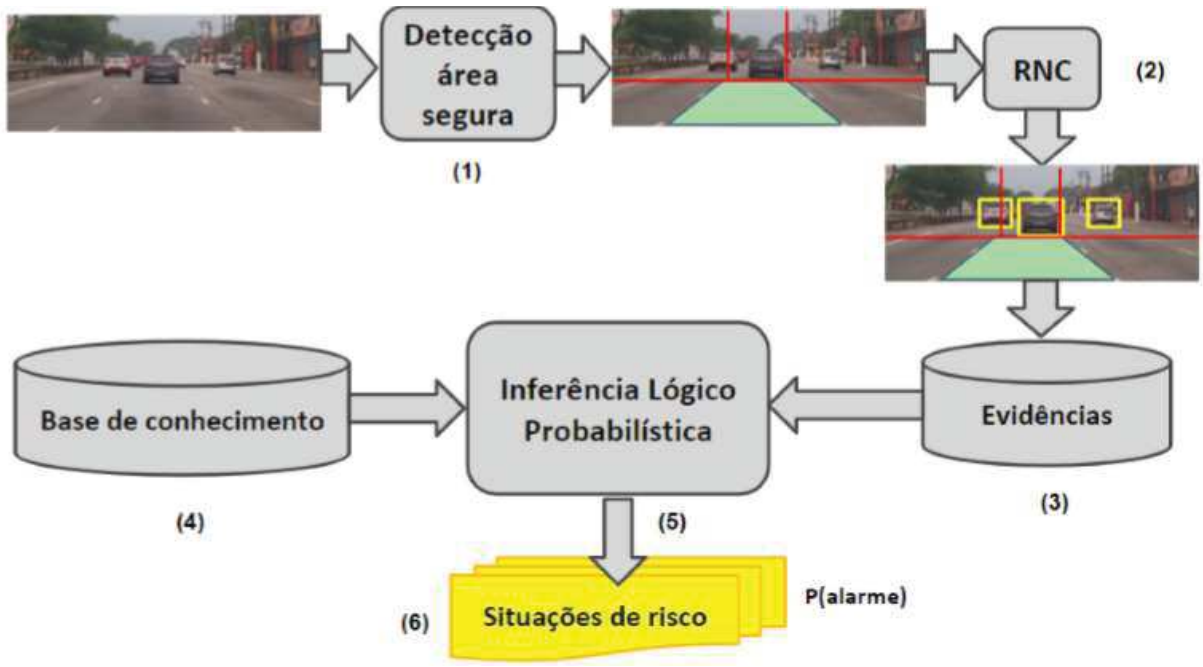
O sistema de inferência lógico probabilístico proposto, também recebe como entrada informações de uma base de conhecimento que contêm regras sobre direção defensiva que tem como objetivo antecipar a eminente invasão da região segura entre o ego veículo e um veículo imediatamente a sua frente na mesma faixa de rolagem.

O sistema realizará inferências do risco da invasão da área segura à frente do ego veículo, com o objetivo de gerar um alerta quando o risco inferido for relevante.

O sistema proposto no trabalho é esquematizado na figura 16.

As próximas subseções referem-se ao o funcionamento de cada etapa enumerada na figura 16, onde os números de cada subseções referem-se aos números dos módulos da figura 16.

Figura 16-Diagrama de blocos do sistema



Fonte: Autor

4.4 ETAPA 1: DETECÇÃO DA ÁREA SEGURA

Nesta etapa está detalhado como foram implementadas as técnicas de visão computacional utilizadas para determinação da área segura. As próximas subseções explicam sequencialmente as etapas.

4.4.1 Determinação da região de interesse (ROI)

A região de interesse para a identificação das faixas de rolagem foi determinada para que fosse possível capturar a faixa de rolamento atual do ego veículo tanto a sua esquerda quanto à sua direita. Todo o resto do quadro é desprezado para não gerar ruído durante a identificação da faixa de rolagem. Por se tratar da primeira etapa de visão computacional, o pré-processamento dos quadros é feito nesta etapa.

Primeiro a imagem é transformada de através do OpenCV de RGB 16-bits para a escala de cinza de 8-bits, através da função *cvtColor* e usando COLOR−RGB2GRAY como parâmetro. A próxima etapa foi utilizar um filtro de suavização de Gauss, utilizou-se a função *GaussianBlur* e como parâmetros, kernel de 5x5 e 5 desvios padrões tanto sigma na direção X quanto na direção Y. A figura 17, mostra a esquerda a imagem real em escala de cinza e a da

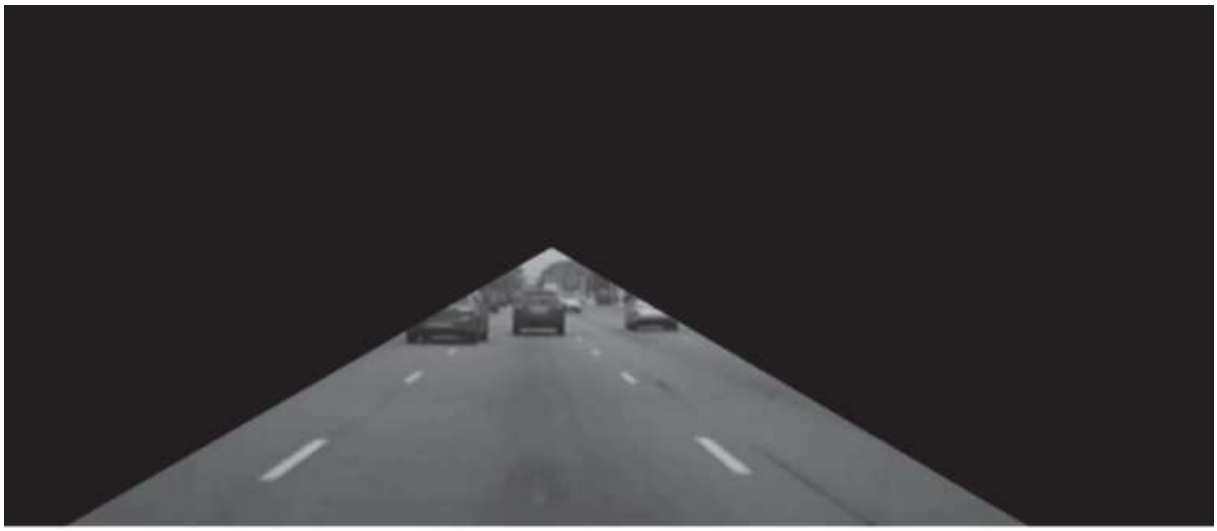
direita mostra a imagem suavizada, essa suavização resultou em uma melhor performance da identificação da faixa de rolamento.

Figura 17-Comparativo de suavização

Fonte: Autor

Uma vez que foi realizado o pré-processamento das imagens, a próxima etapa é manter no quadro apenas a área de interesse da análise. As dimensões da ROI foram determinadas empiricamente através de testes de acordo com o posicionamento da câmera no ego veículo. Na figura 18 é mostrado o resultado do quadro com a ROI definida.

Figura 18-ROI utilizada para detecção das faixas de rolagem



Fonte: Autor

4.4.2 Detecção das bordas nas faixas de rolamento

A detecção de bordas utilizou o algoritmo de Canny (1986), para identificar as faixas de rolamento. A proposta de Canny foi introduzir um algoritmo multi-passos para analisar a possibilidade de existência de uma borda. Canny (1986) utiliza cinco etapas para a detecção de borda:

a) Redução de ruído: Um filtro de Gauss é utilizado para a suavização da imagem e redução do ruído.
b) Cálculo dos gradientes vertical e horizontal de intensidade está representado na equação 18.
c) Orientação da borda: A partir da equação 18, calcula-se a orientação do vetor gradiente em cada pixel pela equação 19
d) Supressão não máxima: Para todos pixels de um quadro, calcula-se qual a direção que mais se aproxima do resultado da direção do gradiente obtida no passo anterior. Canny (1986) limita essas direções em:0º, 45º, 90º e 135º. Para todos os pixels, avalia-se a partir do quadro atual $I_a(x,y)$ se o contraste dos pixels adjacentes, limitados aos ângulos citados anteriormente, se o valor do pixel é menor, então este pixel, em um novo quadro $I_n(x,y)$ será igual a 0. No exemplo da figura 19, a análise do pixel de valor 20 realçado em vermelho e também dos seus adjacentes, para o ângulo de 90º do quadro I_a, se seu valor for menor que um de seus adjacentes seu novo valor em I_n será zero.
e) Limiarização com histerese: Segundo Canny (1986) a limiarização pode não eliminar os ruídos de alta amplitude se um limiar baixo for escolhido, ou ainda eliminar bordas reais se um limiar alto for escolhido. Canny (1986) propôs uma solução que se baseia na afirmação que pixel pertencentes a uma borda, geralmente estão conectados. Para checar essa afirmação, dois limiares são utilizados, L_{baixo} e L_{alto}. Para L_{alto} checar se $I_a(x,y) \geq L_{alto}$, quando o resultado dessa afirmação for verdadeiro, esse pixel é uma borda. Para o mesmo pixel, $I_a(x,y)$, checar os pixels adjacentes na direção perpendicular ao gradiente $\theta(x,y)$ se o valor do pixel for maior que L_{baixo}, então este pixel é uma borda.

$$G(x,y) = \sqrt{\left(\frac{\delta f(x,y)}{\delta x}\right)^2 + \left(\frac{\delta f(x,y)}{\delta y}\right)^2} \tag{18}$$

$$\theta(x,y) = \arctan\left(\frac{\frac{\delta f(x,y)}{\delta y}}{\frac{\delta f(x,y)}{\delta x}}\right) \quad (19)$$

Figura 19-Supressão não máxima

Ia =					
	0	20	20	0	20
	0	0	0	90	150
	0	20	90	150	11
	20	100	140	12	0
	100	135	18	0	0

In =					
	0	20	20	0	20
	0	0	0	90	150
	0	0	90	150	0
	0	100	140	0	0
	100	135	0	0	0

Fonte: Autor

O sistema foi implementado utilizando-se o algoritmo de Canny disponível na biblioteca OpenCV. Os limiares foram definidos utilizando-se o valor médios dos pixels do quadro analisado, reduzido em 30% do seu valor absoluto. Então os limiares máximos são adicionados 5% para o limiar alto e retirado 5% para os limiares baixos. Estes valores foram encontrados empiricamente durante a parametrização do sistema. A figura 20 mostra à esquerda o quadro apenas com a ROI e à direita o resultado depois da aplicação dos algoritmos de Canny. Nota-se que todas as faixas de rolamento foram identificadas.

Figura 20-Identificação das linhas

Fonte: Autor

4.4.3 Criação de linhas a partir das bordas

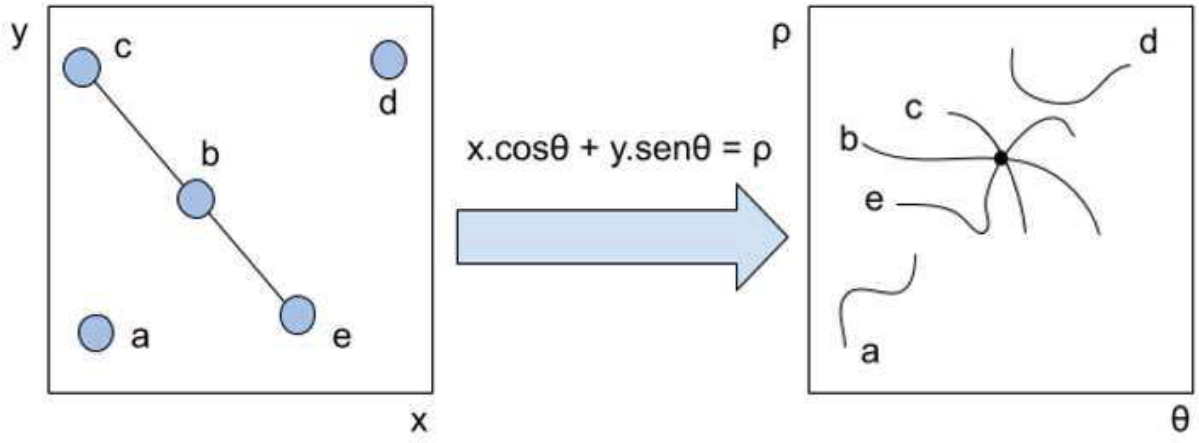
Com as bordas das faixas de rolamento já identificadas, é necessário criar linhas que representem as faixas de rolagem, mas também excluir as bordas que não representam as faixas de rolagem. Para esta tarefa, o sistema foi implementado utilizando-se da transformada de Hough (DUDA; HART, 1972).

O princípio da transformada de Hough é o mapeamento dos pontos de uma imagem em um plano de parâmetros. Gonzalez e Woods (2006) explica que dado um ponto *(x,y)* em quadro e a equação geral da reta, na forma de inclinação-ponto de intersecção $y_i = ax_i + b$. O uso da equação da reta como representação de linhas causa problemas quando a inclinação e a intersecção se aproximam da vertical, o que leva a equação a tender ao infinito. A maneira encontrada para solucionar esse problema foi a representação normal de uma linha, como na equação 20.

$$x\cos\theta + y\theta = \rho \tag{20}$$

A ideia por trás da transformada Hough é aplicar em um quadro uma transformação em que todos os pontos que façam parte de uma mesma curva, sejam representados em um único ponto no domínio da transformada de Hough. A figura 21 mostra a transformada de Hough, a esquerda temos um quadro representado no domínio espacial *(x,y)* e os mesmos pontos, no lado direito representados no domínio da transformada de Hough *(θ,ρ)*. Observa-se que cada ponto agora é uma curva, e para os pontos *c, b* e que no domínio espacial faziam parte de uma reta, na transformada suas curvas têm uma intersecção no mesmo ponto, o que representa que tais pontos pertencem a uma mesma reta.

Figura 21-Transformada de Hough



Fonte: Autor

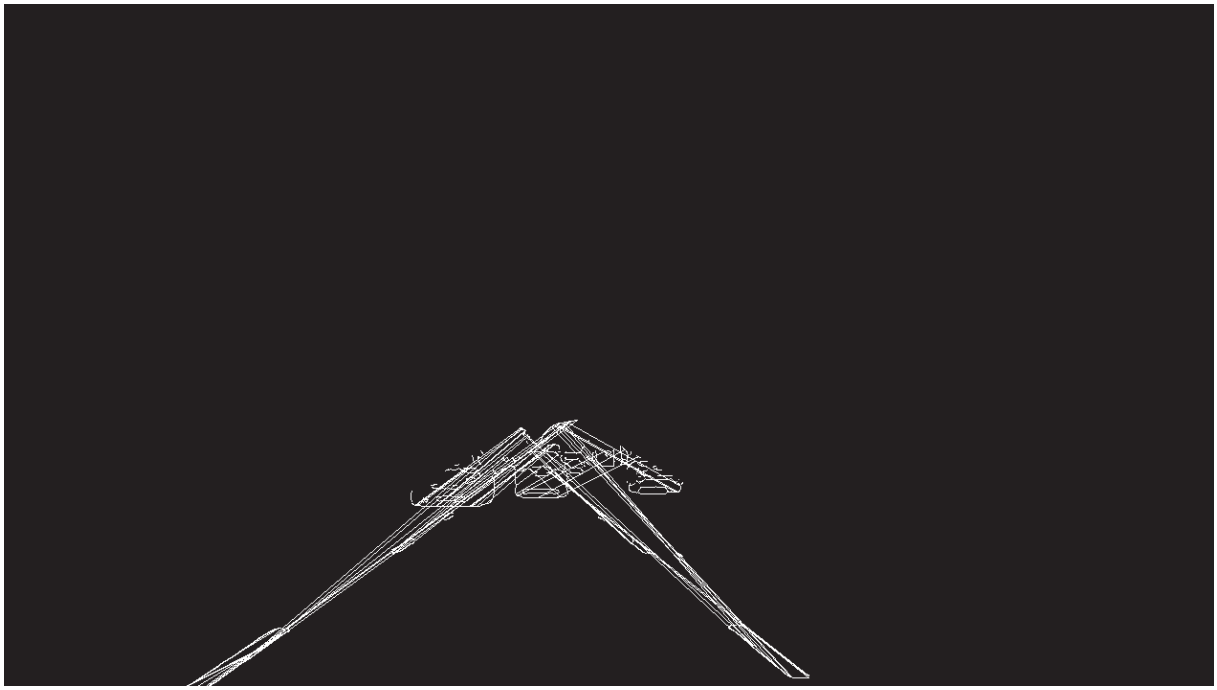
Uma vez que todos os pontos no domínio espacial já tenham sido analisados, as retas formaram um acúmulo de pontos no domínio da transformada de Hough, portanto para a identificação das retas, checam-se os pontos de máximo local no domínio da transformada de Hough.

Neste trabalho foi utilizada a função *HoughLinesP* da biblioteca OpenCV. Como parâmetro esta função necessita definir a resolução de ρ e θ, os valores utilizados foram de *30* e $\pi \backslash 60$ respectivamente, estes parâmetros foram definidos empiricamente avaliando onde havia a maior concentração de pontos relevantes para o contexto do estudo. Outros dois parâmetros já incorporados para esta função, e que ajudam a filtrar parte apenas da informação são o comprimento mínimo da linha e a distância máxima entre uma linha. Para o primeiro, foi usado 10 pixels como valor, este valor baixo ajuda a manter a identificação das linhas sempre presente, apesar de trazer algum ruído. Para a distância entrelinhas, 200 pixels foi o valor estabelecido porque este é o valor observado aproximado da distância entre as faixas de rolamento na região média em *y* do quadro.

Na Figura 22 são apresentadas as linhas que foram identificadas durante a transformada de Hough. Uma outra ferramenta utilizada para reduzir as retas que são mais próximas de serem horizontais são retiradas da lista de linhas que será usada nas próximas etapas. A regra utilizada para retirar essas linhas está representada na equação 21.

$$abs\left(\frac{y_1 - y_2}{x_1 - x_2}\right) > 0{,}5 \tag{21}$$

Figura 22-Linhas identificadas após a transformada de Hough



Fonte: Autor

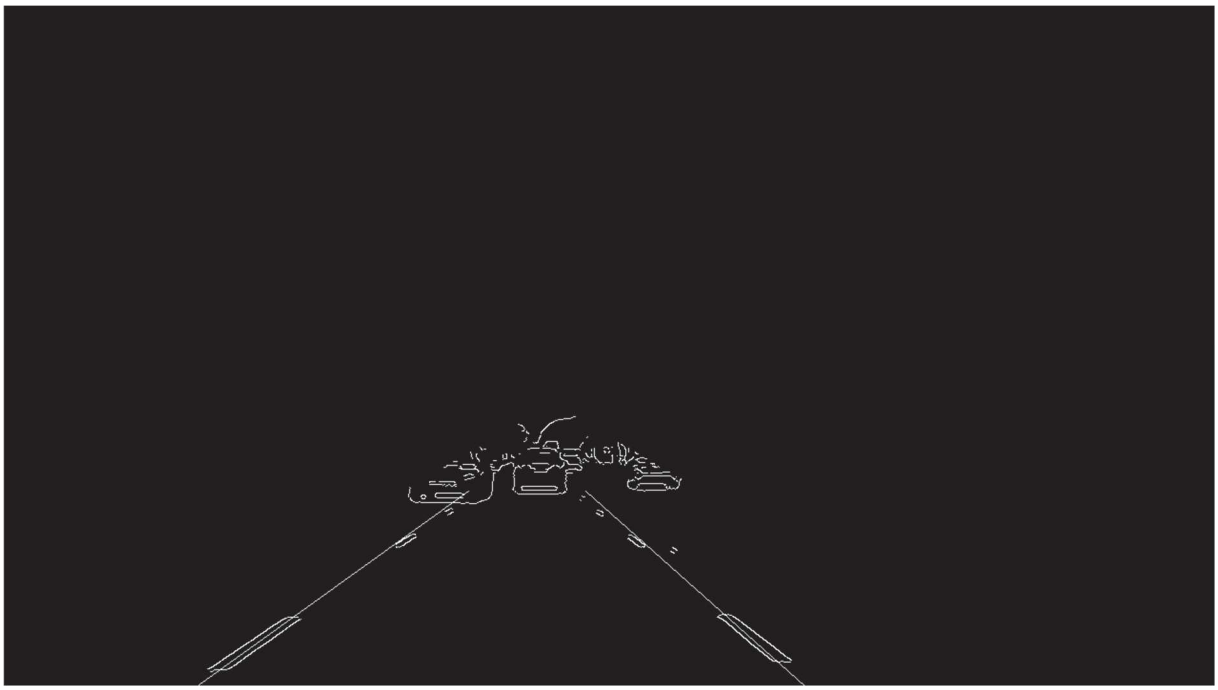
4.4.4 Identificação das faixas de rolamento a partir das linhas

A próxima etapa do sistema é classificar todas as linhas da etapa anterior e separá-las em dois vetores distintos um de linhas a esquerda e outro de linhas a direita. O método utilizado é muito semelhante ao utilizado para verificar se a reta não era horizontal. Nesta etapa é checado se a inclinação é positiva ou negativa, através da equação 21. Porém, resultados negativos classificam linhas esquerdas, enquanto resultados positivos classificam linhas a direita.

Como o objetivo é a criação de uma única linha tanto para a faixa da esquerda como para a faixa da direita, os valores de y podem ser definidos a priori. O valor de y_{min} será zero, uma vez que a linha deve iniciar na margem do quadro. Para o valor y_{max}, foi determinado em 3/4 da altura total de y, este valor foi determinado empiricamente avaliando-se a linha do horizonte no quadro, e aproximadamente a distância de dois veículos.

Para a determinação dos valores de x, tanto na parte superior quanto inferior da linha foi utilizado duas funções do OpenCV. A primeira, *polyfit* utiliza das regras dos mínimos quadrados para gerar um vetor médio das retas já identificadas, enquanto *poly1d* encontra os respectivos valores de x_{min} e x_{max} para os valores de y_{min} e y_{max} que são constantes. Como resultado temos dois vetores que representam as linhas esquerda e direita. Na imagem 23 temos apenas duas linhas, uma a esquerda e outra direita que coincidem com as faixas de rolamento.

Figura 23-Faixas de rolamento identificadas



Fonte: Autor

4.4.5 Geração da área segura

Conhecendo-se os pontos das linhas que limitam as faixas de rolagem esquerda e direita, pode-se gerar uma região que limita toda a região, na figura 24, com as linhas e área segura já no quadro original verifica-se a região segura já segmentada.

Ao final da etapa (1) teremos a cena discretizada, como apresentado na figura 25. Utilizando-seafigura25como exemplo, ela foi dimensionada em ambos os eixos de 0 a 100, o sistema gera um vetor que representa essa discretização. O sistema representa cada área discretizada como um vetor que contêm seus 4 vértices. A região 1 mostrada na 25, como exemplo, seria formada pelo seguinte vetor: V1 = {(0,30),(40,30),(40,100),(0,100)}.

Figura 24-Região Discretizada final

Fonte: Autor

Figura 25-Exemplo da Região Discretizada

Fonte: Autor

4.5 ETAPA 2: SEGMENTAÇÃO E CLASSIFICAÇÃO DAS CENAS

Nesta etapa é apresentada a RNC que foi utilizada na implementação, quais as técnicas utilizadas pela RNC para fazer a detecção e segmentação dos objetos por ela detectados. É apresentada também a base de dados na qual a RNC foi treinada.

O sistema proposto utiliza a RNC chamada *Yolo* do inglês, *You only look once*, que traduzindo para o português significa: Você apenas olha uma vez. *Yolo* foi proposto em Redmon

et al. (2016) e possui uma arquitetura unificada que utiliza apenas uma rede neural tanto para delimitar o objeto quanto para probabilidade da classe do objeto.

Yolo está disponibilizada no *GitHub*, é um projeto de fonte aberta e sua plataforma é escrita em linguagem C e suporta computação via processador gráfico. Pelo fato de ser um projeto de fonte aberta, posteriormente foi também implementada em Python, versão que é usada neste trabalho. uma das versões que são disponibilizadas já é treinada na base de dados *COCO.*

COCO é uma abreviação em inglês para *Commom objects in context* traduzindo para o português: Objetos comuns no contexto. *COCO* como descrito em Lin et al. (2014) é uma base que contém mais de 330 mil imagens e 80 classes.

Das 80 classes presentes na base de dados, este trabalho apenas utilizará 3: carros, motos e caminhões. Porém os tratará de forma igual como veículo.

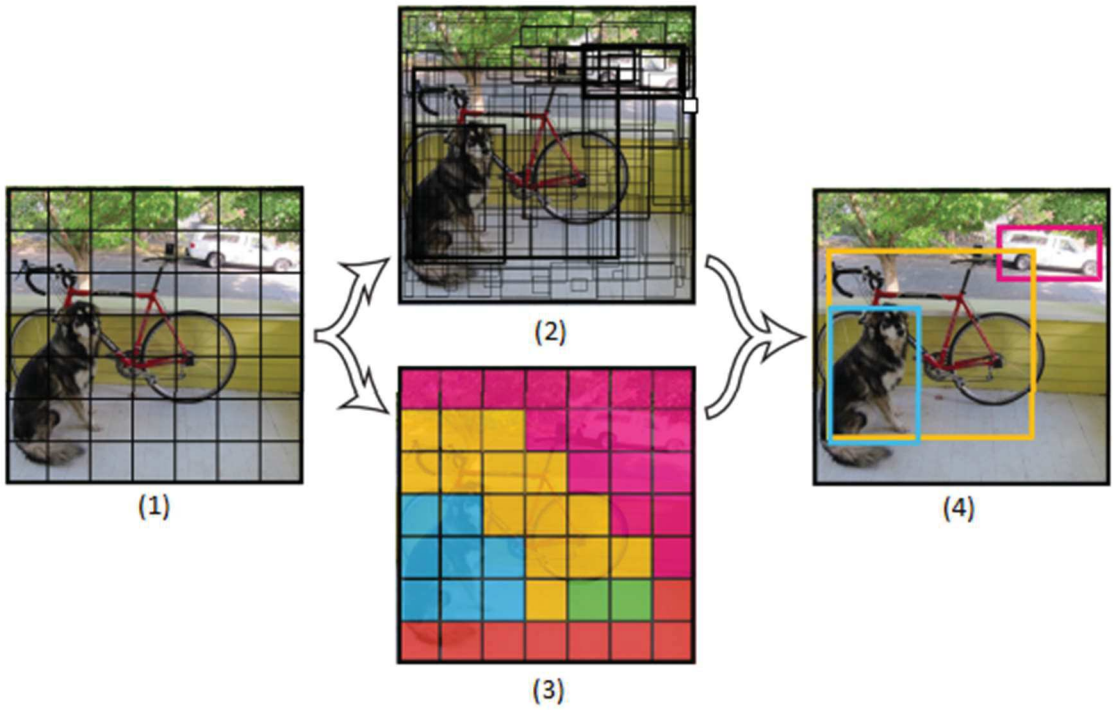
Para a predição dos delimitadores dos objetos de uma cena, *Yolo* usa características da cena inteira para encontrar todos os delimitadores de todas as classes simultaneamente. Uma cena é dividida em uma grade *S x S*, sempre que o centro de um objeto pertencer a uma célula da grade, considera-se que aquela célula é a responsável por detectar o objeto em questão. Cada célula realiza a predição de *B* delimitadores e um valor de confiança. Esse valor de confiança mostra o quão confiante o modelo está de que um delimitador contém um objeto. Cada célula ainda faz a predição condicional *C* da probabilidade da classe do objeto, *P(Classe|Objeto).*

No exemplo da figura 26, (1) mostra a divisão da cena em S x S células, em (2) observa-se a predição dos delimitadores, na imagem 26 o valor de confiança é representado pela espessura do traço do delimitador, quanto mais escuro maior a confiança. Em (3) para cada célula é avaliada a probabilidade daquela célula pertencer a uma classe, por fim em (4) combinam-se as probabilidades de classe com os delimitadores com maior nível de confiança para obter na saída da RNC um objeto delimitado e classificado.

Yolo na sua estrutura de rede possui 24 camadas de convolução em 6 estágios, seguido de 2 camadas totalmente conectadas na saída. Os 4 primeiros estágios de convolução são encerrados por uma função de agregação de máximo valor de tamanho 2 x 2 e deslocamento 2.

O objetivo principal desta etapa é produzir um vetor que contenha todos os veículos detectados, com suas respectivas posições espaciais. Como exemplo, na figura 27 temos a detecção de um veículo, representado por um retângulo amarelo que está na região *c*. Este veículo é descrito como um vetor representado pela letra *A* que está acima do retângulo. O primeiro par ordenado do vetor (45,30), refere-se ao canto superior esquerdo do retângulo, já o segundo par ordenado (59,31), refere-se ao canto inferior esquerdo do retângulo.

Figura 26 - Processo de detecção Yolo



Fonte: Redmon et al., 2016

Figura 27-Vetor do objeto detectado

Fonte: Autor

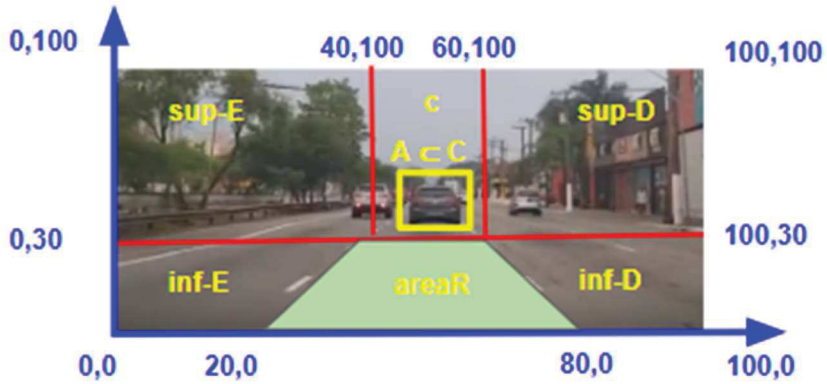
4.6 ETAPA 3: CRIAÇÃO DA BASE DE DADOS DE EVIDÊNCIA

Nesta etapa está detalhado como os dados recebidos da etapa (1) e da etapa (2) são combinados para gerar a base de conhecimento de evidências, e como esse dado é tratado, transformado e enviado para o ProbLog.

A partir das regiões discretizadas da etapa (1), o vetor de saída da RNC da etapa (2), é possível classificar a posição relativa dos veículos detectados pela RNC em cada cena, de acordo com as seis regiões discretizadas de acordo com a figura 15.

No exemplo da figura 28, as regiões discretizadas da etapa (1), em conjunto com o automóvel *A*, identificado pela RNC na etapa (2), são as entradas para esta etapa. A partir dessas duas informações anteriores é possível então classificar o automóvel de acordo com a região que ele se encontra. Ainda usando o exemplo da figura 28 como exemplo, é possível afirmar que o automóvel *A* está contido na região *c*.

Figura 28-Classificação da região do objeto detectado



Fonte: Autor

Nesta etapa (3), os veículos classificados e discretizados por região, são transferidos para um PLP, escrito em ProbLog, em forma de evidências.

4.7 ETAPA 4: BASE DE CONHECIMENTO DIREÇÃO DEFENSIVA

Nesta seção é apresentada a base de conhecimento de direção defensiva que o autor propôs para este trabalho. Esta etapa utiliza as mesmas regiões discretizadas que já foram mencionadas anteriormente para descrever o contexto.

A possibilidade de um carro se movimentar, e mudar de faixa de rolamento é uma regra importante para o sistema, uma vez que define a possibilidade de um veículo poder se movimentar na cena, baseando-se na existência ou não de carros ao seu redor. As regras foram criadas levando-se em conta sua posição atual. Abaixo estão as regras propostas:

a) Um carro está em uma determinada região discretizada;
   - carroEM(x).

b) Um veículo pode se movimentar se ele estiver na região *x* e a região *y* para a qual ele pode ir não esteja ocupada.
   - podeIR(X,Y) :- carroEM(X), \+carroEM(Y).

A regra carroEM(X), busca seu valor na base de conhecimento da etapa (3), e não precisa ser instanciada uma vez que *X* receberá seu valor durante a inferência da query. Por outro lado, *podeIR*(X,Y) precisa ser instanciada pois descreve as restrições impostas pela discretização como mostrado a seguir:

a) Um veículo que está na região inferior direita da cena, pode se movimentar para a esquerda e avançar para a região de risco do ego veículo, ou acelerar para a região superior direita, as sentenças a seguir em ProbLog codificam essas afirmações;
   - podeIR(inf-D,sup-D);
   - podeIR(inf-D,areaR).

b) Um veículo que está na região inferior esquerda da cena, pode se movimentar para a direita e avançar para a região de risco do ego veículo, ou acelerar para a região superior esquerda, as sentenças a seguir em ProbLog codificam essas afirmações;
   - podeIR(inf-E,sup-E);
   - podeIR(inf-E,areaR).

c) Um veículo que está na região central da cena, pode se movimentar para a direita superior ou para a região esquerda superior. O mesmo veículo pode frear e entrar na região de risco, as sentenças a seguir em ProbLog codificam essas afirmações.
   - podeIR(c,areaR);
   - podeIR(c,sup-D);
   - podeIR(c,sup-E).

A seguir estão as regras e os fatos probabilísticos que codificam a seguinte premissa de direção defensiva, manter distância segura entre o ego veículo e os outros veículos da cena:

a) Um veículo tem probabilidade *P* de se movimentar de sua posição atual para a região de risco.
   - P :: mover−areaR.

Entretanto essa regra precisa ser instanciada e deve receber sua probabilidade e restrições de acordo com cada situação risco identificada e explicada abaixo:

a) A probabilidade de um veículo que está na região *inf-D* de invadir a região areaR, tendo como evidência a existência de veículos tanto em *inf-D* quanto em *sup-D* é de 70%. Entre as regiões possíveis que o veículo de *inf-D* pode se locomover, a única

possível é a região *areaR* uma vez que *sup-D* tem um veículo. Portanto o valor dado a probabilidade da movimentação do veículo em *inf-D* de se locomover para a *areaR* é alta, foi estabelecido a probabilidade 70% para esse fato. Para esta probabilidade foi levado em conta que, um veículo a direita que estiver em aceleração deve realizar a manobra de ultrapassagem pela esquerda de acordo com o código brasileiro de trânsito;

- 0.7:: mover−areaR :−podeIR(inf−D,sup−D),podeIR(inf− D,areaR), carroEM(sup−D),carroEM(inf −D).

b) A probabilidade de um veículo que está na região *inf-D* de invadir a região *areaR*, tendo como evidência a existência de veículos apenas em *inf-D* é de 20%. Entre as regiões possíveis que o veículo de *inf-D* pode se locomover, existem duas possíveis movimentações. Para região areaR ou para a região *sup-D.* Portanto o valor dado a probabilidade da movimentação do veículo em *inf-D* de se locomover para a arear é mais baixa que regra anterior uma vez que a região *sup-D* está vazia. Para esta probabilidade foi levado em conta que, um veículo a direita que estiver em aceleração deve realizar a manobra de ultrapassagem pela esquerda de acordo com o código brasileiro de trânsito, entretanto senão existe nenhum outro veículo a sua frente não há necessidade de trocar de faixa imediatamente, por esta razão sua probabilidade de se movimentar para a *areaR* é inferior quando não existe um veículo a frente;

- 0.2:: mover−areaR :−podeIR(inf−D,sup−D), podeIR(inf− D,areaR),\+carroEM(sup−D),carroEM(inf −D).

c) A probabilidade de um veículo que está na região *inf-E* de invadir a região *areaR*, tendo como evidência a existência de veículos tanto em *inf-E* quanto em *sup-E* é de 30%. Entre as regiões possíveis que o veículo de *inf-E* pode se locomover, a única possível é a região *areaR* uma vez que *sup-E* tem um veículo. Portanto o valor dado a probabilidade da movimentação do veículo em *inf-E* de se locomover para a *areaR* é alta, entretanto uma ultrapassagem pela direita é proibida pelo código brasileiro de trânsito e por isso foi estabelecido a probabilidade 30% para esse fato. Para esta probabilidade foi levado em conta que, um veículo a esquerda que estiver em aceleração não deve realizar a manobra de ultrapassagem pela direita, mas apenas quando necessitar trocar de faixa de rolagem para uma conversão a direita;

- 0.3:: mover−areaR :−podeIR(inf−E,sup−E), podeIR(inf− E,areaR),carroEM(sup−E),carroEM(inf −E).

d) A probabilidade de um veículo que está na região *inf-E* de invadir a região *areaR*, tendo como evidência a existência de veículos apenas em *inf-E* é de15%. Entre as regiões possíveis que o veículo de *inf-E* pode se locomover, existem duas possíveis movimentações. Para região *areaR* ou para a região *sup-E*. Portanto o valor dado a probabilidade da movimentação do veículo em *inf-E* de se locomover para a *areaR* é mais baixa que regra anterior uma vez que a região *sup-E* está vazia. Portanto o valor dado a probabilidade da movimentação do veículo em *inf-E* de se locomover para a *areaR* é baixa, uma vez que região a sua frente está livre por isso foi estabelecido a probabilidade 15% para esse fato. Para esta probabilidade foi levado em conta que, um veículo a esquerda que estiver em aceleração não necessita de trocar de faixa de rolagem se não precisar fazer uma conversão a direita;

   – 0.15:: mover−areaR :−podeIR(inf−E,sup−E),podeIR(inf− E,areaR), \+carroEM(sup−E),carroEM(inf −E).

e) A probabilidade de um veículo que está na região c de invadir a região *areaR*, tendo como evidência a existência de veículos em *sup-D* e *sup-E* é de 75%. Entre as regiões possíveis que o veículo de *c* pode se locomover, existe apena uma possível movimentação, para região *areaR*. Portanto o valor dado a probabilidade da movimentação do veículo em c de se locomover para a *areaR* é alta. O valor dado a probabilidade da movimentação do veículo em *c* de se locomover para a *areaR* é alta, uma vez que as regiões a sua frente estão ocupadas. Para esta probabilidade foi levado em conta que, um veículo em *c*, e que no mesmo quadro tiver, veículos a sua esquerda e a sua direita, deve reduzir sua velocidade para trocar de faixa de rolamento e portanto, invadindo a região *areaR*;

   – 0.75:: mover−areaR :−podeIR(c,areaR),podeIR(c,sup−D),podeIR(c,sup−E), carroEM(c), carroEM(sup−D), carroEM(sup−E)

f) A probabilidade de um veículo que está na região *c* de invadir a região *areaR*, tendo como evidência a existência de veículos em *sup-D* é de 25%. Entre as regiões possíveis que o veículo de *c* pode se locomover, existem duas possíveis movimentações, para região arear e *sup-E*. Portanto o valor dado a probabilidade da movimentação do veículo em *c* de se locomover para a *areaR* é baixa. O valor dado a probabilidade da movimentação do veículo em *c* de se locomover para a *areaR* é baixa, uma vez que as regiões *sup-E* está livre. Para esta probabilidade, dado um veículo em *c*, e que no mesmo quadro tiver a possibilidade de se movimentar para a região *sup-E*, que é a região onde se encontra a faixa de rolamento para os veículos

realizarem a ultrapassagem segura, e não tenha a intenção de fazer uma conversão a direita, ele veículo terá baixa probabilidade de invadir a região arear;

- 0.25:: mover−areaR :−podeIR(c,areaR),podeIR(c,sup−D),podeIR(c,sup−E), carroEM(c), carroEM(sup−D),carroEM(sup−E).

g) A probabilidade de um veículo que está na região *c* de invadir a região *areaR*, tendo como evidência a existência de veículos em *sup-E* é de 40%. Entre as regiões possíveis que o veículo de *c* pode se locomover, existem duas possíveis movimentação, para região *areaR* e *sup-D.* Portanto o valor dado a probabilidade da movimentação do veículo em *c* de se locomover para a *areaR* é média. O valor dado a probabilidade é médio, uma vez que ultrapassagem pela direita é proibida pelo código brasileiro de trânsito e por isso foi estabelecido a probabilidade 40% para esse fato. Para esta probabilidade foi levado em conta que, um veículo em *c* não deve realizar a manobra de ultrapassagem pela direita, sendo assim este veículo irá invadir a região *areaR* se necessitar frear para realizar uma conversão a esquerda ou apenas troca de faixa de rolamento;

- 0.4:: mover−areaR :−podeIR(c,areaR),podeIR(c,sup−D),podeIR(c,sup−E), carroEM(c),carroEM(sup−D), carroEM(sup−E).

h) A probabilidade de um veículo que está na região *c* de invadir a região *areaR*, não tendo como evidência a existência de veículos em *sup-E* e *sup-D* é 15%. Entre as regiões possíveis que o veículo de *c* pode se locomover, existem três possíveis movimentação, para região *areaR*, *sup-E* e *sup-D*. Portanto o valor dado a probabilidade da movimentação do veículo em *c* de se locomover para a *areaR* é baixa. Para esta probabilidade foi levado em conta que, um veículo em *c* não tem restrição lateral de movimentação e irá invadir a região arear se necessitar frear para realizar uma conversão a esquerda ou direita, ou ainda apenas para a troca de faixa de rolamento.

- 0.15:: mover−areaR :−podeIR(c,areaR),podeIR(c,sup−D),podeIR(c,sup−E), carroEM(c),carroEM(sup−D), carroEM(sup−E).

A partir das regras acima mencionadas que estão codificadas no sistema através do ProbLog, o programa em lógica contém todas as regras proposta e fatos probabilísticos, propostos neste trabalho, instanciados e que representam as regras de direção segura citadas nesta etapa (4).

4.8 ETAPA 5: INFERÊNCIA LÓGICO PROBABILÍSTICA

Nesta etapa (5) é apresentada a query que o sistema realiza no ProbLog. Esta etapa do sistema roda uma rotina vai linha de comando, que faz com que o programa em ProbLog carregue os dados das etapas (1), (2) e (3) que estão disponibilizadas em um arquivo de texto. Estes dados serão as evidências para a inferência do sistema.

A query que se deseja saber o resultado neste trabalho é probabilidade de um veículo na cena de invadir a região *c*. Mais especificamente, queremos saber a probabilidade total para a regra *P::mover−areaR*. No programa em ProbLog, a sentença *query(mover −areaR)* representa a pesquisa da probabilidade de um veículo invadir a região *areaR*.

A probabilidade calculada para esta pesquisa é marginal uma vez que seu cálculo é feito sobre as evidências observadas no contexto.

4.9 ETAPA 6: SITUAÇÕES DE RISCO

A classificação proposta e implementada no sistema é binária para a existência de risco. Ao fim da etapa (5) o sistema retorna à probabilidade de um veículo invadir a região areaR, essa probabilidade então é comparada com um limiar determinado e $P(risco) > 0.75$ pelo autor. O valor de 0.75 foi determinado para que o sistema seja conservador em relação ao risco da cena, e consiga retornar ao usuário mais situações com potencial de risco de um veículo invadir a sua área de risco.

5 RESULTADOS

Neste capítulo estão apresentados os resultados da avaliação experimental da implementação do sistema que foi proposto no capítulo anterior. Os resultados estão separados em 2 seções diferentes uma para cada sequência de quadros. Inicialmente está detalhado o método de como os resultados serão avaliados.

5.1 MÉTODO DE AVALIAÇÃO

Segundo Saito e Rehmsmeier (2015), a avaliação da performance dos classificadores é importante para poder avaliar sua utilidade e poder compará-los com métodos similares.

O resultado do sistema classificador proposto é binário. O resultado deste trabalho apresenta uma base de dados não equilibrada em relação a positivos e negativos, principalmente pelo motivo que uma sequência de cenas de trânsito apresentar pouca variação em termos de informação de cena para cena. Saito e Rehmsmeier (2015) mostra que para classificadores binários, com base de dados não equilibrada de resultados positivos e negativos, o método de avaliação indicado para avaliar a performance do sistema classificador é a curva de precisão x revocação (PRR). O gráfico de PRR consegue visualmente dar indicações para o momento da análise, assim como uma interpretação intuitiva e prática da performance do classificador.

A resposta da inferência da possibilidade de um veículo estar na área de risco, calculado pelo ProbLog, é comparado com o limiar determinado pelo autor para gerar um alarme de risco. Este valor foi determinado em $P(risco)>0.75$ para classificar uma cena que contém a possibilidade de um veículo estar na área de risco. O resultado do classificador então é comparado com um classificador padrão feito por um especialista.

Para exemplificar a classificação de alarme, a tabela 2 mostra duas situações: uma onde o alarme deve estar ligado e outra onde deve estar desligado. Na tabela o numeral *1* significa alarme ligado, e *0* alarme desligado. Na primeira linha o especialista afirmou que a cena continha risco portanto há alarme. Para o sistema dado um limiar de 0.75 há alarme uma vez que a probabilidade 0.82 é maior que 0.75. Na segunda linha o especialista afirmou que a cena não continha risco portanto não há alarme. Para o sistema dado um limiar de 0.75 não há alarme uma vez que a probabilidade 0.75 não é maior que 0.75.

Tabela 2-Resultado: Especialista x Sistema

Especialista (Alarme)	Sistema (Alarme)	Sistema P(risco)
1	1	0.82
0	0	0.75

Fonte: Autor

O gráfico de PRR precisa que os resultados sejam agrupados para a montagem do gráfico, quatro categorias são necessárias:

a) Verdadeiro Positivo (VP): Uma classificação VP se dá quando tanto o especialista quanto o sistema classificam a cena como alarme;
b) Verdadeiro Negativo (VN): Uma classificação VN se dá quando tanto o especialista quanto o sistema classificam a cena como não alarme;
c) Falso Positivo (FP): Uma classificação FP se dá quando o especialista não indica alarme enquanto o sistema classifica a cena como alarme;
d) Falso Negativo (FN): Uma classificação FN se dá quando o especialista indica alarme enquanto o sistema classifica a cena como não alarme.

A tabela verdade 3 exemplifica como será categorizado a classificação de cada quadro da sequência de quadros da experimentação.

Tabela 3-Tabela Verdade

Especialista	Sistema	Categoria do resultado
1	1	VP
0	0	VN
0	1	FP
1	0	FN

Fonte: Autor

Após todos os quadros classificados terem sido categorizados, deve-se calcular tanto o valor de precisão conforme a equação 5.1 tanto a revocação conforme a equação 22.

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

$$Revocação = \frac{\sum VP}{\sum VP + \sum FN} \tag{23}$$

O objetivo da precisão é calcular qual é a proporção de verdadeiros positivos entre todos os positivos, sejam verdadeiros ou falso, ou seja, qual foi a proporção de classificações de alarme que foram classificados corretamente sobre todos os alarmes. Já o objetivo do cálculo da revocação é entender qual é a proporção da classificação de alarmes que foram classificados corretamente sobre todas as ocorrências que deveriam ser alarme.

O gráfico de PRR é formado por um par dos valores de precisão e revocação, calculados de acordo com as equações 22 e 23 respectivamente, formando-se assim um ponto cartesiano sendo que no eixo *x* refere-se a revocação e o eixo *y* precisão.

Para se obter a curva, é necessário que se varie algum parâmetro do sistema em vários níveis para que se obtenha pontos suficientes para gerar o gráfico de PRR.

Neste trabalho, o parâmetro que será variado é o limiar que determina a partir de qual probabilidade de risco o alarme é acionado. Foram determinados 10 limiares iniciando em 0.1 até 0.9, além do 0.85.

Para poder avaliar qual foi o limiar que apresentou melhor performance, será utilizado o *F1 Score*, que é a média harmônica entre o par precisão e revocação, e é representado na 24. Para este trabalho, o melhor balanço entre precisão e revocação é dado pelo maior valor *do F1 Score*.

$$F1Score = 2\frac{precisãoxrevocação}{precisão + revocação} \tag{24}$$

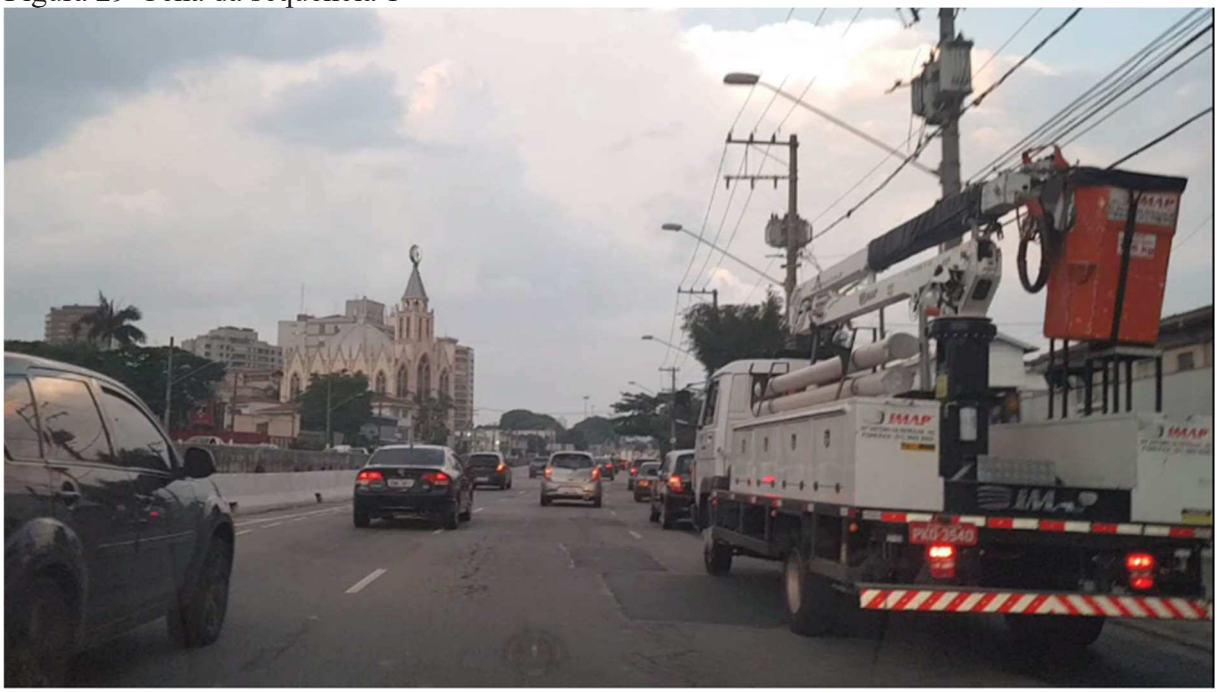
Os gráficos de PRR foram gerados com o auxílio da biblioteca *sklearn* do Python.

A partir da próxima seção serão apresentados os dados obtidos pelo sistema em duas diferentes cenas de tráfego de veículos.

5.2 SEQUÊNCIA 1

A sequência1 representa uma condição de trânsito de tráfego congestionado, existe fila de carros em todas as faixas de rolagem, como pode ser visto em uma cena extraída da sequência1 na figura 29. Esta sequência de quadros apresenta 224 cenas consecutivas.

Figura 29-Cena da sequência 1



Fonte: Autor

A classificação das 224 cenas pelo especialista é mostrada na tabela 6.

Tabela 4-Classificação do Especialista

	Alarme	Não Alarme
Quantidade	171	53
%	76.3	23.7

Fonte: Autor

Os resultados a seguir foram obtidos através da variação do limiar de alarme.

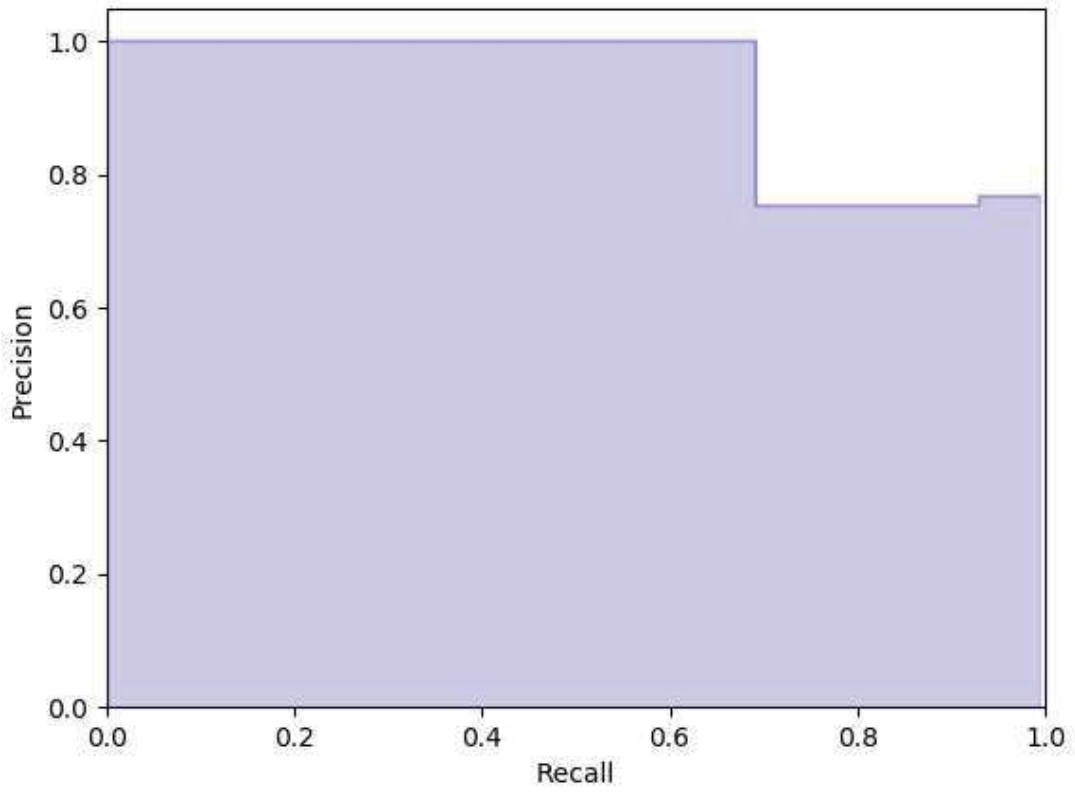
Na tabela 5 são apresentados os resultados de PRR e F1 Score para as 224 cenas e os 10 limiares.

Tabela 5-Resultado Sequência 1

Limiar	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.85	0.90
VP	171	171	171	160	160	160	160	155	0	0
FP	52	52	52	52	52	52	25	0	0	0
FN	0	0	0	11	11	11	11	16	171	171
VN	1	1	1	1	1	1	1	53	53	53
Precisão	0.766	0.766	0.766	0.754	0.754	0.754	0.754	1	-	-
Revocação	1	1	1	0.935	0.935	0.935	0.935	0.906	0	0
F_1 Score	0.868	0.868	0.868	0.835	0.835	0.835	0.835	0.950	0	0

Fonte: Autor

Figura 30– Precisão x Revocação: Sequência 1



Fonte: Autor

O resultado do sistema para a sequência 1 mostra que o limiar em que o sistema tem melhor performance de avaliar a possibilidade de um veículo estar na área de risco foi 0.80, pois foi o que atingiu o melhor F1 Score. Isso demostra que o sistema apresentou um resultado bem próximo ao resultado do especialista, e adicionalmente todas as situações de alarme classificadas pelo sistema nenhuma era alarme falso (FN), ou seja, precisão 1. Mesmo com uma precisão alta não houve uma queda significativa na revocação. O resultado mostrou que entre

todas as cenas que continham a possibilidade de um veículo estar na área de risco na sequência 1, 90,6% das cenas em que existia a possibilidade de um veículo estar na área de risco foram identificadas pelo sistema.

Mesmo que o limiar escolhido fosse o limite mínimo, a precisão não foi reduzida drasticamente, uma vez que 76,6% dos alarmes ainda são verdadeiros (VP). Isso significa que mesmo o que condutor faça esta alteração o sistema não irá reagir gerando alarme em todas as situações.

5.3 SEQUÊNCIA 2

A sequência 2 representa uma condição de trânsito de tráfego com bastante carros, existem carros em todas as faixas de rolagem, como pode ser visto em uma cena extraída da sequência 2 na figura 31. Esta sequência de quadros apresenta 368 cenas consecutivas.

Figura 31-Cena da sequência 2

Fonte: Autor

A classificação das 368 cenas pelo especialista é mostrada na tabela 6.

Tabela 6-Classificação do Especialista

	Alarme	Não Alarme
Quantidade	58	310
%	15.7	84.3

Fonte: Autor

Os resultados a seguir foram obtidos através da variação do limiar de alarme.

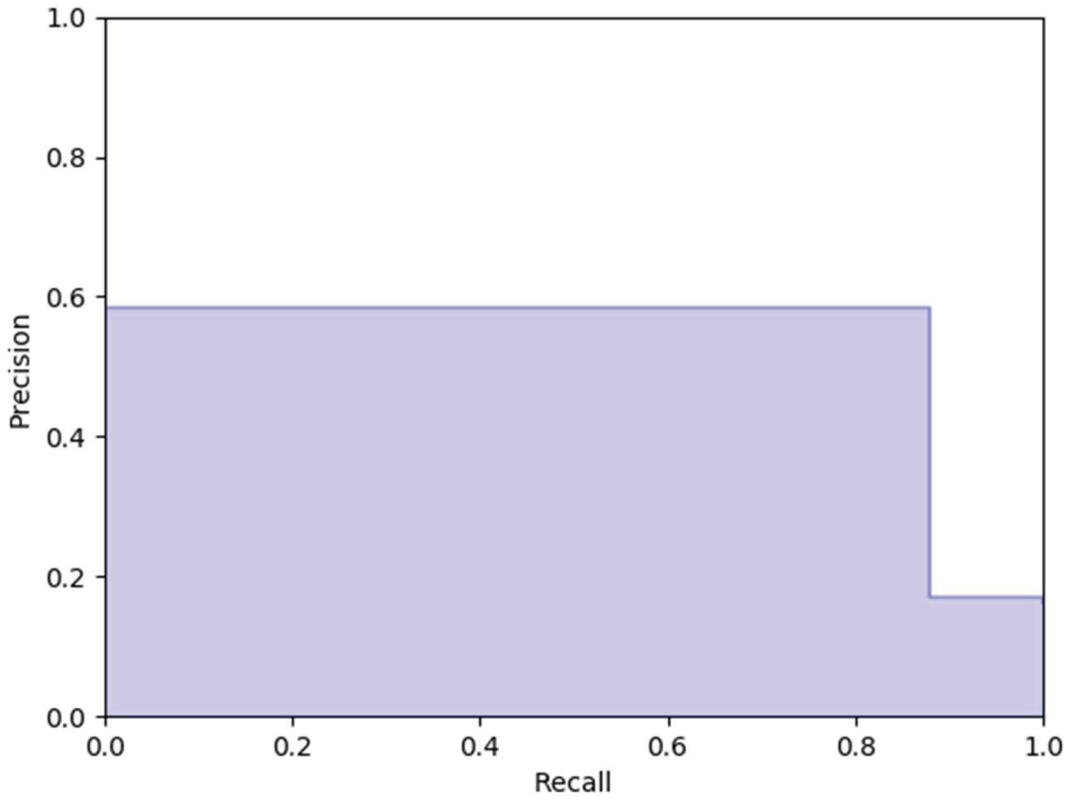
Na tabela 7 está apresentado os resultados de PRR e F1 Score para as 368 cenas e os 10 limiares.

Tabela 7-Resultado Sequência 2

Limiar	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.85	0.90
VP	58	58	58	58	58	58	58	51	51	51
FP	297	297	280	280	280	280	279	37	36	36
FN	0	0	0	0	0	0	0	7	7	7
VN	13	13	30	30	30	30	31	273	274	274
Precisão	0.163	0.163	0.171	0.171	0.171	0.171	0.172	.579	0.586	0.586
Revocação	1	1	1	1	1	1	1	0.879	0.879	0.879
F_1 Score	0.280	0.280	0.292	0.292	0.292	0.292	0.293	0.698	0.703	0.703

Fonte: Autor

Figura 32-Precisão x Revocação: Sequência 2



Fonte: Autor

O resultado do sistema para a sequência 2 mostra que o limiar em que o sistema tem melhor performance de avaliar a possibilidade de um veículo estar na área de risco foi 0.85, pois foi que atingiu o melhor F_1 Score.

A precisão para o limiar de 0.85 foi de 0.586, que significa que 58,6% das cenas que continham a possibilidade de um veículo estar na área de risco indicadas pelo sistema eram realmente situações de risco, houve 41,4% de alarme falso. O valor de revocação para a sequência 2 mostra que 87,9% das cenas que continham a possibilidade de um veículo estar na área de risco identificadas pelo especialista, também foram identificadas pelo sistema.

Diferentemente da sequência 1, a redução do limiar de risco na sequência 2 mostra uma redução de performance do sistema. A precisão cai para 16,3% no limiar de 0.1. É importante destacar também que na sequência 2, os dados classificados pelo especialista apresentavam mais de 84% das cenas classificadas como não alarme, portanto quando variado o limiar para um valor inferior ao do limiar do especialista (0.75) era esperado que mais FP fossem identificados.

Para os limiares de acima de 0.8 pode-se notar que ainda houve aproximadamente 10% de FP na sequência 2, isso deveu-se a ao sistema não conseguir distinguir veículos que estão em movimento de veículos que estão estacionados. Na figura 33 é possível ver um veículo estacionado, isso levou o sistema a entender que havia um veículo que fazia parte da cena, na região de discretização inferior direita. Portanto a inferência do risco dado a existência desse veículo é mais alta do que a mesma cena sem a identificação do veículo o que gerou um FP.

Figura 33-Veículo estacionado

Fonte: Autor

6 CONCLUSÃO

Neste trabalho foi apresentado um modelo de dados entre redes neurais convolucionais e um sistema lógico probabilístico, com o objetivo de se obter um sistema de assistência ao motorista de veículos que previna ou mitigue o risco de colisão. Esses sistemas já estão atualmente em veículos vendidos no mercado, e em certo grau serão obrigatórios em veículos na comunidade Europeia e Estados Unidos a partir de 2020.

A criação da região segura, através da transformada de Hough apresentou alguns problemas devido principalmente em relação a imperfeições na pintura das faixas de rolagem e também identificou algumas imperfeições da pista, o que gerou alguma variação na região de detecção, que não gerou de fato nenhum resultado negativo para o sistema, porém o sistema não foi avaliado em muitas condições de iluminação e qualidade da pintura das faixas. Mais adiante na seção de trabalhos futuros é apresentado algumas melhorias que podem ser implementadas.

A utilização de RNC para segmentação e classificação de veículos se mostrou muito eficiente, apesar do objetivo deste trabalho não ser a avaliação da eficiência da RNC implementada, durante os testes e analises não houve quaisquer problemas em termos de classificação, uma única ressalva foi o tamanho do polígono gerado para segmentar os veículos. Ele apresentava tamanho aproximado de 30% maior que o objeto, ele foi corrigido matematicamente dentro do sistema. Esse aumento de tamanho não interferiu no resultado do teste porque a discretização foi feita através do centro do objeto detectado, mas para aplicações onde a borda é importante pode ser tornar um problema.

A principal conclusão que este trabalho teve foi mostrar que é promissora a pesquisa de soluções para sistemas ADAS, que incorporam o uso de raciocínio probabilístico. A utilização do ProbLog para este tipo de aplicação é apropriada por conseguir com poucas linhas de código modelar um contexto de situações de risco conhecidas, outro fator importante é que as relações condicionais entre duas situações não precisam ser codificadas em uma nova sentença, o ProbLog realiza a inferência condicional das duas, o que significa que algumas sentenças simples conseguem modelar um contexto condicional mais complexo.

Para poder comparar de forma bem sintética o Problog com Redes Lógicas de Markov (RLM) a seguir é descrito brevemente sua definição. As RLM são uma coleção de fórmulas de lógica em primeira ordem que estão associadas a pesos, esses pesos são números reais. Uma RLM tem como ideia básica reduzir as restrições que são impostas por uma base de conhecimento. O objetivo de determinar pesos para cada fórmula é que quando um mundo não

satisfaz todas as fórmulas, ele não seja impossível como em lógica de primeira ordem, mas tenha uma probabilidade menor de ocorrer.

SZTYLER, et al. (2018) compara soluções que utilizam RLM com Problog. MLN conforme apresentado pro SZTYLER, et al. (2018) utiliza-se de pesos no modelamento das probabilidades e requer um grande esforço de um especialista para obtê-las, enquanto do Problog cada sentença recebe explicitamente uma probabilidade. A grande vantagem na inferência do Problog em comparação com RML está associada a hipótese de mundo fechado no Problog, o que resulta em melhores tempos de resposta na inferência.

A avaliação da sequência 1 mostra que uma cena com muita informação teve um bom resultado como visto no capítulo anterior, apesar da detecção da faixa de rolagem ter sido pior devido a manutenção de buracos sem que houvesse nova pintura das faixas de rolagem, conforme pode ser visto na figura 29, o resultado da inferência do risco foi a esperada.

Quando comparada com a cena 2, a vantagem da cena 1 ter mais informações ajuda a evitar um ruído inesperado e não planejado, que foram carros estacionados como pode ser observado no destaque em vermelho na figura 33. Os ruídos dos carros estacionados geraram muitos FP na análise, porque na regra de boa conduta do sistema, um carro a direita do ego veículo é considerado com mais risco por existir uma probabilidade de se tratar de uma ultrapassagem pela direita e essa manobra ser proibida pelo código brasileiro de trânsito. Uma possível solução, seria reduzir a área de interesse da segmentação, e com isso eliminar esse veículo da análise. Outra variável que está presente na cena 2 em alguns momentos são 4 faixas de rolagem, o que também contribuiu para haver o ruído do carro estacionado.

O trabalho ficou restrito a dados gerados por apenas duas cenas, isso deve-se a necessidade de mais tempo para obtenção das cenas, como também um tempo para a anotação de cada quadro da cena. Por se tratar de um trabalho sem apoio de bolsas e com o Autor sendo a única mão de obra disponível, trabalhando em tempo parcial, não foi possível dentro do cronograma tempo para a obtenção de mais dados.

6.1 TRABALHOS FUTUROS

Para os próximos trabalhos, nas técnicas de visão computacional, seria importante que a parte do sistema que realiza a identificação das faixas de rolagem seja aprimorado, pois foi necessário procurar trechos de avenidas onde havia quase zero problemas de sinalização da via, a utilização de transformação da perspectiva é uma solução que também pode ser parte de uma

nova etapa, uma vez que auxiliaria na identificação das faixa de rolamento em curvas e redução da distorção causada pela captura da sequência de quadros pela câmera. A sequência de quadros foi capturada de dentro do ego veículo, e isso causou em várias situações reflexos gerados pelo para brisas dependendo da incidência da luz do sol. Neste caso seria importante que fosse possível a captura da imagem com a câmera do lado externo e com o uso de uma lente para ajudar a filtrar a incidência direta de luz sobre a lente.

Como mostrado no início do trabalho, os sistemas ADAS atualmente utilizam-se de fusão de dados para melhorar e aprimorar a detecção de situações de risco de colisão, portanto nos próximos trabalhos a implementação de parâmetros do carro como velocidade e ângulo do esterçamento do ego veículos contribuiriam para aumentar os dados disponíveis, e com isso poderiam ser propostas outras regras lógicas utilizando-se desses parâmetros, e portanto abranger mais situações de risco.

A inclusão de soluções que utilizam-se raciocínio temporal em próximos trabalhos, como por exemplo filtros de Kalman, que apareceram na revisão bibliográfica, modelos ocultos de Markov ou redes Bayesianas dinâmicas, poderia contribuir em propor modelos de transição temporal de ações dos veículos da cena, e usá-los como evidência e parâmetros para os sistema lógico probabilístico. Como o tráfego de veículos é dinâmico, a continuidade de uma ação do passado, pode fazer parte de uma ação real futura. Por exemplo, uma troca de faixa de rolamento por um carro que se iniciou dentro da faixa, mas que se mantém até que o automóvel cruze a faixa de rolamento.

REFERÊNCIAS

BENGLER, K. et al. Three Decades of Driver Assistance Systems: Review and Future Perspectives. **IEEE Intelligent Transportation Systems Magazine**, v. 6, p. 6-22, 2014. ISSN ISSN: 1939-1390.

CANNY, J. A Computational Approach to Edge Detection. **IEEE Trans. Pattern Anal. Mach. Intell.**, Washington, v. 8, p. 679-698, jun. 1986. ISSN ISSN: 0162-8828. Disponivel em: <https://doi.org/10.1109/TPAMI.1986.4767851>.

COHN, A. G. et al. Cognitive Vision: Integrating Symbolic Qualitative Representations with Computer Vision. In: CHRISTENSEN, H. I.; NAGEL, H.-H. **Cognitive Vision Systems:** Sampling the Spectrum of Approaches. Berlin: Springer Berlin Heidelberg, 2006. p. 221-246. ISBN ISBN: 978-3-540-33972-4. Disponivel em: <https://doi.org/10.1007/11414353_14>.

CORDTS, M. et al. The Cityscapes Dataset for Semantic Urban Scene Understanding. **CoRR**, v. abs/1604.01685, 2016. Disponivel em: <http://arxiv.org/abs/1604.01685>.

COUCHOT, J.-F. et al. Steganalysis via a Convolutional Neural Network using Large Convolution Filters. **CoRR**, v. abs/1605.07946, 2016. Disponivel em: <http://arxiv.org/abs/1605.07946>.

DAI, X. et al. Temporal Context Network for Activity Localization in Videos. **CoRR**, v. abs/1708.02349, 2017. Disponivel em: <http://arxiv.org/abs/1708.02349>.

DENG, J. et al. **ImageNet:** A Large-Scale Hierarchical Image Database. CVPR09. [S.l.]: [s.n.]. 2009.

DENG, J. et al. **ImageNet:** A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition. [S.l.]: [s.n.]. jun. 2009. p. 248-255.

DENG, L.; YU, D. Deep Learning: Methods and Applications. **Foundations and Trends® in Signal Processing**, v. 7, p. 197-387, 2014. ISSN ISSN: 1932-8346. Disponivel em: <http://dx.doi.org/10.1561/2000000039>.

DRIES, A. et al. **ProbLog2:** Probabilistic Logic Programming. Machine Learning and Knowledge Discovery in Databases. Cham: Springer International Publishing. 2015. p. 312-315.

DUBBA, K. S. R. et al. Learning Relational Event Models from Video. **J. Artif. Intell. Res.**, v. 53, p. 41-90, 2015.

DUDA, R. O.; HART, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. **Commun. ACM**, New York, NY, USA, v. 15, p. 11-15, jan. 1972. ISSN ISSN: 0001-0782. Disponivel em: <http://doi.acm.org/10.1145/361237.361242>.

FETH, P. et al. Dynamic Risk Assessment for Vehicles of Higher Automation Levels by Deep Learning. **CoRR**, v. abs/1806.07635, 2018. Disponivel em: <http://arxiv.org/abs/1806.07635>.

FIERENS, D. et al. **Inference and learning in probabilistic logic programs using weighted Boolean formulas. CoRR,** v. abs/1304.6810, 2013. Disponivel em: <http://arxiv.org/abs/1304.6810>.

FRITSCH, J.; KUEHNL, T.; GEIGER, A. **A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms**. International Conference on Intelligent Transportation Systems (ITSC). [S.l.]: [s.n.]. 2013.

GEIGER, A.; LENZ, P.; URTASUN, R. **Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite**. Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.]: [s.n.]. 2012.

GIBSON, A.; PATTERSON, J. **Deep Learning**. [S.l.]: O'Reilly Media, Inc., 2017. ISBN ISBN: 9781491924570. https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing (3rd Edition)**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN ISBN: 013168728X.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

HAFNER, M. R. et al. Cooperative Collision Avoidance at Intersections: Algorithms and Experiments. **IEEE Transactions on Intelligent Transportation Systems**, v. 14, p. 1162-1175, set. 2013. ISSN ISSN: 1524-9050.

HAGAN, M. T. et al. **Neural Network Design (2nd Edition)**. [S.l.]: eBook, 2014. ISBN ISBN: 978-0-9717321-1-7. http://hagan.okstate.edu/nnd.html.

INOUE, N.; KURIYA, Y.; KOBAYASHI, I. K. Recognizing potential traffic risks through logic-based deep scene understanding, 2015.

JANHUNEN, T. **Representing Normal Programs with Clauses**. Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004. [S.l.]: [s.n.]. 2004. p. 358-362.

KARPATHY, A. et al. **Large-Scale Video Classification with Convolutional Neural Networks**. Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Washington: IEEE Computer Society. 2014. p. 1725-1732.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F., et al. **Advances in Neural Information Processing Systems 25**. [S.l.]: Curran Associates, Inc., 2012. p. 1097-1105. Disponivel em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

KUKKALA, V. K. et al. Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles. **IEEE Consumer Electronics Magazine**, v. 7, p. 18-25, set. 2018. ISSN ISSN: 2162-2248.

LE, Q. V. et al. **Learning Hierarchical Invariant Spatio-temporal Features for Action Recognition with Independent Subspace Analysis**. Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. Washington: IEEE Computer Society. 2011. p. 3361-3368.

LI, Z. et al. VideoLSTM convolves, attends and flows for action recognition. **Computer Vision and Image Understanding**, v. 166, p. 41-50, 2018. ISSN ISSN: 1077-3142. Disponivel em: <http://www.sciencedirect.com/science/article/pii/S1077314217301741>.
LIN, T.-Y. et al. Microsoft COCO: Common Objects in Context. **CoRR**, v. abs/1405.0312, 2014. Disponivel em: <http://arxiv.org/abs/1405.0312>.

MANTADELIS, T.; JANSSENS, G. **Dedicated Tabling for a Probabilistic Setting**. Technical Communications of the 26th International Conference on Logic Programming. Dagstuhl: Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik. 2010. p. 124-133.

MOGELMOSE, A.; TRIVEDI, M. M.; MOESLUND, T. B. Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey. **Trans. Intell. Transport. Sys.**, Piscataway, v. 13, p. 1484-1497, dez. 2012. ISSN ISSN: 1524-9050. Disponivel em: <http://dx.doi.org/10.1109/TITS.2012.2209421>.

PRABU, U. **Survey on Collision Avoidance in VANET**. [S.l.]: [s.n.]. mar. 2015.
PYO, J.; BANG, J.; JEONG, Y. **Front collision warning based on vehicle detection using CNN**. 2016 International SoC Design Conference (ISOCC). [S.l.]: [s.n.]. out. 2016. p. 163-164.

RAVANBAKHSH, M. et al. Action Recognition with Image Based CNN Features. **CoRR**, v. abs/1512.03980, 2015. Disponivel em: <http://arxiv.org/abs/1512.03980>.
REDMON, J. et al. **You Only Look Once:** Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. [S.l.]: [s.n.]. 2016. p. 779-788.

REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **CoRR**, v. abs/1506.01497, 2015. Disponivel em: <http://arxiv.org/abs/1506.01497>.

ROADMAP, E. N. C. A. P. 2. **Euro NCAP 2025 Roadmap:** IN PURSUIT OF VISION ZERO. [S.l.]: [s.n.]. 2017.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **PLOS ONE**, v. 10, p. 1-21, mar. 2015. Disponivel em: <https://doi.org/10.1371/journal.pone.0118432>.

SCHUBERT, R.; RICHTER, E.; WANIELIK, G. **Comparison and evaluation of advanced motion models for vehicle tracking**. 2008 11th International Conference on Information Fusion. [S.l.]: [s.n.]. jun. 2008. p. 1-6.

SHTERIONOV, D. **Design and Development of Probabilistic Inference Pipelines**. KU Leuven – Faculty of Engineering, Department of Computer Science. [S.l.]. 2015.

SIMONYAN, K.; ZISSERMAN, A. Two-Stream Convolutional Networks for Action Recognition in Videos. In: GHAHRAMANI, Z., et al. **Advances in Neural Information Processing Systems 27**. [S.l.]: Curran Associates, Inc., 2014. p. 568-576. Disponivel em: <http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf>.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **CoRR**, v. abs/1409.1556, 2014. Disponivel em: <http://arxiv.org/abs/1409.1556>.

SIVARAMAN, S.; TRIVEDI, M. M. A General Active Learning Framework for On-road Vehicle Recognition and Tracking, 2010. Disponivel em: <http://cvrr.ucsd.edu/publications/2010/sivaraman_ITS10.pdf>.

SOUZA, C. R. C.; SANTOS, P. E. **Probabilistic Logic Reasoning about Traffic Scenes**. Towards Autonomous Robotic Systems. Berlin: Springer Berlin Heidelberg. 2011. p. 219-230.

SZEGEDY, C. et al. Going Deeper with Convolutions. **CoRR**, v. abs/1409.4842, 2014. Disponivel em: <http://arxiv.org/abs/1409.4842>.

SZTYLER, T et al. **Modeling and Reasoning with ProbLog: An Application in Recognizing Complex Activities** 259-264. 10.1109/PERCOMW.2018.8480299. Disponivel em: < https://homes.di.unimi.it/civitarese/publications/18-CoMoRea.pdf>.

TAKUMA NAKAMORI, M. I. . N. I. . N. Vehicle front scene watching from the sequence of road images. **SPIE Optics + Photonics**, 2002. Disponivel em: <https://doi.org/10.1117/12.452137>.

TAYLOR, W. et al. **Convolutional Learning of Spatio-temporal Features**. Proc. European Conference on Computer Vision (ECCV'10). [S.l.]: [s.n.]. 2010.

WANG, J.; ZHAO, P.; HOI, S. C. H. **Exact Soft Confidence-weighted Learning**. Proceedings of the 29th International Coference on International Conference on Machine Learning. USA: Omnipress. 2012. p. 107-114.

WANG, L.; QIAO, Y.; TANG, X. **Action Recognition With Trajectory-Pooled Deep-Convolutional Descriptors**. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.]: [s.n.]. jun. 2015.