UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

Yan Victor Ribeiro Marim

DETECÇÃO DE OBJETOS: ESTUDO E APLICAÇÃO DA ARQUITETURA R-CNN

Vitória-ES

Julho/2019

Yan Victor Ribeiro Marim

DETECÇÃO DE OBJETOS: ESTUDO E APLICAÇÃO DA ARQUITETURA R-CNN

Parte manuscrita do Projeto de Graduação do aluno Yan Victor Ribeiro Marim, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Julho/2019

Yan Victor Ribeiro Marim

DETECÇÃO DE OBJETOS: ESTUDO E APLICAÇÃO DA ARQUITETURA R-CNN

Parte manuscrita do Projeto de Graduação do aluno Yan Victor Ribeiro Marim, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 18 de Julho de 2019.

COMISSÃO EXAMINADORA:

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador

Msc. Matheus Vieira Lessa Ribeiro
Universidade Federal do Espírito Santo
Coorientador

Prof. Dra. Raquel Frizera Vassallo
Universidade Federal do Espírito Santo
Examinador

Prof. Msc. Fabio Ricardo Oliveira Bento
Instituto Federal do Espírito Santo
Examinador

Vitória-ES

Julho/2019

AGRADECIMENTOS

Aos meus pais, pelo apoio e dedicação a mim, não só durante toda a graduação, como durante toda a minha vida.

À minha namorada por toda compreensão e apoio durante toda a graduação.

Ao meu orientador, Jorge Leonid Aching Samatelo, e ao meu coorientador, Matheus Vieira Lessa Ribeiro, por toda ajuda, orientação e dedicação durante o desenvolvimento deste trabalho.

À banca examinadora por aceitar o convite e pelo tempo investido para leitura e avaliação deste trabalho.

RESUMO

A Visão Computacional é uma área da Inteligência Artificial que visa dar aos computadores um entendimento visual do mundo. Este entendimento visual do mundo tem como um de seus obstáculos a capacidade de detectar objetos a partir de imagens. O problema de detecção de objetos consiste em dois problemas distintos: localização e classificação de um objeto. A partir de 2012 houve um grande avanço na resolução do problema de classificação de imagens através da utilização de redes neurais convolucionais profundas, permitindo que soluções para o problema de detecção de objetos sejam aprimoradas. Este trabalho se propõe a aplicar técnicas de *Deep Learning*, baseadas em redes neurais convolucionais, ao problema de detecção de objetos, especificamente, a detecção de veículos presentes em imagens de monitoramento de rodovias. Para isso, são utilizadas técnicas de pré-processamento de imagens e *Transfer Learning* (tradução livre, transferência de aprendizado). Além disso, é realizado um estudo aprofundado sobre a arquitetura de detecção utilizada (R-CNN), detalhando as etapas necessárias para que a detecção seja concluída. Por fim, é obtida uma acurácia de 86% na classificação de regiões propostas nas imagens do banco de dados UA-DETRAC, considerando as classes de interesse e uma classe adicional, destinada para regiões sem veículos.

Palavras-chave: *Deep-learning*; Redes neurais convolucionais; *Transfer-learning*, detecção de objetos.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Network*
BB	*Bounding Box*
CNN	*Convolutional Neural Network*
DL	*Deep Learning*
FPS	*Frames Per Second*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
IOU	*Intersection Over Union*
ML	*Machine Learning*
NMS	*Non-Maximum Suppression*
R-CNN	*Regions with Convolutional Neural Networks*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*
UFES	Universidade Federal do Espírito Santo
VC	Visão Computacional

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação e Objeto de Pesquisa

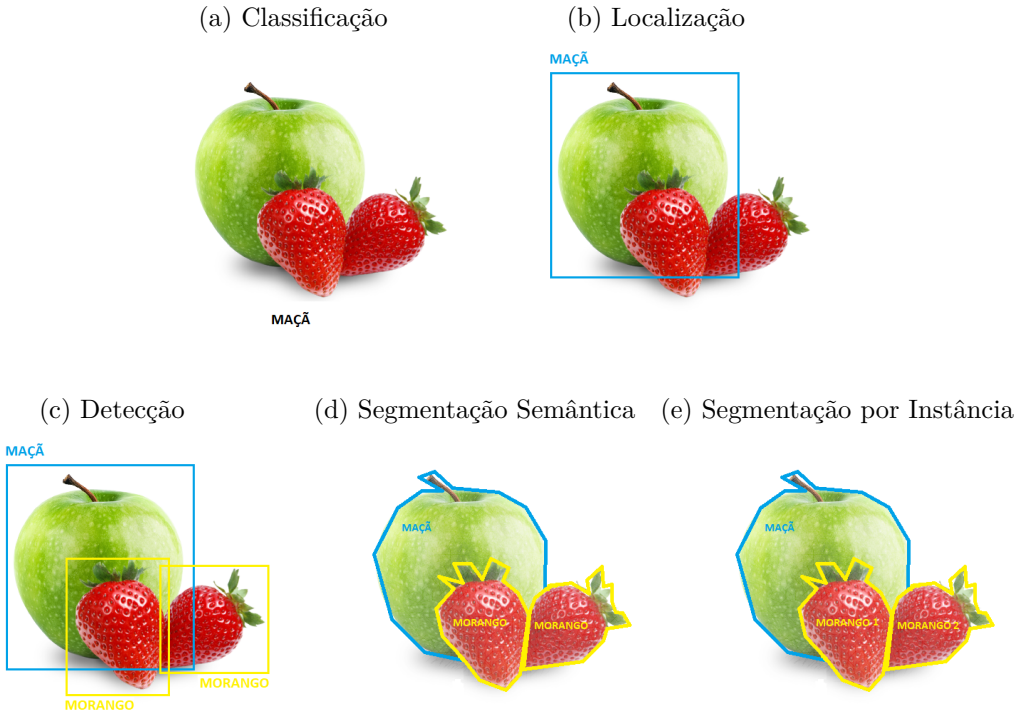
Visão Computacional - VC é uma área da Inteligência Artificial - IA que visa dar aos computadores um entendimento visual do mundo (MARENGONI; STRINGHINI, 2009). O objetivo da VC é simular a visão humana usando imagens digitais. Um sistema de VC está conformado pelas seguintes etapas, segundo (NIXON; AGUADO, 2012): (*i*) a etapa de aquisição da imagem é a conversão de uma cena do mundo real para dados binários interpretados como imagens digitais. Este processo é realizado por dispositivos de aquisição de imagem digitais, como câmeras digitais, *webcams*, entre outros; (*ii*) a etapa de processamento da imagem é a extração de características relevantes da imagem previamente adquirida, por exemplo, contornos, formas e segmentação; (*iii*) na etapa de análise e entendimento de imagem, utilizando as imagens previamente processadas, espera-se que o computador tenha a capacidade de realizar operações a partir das características extraídas, por exemplo, detectar objetos, seguir o movimento de um objeto em um vídeo.

Considerando que a visão humana reflete diretamente na sua capacidade de tomada de decisão, fornecer um entendimento visual do mundo aos computadores lhes concederia uma melhor capacidade de tomada de decisão. O entendimento visual do mundo de humanos e máquinas passa pela capacidade de detectar objetos a partir de imagens (LIN et al., 2014). Entretanto, o problema de detecção de objetos pode ser abordado de diversas formas. Algumas dessas abordagens estão listadas a seguir: (*i*) classificação, em que uma imagem é classificada pelo objeto mais relevante presente; (*ii*) localização, em que se localiza um único objeto em uma imagem; (*iii*) detecção, localiza e classifica vários objetos em uma imagem; (*iv*) segmentação semântica, associa cada pixel de uma imagem a uma classe de objeto; (*v*) segmentação por instância, associa cada pixel de uma imagem a uma instância de objeto. A Figura 1, ilustra as diferentes abordagens descritas.

A proposta deste trabalho é abordar o problema de detecção de objetos via *Deep Learning* - DL (tradução livre, aprendizado profundo), uma área de pesquisa relacionada a *Machine Learning* - ML (tradução livre, aprendizado de máquina), que torna possível o uso de modelos computacionais compostos de várias camadas de processamento e que aprendem a representar os dados de entrada em diversos níveis de abstração (LECUN; BENGIO; HINTON, 2015).

Dentro da área de DL as arquiteturas de redes neurais profundas de maior importância são as *Convolutional Neural Networks* - CNN (tradução livre, redes neurais convolucionais) propostas em 1998 por Yann LeCun (LECUN et al., 1998a). Estas arquiteturas foram

Figura 1 – Tipos de identificação de objetos



Fonte: Produção do próprio autor.

aplicadas ao problema de classificação de dígitos manuscritos, obtendo uma taxa de classificação de $99,7\%$ sobre o banco de dados MNIST (WAN et al., 2013). Com o passar dos anos, o aumento do poder computacional das máquinas permitiu que esta arquitetura fosse utilizada no *Imagenet Large Scale Visual Recognition Challenge* - ILSVRC (RUSSAKOVSKY et al., 2014), obtendo uma taxa de erro de classificação $10,9\%$ menor que o segundo qualificado.

Neste trabalho será abordada uma arquitetura de detecção de objetos baseada em CNN : a *Regions with Convolutional Neural Networks* - R-CNN (GIRSHICK et al., 2014). O trabalho será focado no estudo e implementação dessa arquitetura. Posteriormente, a arquitetura será avaliada utilizando como caso de estudo o problema de detecção de veículos em *frames* de sequências de vídeo. A base de dados que será utilizada para o treinamento e validação da arquitetura implementada será a base de dados UA-DETRAC[1], apresentado por (WEN et al., 2015).

[1] Disponível em *http://detrac-db.rit.albany.edu/*

1.2 Objetivos

Objetivo Geral

- Este trabalho tem como objetivo geral o estudo e implementação da arquitetura R-CNN, proposta por (GIRSHICK et al., 2014), para o problema de detecção de objetos. Além disso, pretende-se aplicar essa arquitetura ao problema de detecção de veículos em *frames* de sequências de vídeo.

Objetivos Específicos

- Realizar uma revisão bibliográfica de trabalhos que estudam o problema de detecção de objetos baseadas em arquiteturas de redes neurais CNN;
- Estudar e implementar a arquitetura R-CNN na detecção de objetos;
- Obter um banco de vídeos de fluxo veicular para a avaliação das arquiteturas implementadas;
- Validar e testar o sistema implementado na detecção de veículos;

1.3 Justificativa

Alguns dos trabalhos relacionados a este projeto disponíveis na literatura são apresentados a seguir.

- ***Empirical Evaluation of Deep Learning on Highway Driving*** (HUVAL et al., 2015) é um trabalho em que os autores obtêm a própria base de dados a partir de gravações feitas por uma câmera instalada dentro de um veículo. Sob essa base de dados, aplica-se uma arquitetura de CNN, *Overfeat CNN detector*, para detecção de veículos e as faixas de sinalização das pistas em rodovias.
- ***Evolving Boxes for fast Vehicle Detection*** (WANG et al., 2017) aplica o *framework* de DL, *Evolving Boxes*, para detecção de veículos em uma base de dados composta por *frames* de uma câmera de monitoramento de trânsito. Atingindo tempos de execução entre 9 e 13 FPS em uma GPU comercial.

- ***Vehicle detection from 3d lidar using fully convolutional network*** (LI; ZHANG; XIA, 2016) aplica DL para detecção de veículos a partir de dados obtidos por sensores LIDAR (SCHWARZ, 2010). Obtendo resultados similares ao estado da arte para este tipo de aplicação.

- ***Deep learning approach for car detection in UAV imagery*** (AMMOUR et al., 2017) apresenta uma solução para detecção e contagem de carros em imagens capturadas por UAVs - *Unmanned Aerial Vehicles*. As imagens são segmentadas para geração de regiões candidatas a serem carros, são extraídas características das regiões candidatas via CNN e, por fim, as regiões são classificadas via *Support Vector Machines* - SVM. O método obteve acurácia próxima a 90% em algumas imagens, superando o estado da arte para detecção de veículos neste tipo de imagem.

Percebe-se que os trabalhos anteriores são focados ao tipo de ambiente e a resolução do problema específico abordado (HUVAL et al., 2015)(LI; ZHANG; XIA, 2016)(AMMOUR et al., 2017) ou utilizam arquiteturas de DL fora do escopo deste trabalho (WANG et al., 2017). Alternativamente, este trabalho se propõe a estudar genericamente o *pipeline* clássico de detecção de objetos utilizando a arquitetura precursora do desenvolvimento da detecção de objetos utilizando CNNs. Posteriormente, esta arquitetura é implementada utilizando o problema de detecção veicular como caso de estudo.

1.4 Estrutura do Texto

O presente trabalho está estruturado da seguinte maneira:

- **Introdução**: este capítulo inicial tem como objetivo contextualiar o trabalho e o problema aqui estudado, apresentando o problema e ideias iniciais sobre a solução do mesmo. Além disso, são apresentados os objetivos da realização deste projeto de graduação;

- **Embasamento Teórico**: este capítulo dedica-se a explicar a fundamentação teórica por trás dos conceitos utilizados neste trabalho.

- **Proposta**: este capítulo é dedicado à apresentação da solução proposta para o problema em estudo.

- **Resultados**: neste capítulo são apresentados o banco de dados, as métricas de desempenho usadas nos experimentos e os resultados obtidos.

- **Conclusões e Projetos Futuros**: no capítulo final deste trabalho são apresentadas as conclusões e os trabalhos futuros.

2 EMBASAMENTO TEÓRICO

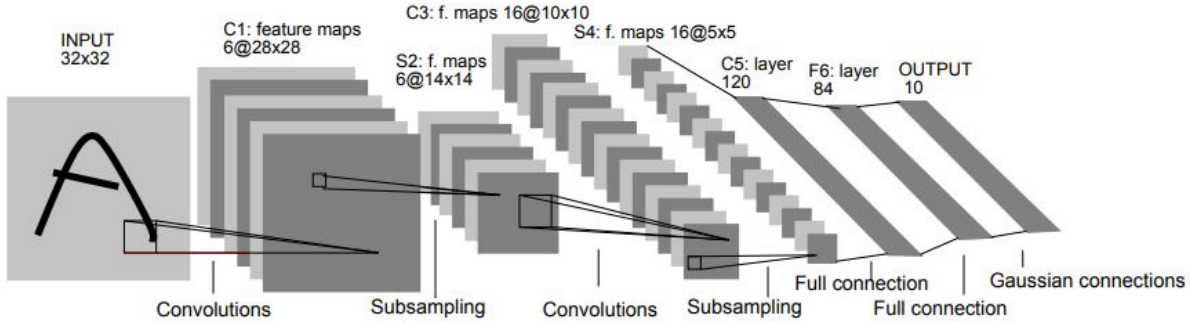
2.1 Introdução

Neste capítulo serão explicados os conceitos teóricos necessários para a realização deste trabalho. O capítulo inicia abordando o conceito de CNN e, em seguida, apresenta os componentes do *pipeline* para detecção de objetos. Ao final, é apresentada a técnica de detecção de objetos baseada em CNN que será implementada neste trabalho e o problema em que essa técnica será aplicada.

2.2 CNN

As CNN são redes neurais de aprendizado profundo baseadas no mecanismo de percepção visual dos seres vivos, onde a combinação hierárquica de uma estrutura de percepção simples pode funcionar como uma estrutura de percepção complexa (HUBEL; WIESEL, 1968). Um exemplo da estrutura básica de uma CNN é apresentado pela Figura 2.

Figura 2 – Exemplo da estrutura de uma CNN



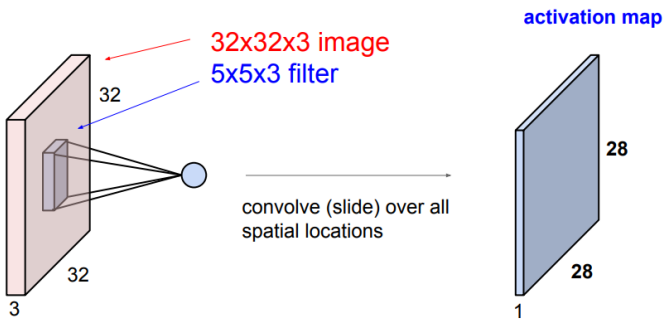
Fonte: LECUN et al.,1998a.

Os tópicos a seguir apresentam os componentes básicos de uma CNN que, ao serem combinados sequencialmente, formarão, por exemplo, uma estrutura de classificação de objetos complexa.

- **Camada de Convolução** : A camada de convolução é composta por diversos filtros,

também denominados *kernels*, que são aplicados à imagem de entrada. Cada filtro aplicado gera um mapa de característica, saída fornecida pela ativação resultante da imagem inteira por cada filtro, que ressalta alguma característica na imagem, por exemplo, o contorno dos objetos presentes (KARPATHY et al., 2016). O propósito desta camada é a extração de características de baixo nível da imagem via a operação de convolução. Cabe ressaltar que tanto os filtros quanto a imagem são representados computacionalmente como matrizes, logo, essa filtragem é realizada via operações matriciais. A Figura 3 exemplifica a operação realizada em uma camada de convolução.

Figura 3 – Obtenção do mapa de características via Camada de Convolução



Fonte: KARPATHY et al.,2016.

Formalmente, o valor da característica na localização (i, j) referente ao k-ésimo mapa de características, resultado das operações de cada filtro, da l-ésima camada, conjunto de filtros aplicados a uma mesma entrada, $z^l_{i,j,k}$ é calculado como:

$$z^l_{i,j,k} = \mathbf{w}^{l\,T}_k \mathbf{x}^l_{i,j} + b^l_k. \tag{2.1}$$

onde $\mathbf{x}^l_{i,j}$ é o fragmento da entrada centrada na posição (i, j) referente à l-ésima camada; $\mathbf{w}^l_k$ e b^l_k são o vetor de pesos e o termo de bias do k-ésimo filtro da l-ésima camada, respectivamente.

Em geral, é aplicada uma função de ativação não-linear aos resultados da convolução (KARPATHY et al., 2016). Embora tal função não seja obrigatória a uma camada de convolução, ela permite que a CNN detecte e aprenda características não-lineares, fazendo com que quase todas as camadas convolucionais sejam seguidas de uma função de ativação não-linear.

Seja $f(\bullet)$ a função de ativação não-linear, então, o valor que a função de ativação retorna quando a entrada é a característica convolucional $z^l_{i,j,k}$ pode ser calculado como:

$$a^l_{i,j,k} = f(z^l_{i,j,k}). \tag{2.2}$$

Funções de ativação comumente utilizadas são as funções sigmoide, tangente hiperbólico (LECUN et al., 1998b) e, mais recentemente, ReLU (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) (NAIR; HINTON, 2010).

- **Camada de *Pooling*** : A camada de *pooling* visa reduzir o tamanho dos mapas de características considerando a semelhança de valores vizinhos (KARPATHY et al., 2016). Esse objetivo é alcançado ao definir um novo mapa de característica através do *downsampling* do mapa de característica gerado pela camada de convolução. O *downsampling* é realizado ao associar um conjunto de valores vizinhos do mapa de característica anterior a um único valor no novo mapa de característica.

  Formalmente, seja a função de *pooling* escrita como $pool(\bullet)$, então, a saída gerada pela função de *pooling* para o mapa de características $\mathbf{a}^l_{:,:,k}$, é representada pela expressão:

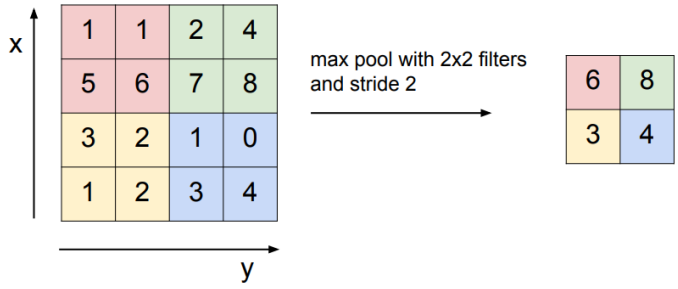
$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall\ (m,n) \in \mathcal{R}_{i,j} \tag{2.3}$$

  onde $\mathcal{R}_{i,j}$ é uma vizinhança local centrada na posição (i,j). As operações típicas de *pooling* são:

  - *max pooling*, que retorna apenas o maior valor dentro de seu campo receptivo (BOUREAU; PONCE; LECUN, 2010);
  - *average pooling*, que retorna a média dos valores do seu campo receptivo (WANG et al., 2012).

  A Figura 4 ilustra a operação de *max pooling*.

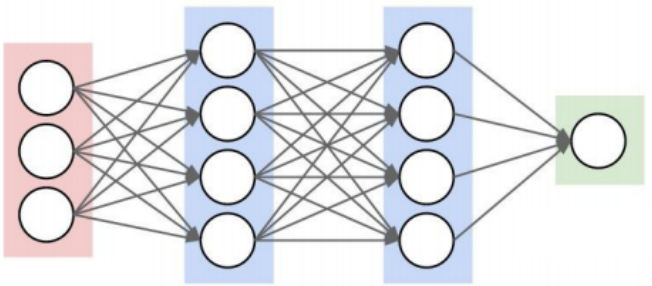
Figura 4 – *Max Pooling*



Fonte: KARPATHY et al.,2016.

- **Camadas totalmente conectadas** : As *Fully Connected Layers* (tradução livre, camadas totalmente conectadas) são camadas de uma *Artificial Neural Network* - ANN (NIELSEN, 2015) nas quais cada nó de uma camada está conectado a todos os outros nós da camada subsequente. A Figura 5 apresenta uma representação de *Fully Connected Layers*.

Figura 5 – *Fully Connected Layers*



Fonte: KARPATHY et al.,2016.

As características extraídas anteriormente pelas diversas camadas de convolução e *pooling* servirão de entrada para as camadas totalmente conectadas que irão interpretar essas características de alto nível de abstração e realizar operações complexas, por exemplo, classificar objetos (YOSINSKI et al., 2014).

Após as operações nas camadas totalmente conectadas tem-se a camada de saída. O tamanho da camada de saída, ou seja, o número de neurônios, é igual ao número de classes no problema. Além disso, cada neurônio de saída apresentará a probabilidade de compatibilidade da entrada com a classe associada a ele via função de ativação de *softmax*.

2.2.1 Treinamento da CNN via *Backpropagation*

O aprendizado de uma CNN é baseado no reajuste proporcional ao erro calculado na saída dos seus parâmetros livres, *bias* e pesos. Esta etapa de reajuste é responsável pelo treino da CNN, logo, ao final da execução do reajuste, a CNN é considerada treinada para a entrada fornecida. Assim, é esperado que, após ser treinada para diferentes entradas, a CNN responda corretamente a uma entrada não treinada.

Esse reajuste é, usualmente, realizado a partir do algoritmo de *backpropagation* (RUMELHART et al., 1985) (LECUN et al., 1989). O algoritmo calcula o gradiente de uma função de custo para determinar o ajuste dos parâmetros da CNN a fim de minimizar os erros que afetam o desempenho da rede, ou seja, os valores de saída.

Formalmente, seja $\boldsymbol{\theta}$ o conjunto de todos os parâmetros livres da CNN (*bias* e pesos). O parâmetro ótimo, $\boldsymbol{\theta}_{opt}$, para uma tarefa específica pode ser obtida ao minimizar uma função de custo apropriada. Supondo um conjunto de treinamento de N pares entrada-saída $\{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{N}$, onde, $\mathbf{x}_k$ é o k-ésimo valor de entrada e $\mathbf{y}_k$ é o k-ésimo valor de saída esperada e $\mathbf{o}^k$ é a saída gerada pela CNN para a entrada $\mathbf{x}_k$, então uma função de custo a minimizar

é a média das perdas sobre o conjunto de treinamento, ou seja:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N} l(\boldsymbol{\theta}, \mathbf{y}_k, \mathbf{o}_k) \tag{2.4}$$

Treinar uma CNN é um problema de otimização global, ou seja, problemas cuja função a ser otimizada possui mínimos locais na região de interesse (TÖRN; ŽILINSKAS, 1989). Ao minimizar a função de custo, é possível encontrar o conjunto de parâmetros ótimo. Ou seja:

$$\boldsymbol{\theta}_{opt} = min_{\boldsymbol{\theta}\in\mathcal{A}}\{\mathcal{L}(\boldsymbol{\theta})\} \tag{2.5}$$

em que $\mathcal{A}$ é o espaço de parâmetros.

A solução comumente utilizada para otimizar os parâmetros de uma rede CNN ao solucionar a expressão (2.5) é o algoritmo *Stochastic Gradient Descent* (SGD) (WIJNHOVEN; WITH, 2010)(ZINKEVICH et al., 2010), como pode ser observado em (GIRSHICK, 2015)(GIRSHICK et al., 2014)(REN et al., 2015).

2.3 *Pipeline* para Detecção de Objetos

A detecção de objetos consiste na realização de duas tarefas: (*i*) classificação, em que se associa um objeto a uma classe, e (*ii*) localização, que consiste na obtenção das coordenadas dos pontos que delimitam um retângulo circunscrito ao objeto, tal retângulo é denominado na literatura como *bounding box* - BB (tradução livre, retângulo circunscrito)(FELZENSZWALB et al., 2010). Entretanto, algumas questões fundamentais devem ser consideradas para a realização da detecção de objetos: (*i*) Como procurar objetos de tamanhos significativamente diferentes na imagem? (*ii*) Como classificar uma imagem? (*iii*) O BB referente ao objeto detectado é único e adequado? As considerações apresentadas são solucionadas, respectivamente, pelos tópicos abordados nesta seção.

2.3.1 *Region Proposal Algorithm*

Uma maneira de buscar objetos numa imagem é utilizar uma *sliding window* (tradução livre, janela deslizante), como pode ser visto em (DALAL; TRIGGS, 2005). Uma janela, de tamanho fixo definido, percorrerá toda a imagem até que algum objeto conhecido esteja totalmente inserido nesta janela. Todavia, são necessárias várias iterações deste

procedimento para tamanhos diferentes de janela para que uma busca exaustiva possa ser realizada. Desta forma, essa busca exaustiva por objetos em uma imagem tem um alto custo computacional.

A busca por objetos via *sliding window* pode ser intuitiva ao primeiro momento, entretanto, seu alto custo computacional é um empecilho. Por esse motivo, a fim de evitar iterações desnecessárias, que aumentam o custo computacional do algoritmo, é desejável que a busca seja realizada apenas em algumas regiões da imagem em que exista uma alta probabilidade de existência de objeto. A definição de regiões com alta probabilidade de existência de um objeto é realizada via um *Region Proposal Algorithm* (tradução livre, algoritmo de propostas de região).

O conjunto de regiões propostas pode ser obtido a partir de vários métodos, entre eles, os métodos de *Objectness* proposto por (ALEXE; DESELAERS; FERRARI, 2012) e o método de *Selective Search* proposto por (UIJLINGS et al., 2013).

Objectness é uma medida associada a uma janela de uma imagem a fim de mensurar a probabilidade desta janela pertencer a um objeto. Entretanto, a medida por si só não define regiões prováveis de conter um objeto. Ao invés disso, dado um conjunto de regiões previamente definido, o *Objectness* é utilizado para definir os valores de probabilidade associada a cada uma das regiões e, a partir desses valores, reduzir a quantidade de regiões a serem avaliadas.

O *Selective Search* define as regiões propostas a partir dos seguintes procedimentos:

1. Sobre-segmentação da imagem, como proposto em (FELZENSZWALB; HUTTENLOCHER, 2004), ou seja, divisão da imagem em pequenos grupos.
2. A cada iteração, agrupar dois grupos similares em um grupo maior.
3. Repetir o item 2, reduzindo a exigência de similaridade para unir dois grupos, até que só reste um único grupo na imagem, ou seja, o algoritmo para assim que a imagem inteira seja o único grupo proposto.
4. Por fim, utilizar todos os grupos gerados em todas as iterações do algoritmo como candidatos a objetos.

Figura 6 – *Region Proposal* via *Selective Search*

Fonte: UIJLINGS et al.,2013.

A Figura 6 mostra os grupos segmentados (parte superior) e as regiões propostas (parte inferior) em três iterações diferentes. A união de vários grupos pequenos em grupos maiores e a utilização de todos os grupos gerados a cada iteração como possíveis candidatos a objetos permite que o *Selective Search* proponha objetos independente do seu tamanho e nível hierárquico na imagem, por exemplo, a mulher dentro da TV e a TV em si.

2.3.2 Classificação

A classificação de uma imagem, ou região de uma imagem, pode ser realizada por classificadores binários (GIRSHICK et al., 2014) ou por camadas totalmente conectadas de uma CNN (GIRSHICK, 2015).

Os classificadores binários são associados a uma única classe, ou seja, estes classificadores dizem se uma imagem de entrada pertence ou não à classe associada ao próprio classificador, como pode ser visto em (GIRSHICK et al., 2014). Desta maneira, são necessários vários classificadores binários para que a imagem de entrada possa ser classificada em um conjunto de classes. Entretanto, quando a classificação é realizada por uma camada totalmente conectada, a camada de saída apresenta as probabilidades de a imagem pertencer a cada uma das classes possíveis, definidas previamente, isto pode ser visto em (GIRSHICK, 2015). Assim, a classe de saída que possua a maior probabilidade é definida como a classe do objeto .

2.3.3 *Non-Maximum Supression*

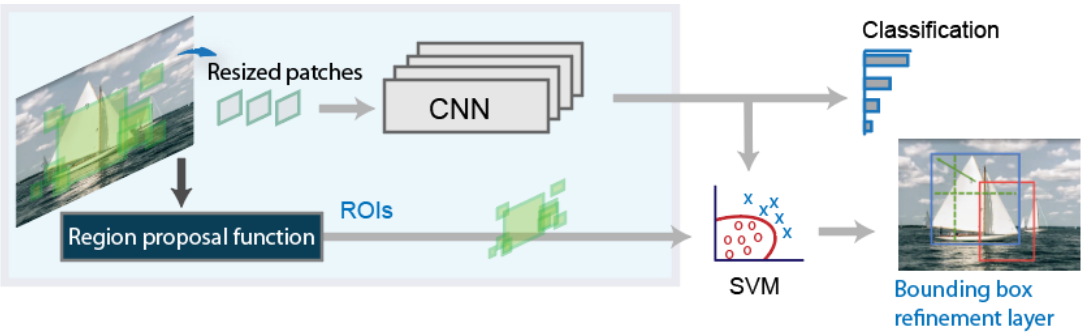
É esperado que a detecção de objetos em imagens forneça saídas repetidas para um mesmo objeto. Esta situação indica que o algoritmo detectou corretamente um objeto existente. Entretanto, não é desejável que um mesmo objeto gere mais de uma saída, ou seja, vários BBs.

Por esse motivo, é aplicado um algoritmo de *Non-Maximum Supression*, apresentado em (FELZENSZWALB et al., 2010), para que BBs redundantes sejam descartados. O *Non-Maximum Supression* leva em consideração a relação da área de intersecção e união (IOU - *Intersection over Union*) de dois BBs, caso essa relação seja superior a um valor limite previamente definido, o menor BB será descartado. Assim, o processo de supressão se repete até que haja apenas um BB ao final.

2.3.4 *Bounding Box Regression*

A existência de um único BB para um objeto não garante que este BB seja a solução desejada. O BB pode ser, por exemplo, maior ou menor que o desejado. O problema da definição das coordenadas finais do BB é solucionado a partir de um algoritmo de *Bounding Box Regression*, proposto em (FELZENSZWALB et al., 2010). Considerando uma CNN para a detecção de objetos, ao final da etapa de extração de características, as características adquiridas servirão de entrada para dois conjuntos independentes de camadas totalmente conectadas: destinados ao cômputo das probabilidades das classes que o objeto será classificado e ao refinamento do BB. A Figura 7 ilustra a divisão entre a classificação e o refinamento dos BBs na arquitetura R-CNN.

Figura 7 – Classificação e Refinamento de BBs na arquitetura R-CNN



Fonte: MATHWORKS,2019.

Esse novo conjunto de camadas totalmente conectadas fará o refinamento do BB a partir de um processo semelhante àquele destinado à classificação. A saída desse conjunto serão as coordenadas do BB refinado, que serão comparadas com as coordenadas do BB desejado definido previamente, denominado *Ground Truth*. O valor de erro obtido via comparação do BB com o *Ground Truth* acarretará no reajuste dos pesos das camadas totalmente conectadas via *backpropagation*, dessa maneira, treinando essas novas camadas. Ao final do treinamento da rede, espera-se que para uma nova entrada as coordenadas do BB refinado tenda às coordenadas do *Ground Truth*.

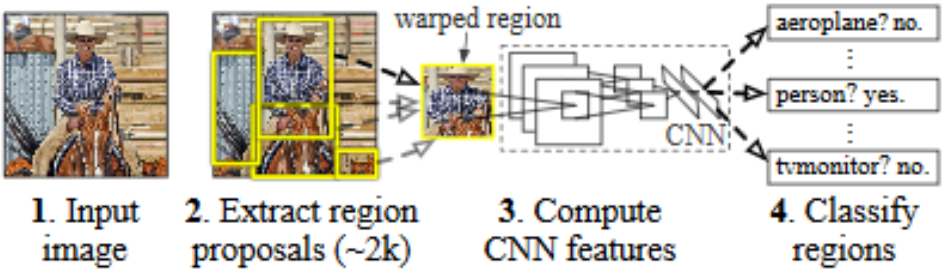
2.4 Técnicas de Detecção de Objetos baseadas em DL

Seguindo o *pipeline* de detecção de objetos, apresentado na seção anterior, são derivadas várias técnicas baseadas em DL. Estas técnicas têm como componente fundamental as CNNs. A CNN extrai características da imagem que serão utilizadas pelos classificadores binários ou pelas camadas totalmente conectadas para que a classificação da imagem, ou região de uma imagem, seja realizada. A subseção a seguir apresenta a técnica que será implementada neste trabalho.

2.4.1 RCNN

A arquitetura R-CNN para detecção de objetos, proposta em (GIRSHICK et al., 2014), é composta por três etapas principais: (*i*) definição do conjunto de regiões propostas; (*ii*) extração de características de cada região proposta via CNN e, por fim; (*iii*) classificação binária da região proposta via classificadores específicos de classe. A Figura 8 ilustra as etapas descritas.

Figura 8 – *Pipeline* de detecção de objetos via RCNN



Fonte: GIRSHICK et al.,2014.

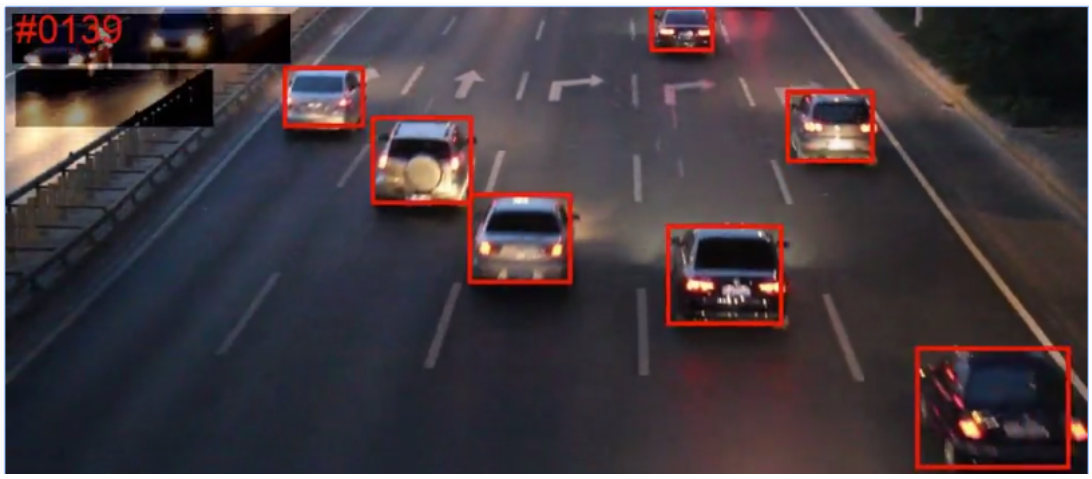
A definição do conjunto de regiões propostas é realizada a partir do algoritmo *Selective Search*, gerando aproximadamente 2 mil propostas de região (UIJLINGS et al., 2013). Em seguida, cada região proposta é redimensionada e passada como parâmetro de entrada para uma CNN, que irá gerar os vetores de características extraídas para cada região proposta. Vale ressaltar que a CNN é apenas um dos componentes da arquitetura R-CNN. Desta forma, a sigla R-CNN é referente a todas as etapas combinadas para que a detecção seja realizada, enquanto a CNN é utilizada apenas para a classificação das regiões propostas na primeira etapa da arquitetura.

Ao final, cada vetor de característica de cada região proposta é analisado e classificado pelos classificadores binários. Além disso, regiões propostas redundantes, ou seja, que possuam um IOU superior a um limiar, são descartadas via o algoritmo *Non-Maximum Suppression*.

2.5 Detecção de Veículos

O problema em estudo é a detecção de veículos em *frames* de uma sequência de vídeo, ou seja, a partir de sequências gravadas por câmeras de monitoramento de trânsito o código a ser implementado deve ser capaz de localizar (obter as coordenadas dos BBs) e classificar um veículo. A Figura 9 ilustra o tipo de detecção que o código deve ser capaz de realizar em algumas imagens.

Figura 9 – Detecção de veículos esperada do código a ser implementado



Fonte: WEN et al.,2015.

2.6 Resumo

Neste capítulo foram apresentadas as bases teóricas dos componentes de um *pipeline* clássico de detecção de objetos. Desde a proposição de regiões prováveis de se encontrar um objeto, extração de características de níveis altos de abstração da imagem, classificação de um objeto e o refinamento do BB que engloba um objeto. As ferramentas teóricas descritas neste capítulo estão diretamente relacionadas às tarefas necessárias para implementação da arquitetura proposta e descrita no capítulo a seguir para solucionar o problema de detecção de veículos.

3 PROPOSTA

3.1 Introdução

Neste capítulo serão apresentadas cada uma das etapas necessárias para implementação da arquitetura R-CNN, aplicadas ao problema de detecção de veículos. Desta forma, o capítulo é divido entre cada uma dessas etapas, que são: *(i)* pré-processamento de dados, *(ii)* treinamento da CNN e *(iii)* refinamento das regiões classificadas.

3.2 Pré-Processamento de Dados

Analisando o *pipeline* da R-CNN, percebe-se que, para solucionar a tarefa de detecção de objetos, a CNN deve ser capaz de classificar regiões da imagem, obtidas a partir de um *Region Proposal Algorithm*, neste caso, o *Selective Search*. Desta forma, o treinamento da CNN deve ser realizado a partir de regiões das imagens, que contenham veículos ou não.

Considerando que, inicialmente, apenas as imagens originais estão disponíveis, é necessária uma etapa de pré-processamento para que se tenha um banco de dados coerente para o treinamento da rede. Desta forma, o banco de dados que será utilizado para o treinamento da rede será composto de regiões que contenham os veículos divididos pelas seguintes classes: '*car*' (carros), '*bus*' (ônibus), '*van*' (vans) e '*others*' (principalmente caminhões, mas, são veículos não pertencentes às outras classes). Não obstante, foi atribuída uma classe para regiões que não contenham nenhum veículo, definida como '*background*'. A Figura 10 ilustra alguns exemplos das classes.

A geração das regiões que irão compor o conjunto de treino é realizado a partir da aplicação do *Selective Search* nas imagens redimensionadas para 224×224 do banco de dados. Caso contrário, o tempo necessário para execução do *Selective Search* em todas as imagens em seu tamanho original, aproximadamente 20 dias, tornaria o trabalho muito custoso. Regiões propostas pelo *Selective Search* que tenham IOU superior a $0,5$ com um *Ground Truth* presente na imagem serão consideradas amostras válidas para a classe do *Ground Truth* associado. Entretanto, uma amostra só poderá ser considerada negativa, denominadas *background*, caso a região apresente um IOU máximo de $0,3$ com qualquer *Ground Truth* presente na imagem. Além disso, para este problema específico, foram utilizados outros critérios para definição das amostras de *background* que serão apresentados no próximo

Figura 10 – Exemplo de veículos pertencentes a cada classe.

Fonte: Produção do próprio autor.

capítulo.

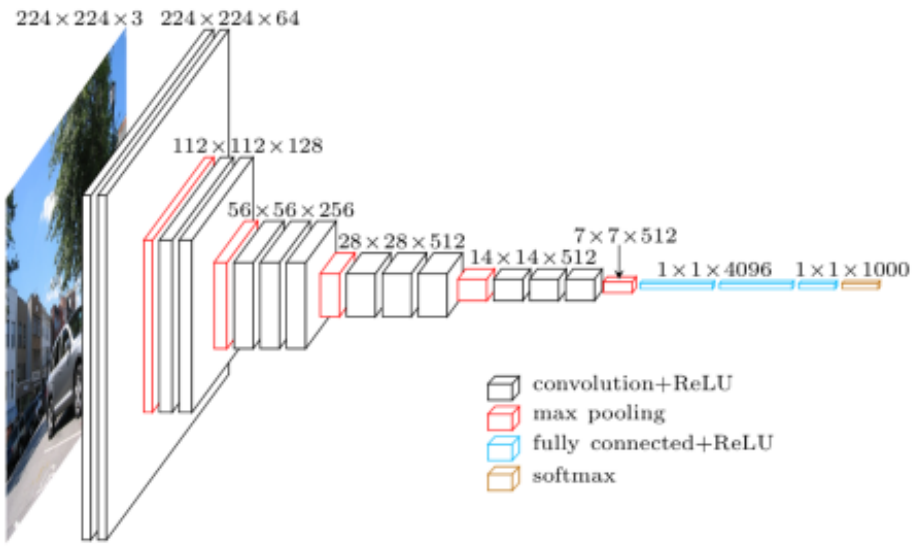
3.3 Treinamento da CNN

Após a etapa de pré-processamento de dados, o banco de dados necessário para que o treinamento da CNN seja realizado é obtido. Entretanto, o treinamento de uma CNN profunda demanda tempo. Redes conhecidas como ResNet, VGG ou Inception (HE et al., 2016)(SIMONYAN; ZISSERMAN, 2014)(SZEGEDY et al., 2015) relatam tempo de treinamento de semanas.

Para resolver este problema, foi utilizada a técnica de *Transfer Learning* (YOSINSKI et al., 2014). O objetivo desta técnica é aproveitar os pesos de redes que foram previamente treinadas para um problema similar ao problema a ser resolvido, neste caso, classificação de veículos. São utilizados os pesos e camadas iniciais de uma rede pré-treinada, sendo que as últimas camadas são descartadas e novas camadas são treinadas de modo que o novo problema seja resolvido. A premissa utilizada é que camadas iniciais da rede tendem a aprender características abstratas das imagens como forma, contornos e texturas, enquanto camadas mais profundas serão mais especializadas (YOSINSKI et al., 2014). Desta forma, é possível aproveitar camadas iniciais e retreinar as camadas finais para resolução do novo problema.

A rede utilizada para realização do *Transfer Learning* foi a VGG16 (SIMONYAN; ZISSERMAN, 2014). Foram utilizados os pesos e camadas convolucionais da VGG, pré-treinados para o problema de classificação proposto pelo ImageNet (DENG et al., 2009), e foram descartadas as últimas camadas da VGG, as *Fully Connected Layers*, para inserção de novas camadas que serão treinadas para o problema atual. A Figura 11 apresenta a arquitetura da VGG.

Figura 11 – Arquitetura VGG



Fonte: SIMONYAN; ZISSERMAN,2014

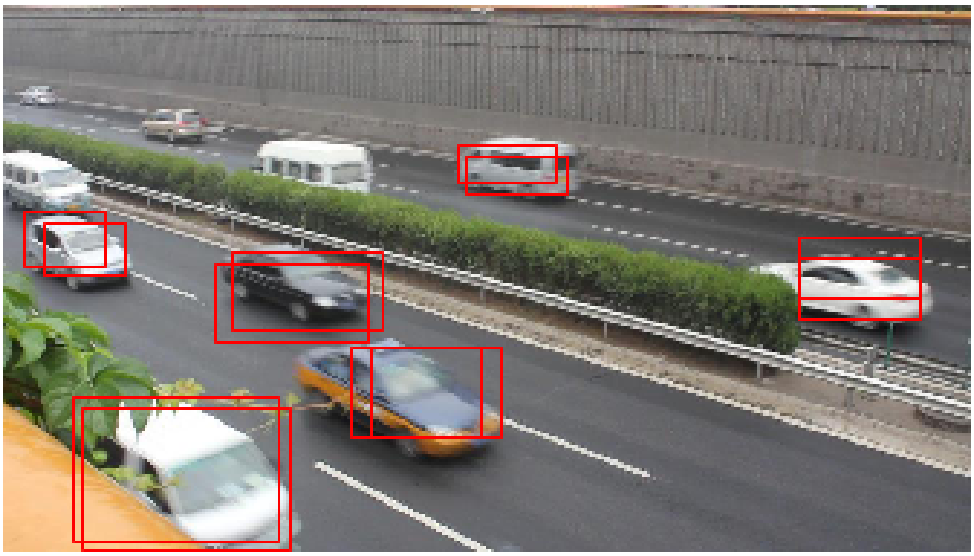
3.4 Refinamento das regiões classificadas

Após a classificação das imagens propostas pelo *Selective Search*, o resultado da detecção de objetos é obtido. Contudo, é possível que existam BBs redundantes, ou seja, vários BBs podem estar associado a um único objeto, como ilustra a Figura 12.

Partindo dessa premissa, é necessária uma etapa de refinamento das regiões classificadas. O algoritmo de *Non-Maximum Suppression* é aplicado para remover BBs que tenham um alto IOU entre si. Além da utilização do *Non-Maximum Suppression*, o problema em questão possui desafios particulares que necessitam mais etapas de refinamento da classificação que serão detalhados no próximo capítulo.

Ao final, são obtidas as regiões propostas, a classificação das regiões via CNN e o refinamento das regiões classificadas. Desta forma, o *pipeline* da arquitetura R-CNN está completo.

Figura 12 – Exemplo de mais de um BB associado a um objeto.



Fonte: Produção do próprio autor.

3.5 Resumo

Este capítulo é focado na explicação das etapas necessárias para o desenvolvimento da solução proposta para o problema em estudo. O próximo capítulo apresentará detalhes da implementação dessas etapas assim como os resultados obtidos.

4 RESULTADOS

4.1 Introdução

No presente capítulo são apresentados os resultados obtidos após a implementação da arquitetura proposta. O capítulo inicia descrevendo o banco de dados utilizado, a etapa de pré-processamento e o treinamento. Em seguida, são apresentadas as métricas para avaliar o desempenho da arquitetura assim como o resultado dos experimentos. Por fim, é realizada uma comparação entre os resultados deste trabalho com outras arquiteturas utilizadas aplicadas ao mesmo banco de dados.

4.2 Recursos Computacionais

Recursos de *Software* As etapas de treinamento e teste da arquitetura proposta foram realizadas utilizando *Keras* com *backend* baseado em *Tensorflow*, *software* de código aberto desenvolvido pela *Google brain Team*[1], destinado à programação numérica utilizando programação baseada em fluxo de dados em grafos (ABADI et al., 2016). Apesar de ser utilizado para outros propósitos, devido ao amplo suporte à utilização de GPUs, o *Tensorflow* é frequentemente utilizado em problemas de ML e DL para acelerar o treinamento de redes neurais.

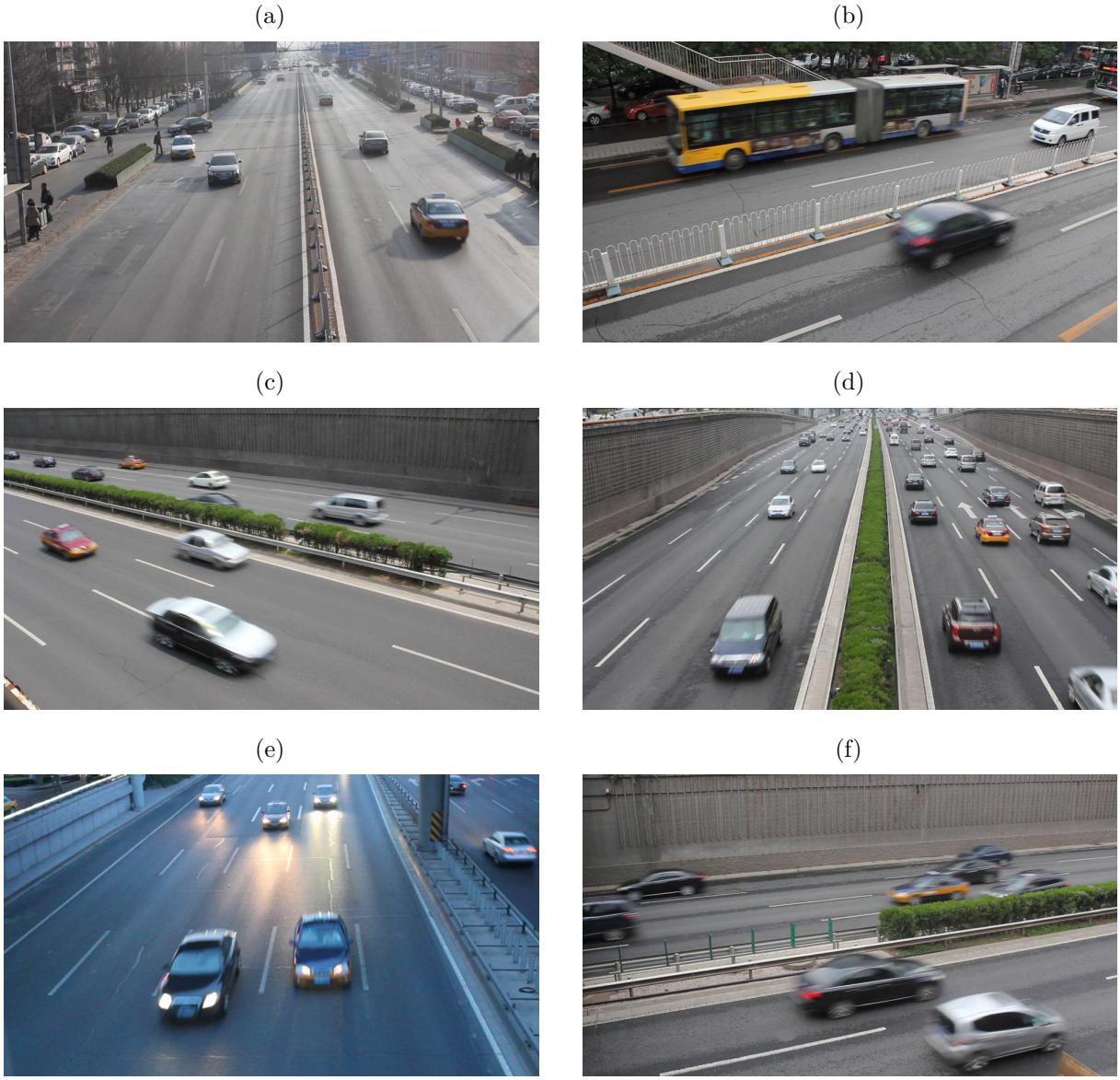
Recursos de *Hardware* O *hardware* necessário para o desenvolvimento do trabalho se restringe a, no mínimo, um computador com capacidade de processamento suficiente para o treinamento da arquitetura de CNN que foi implementada. Desta forma, o trabalho foi realizado, principalmente, no computador pessoal do autor. Visando acelerar o processo de treino e teste das redes implementadas, além do computador pessoal do autor, foram utilizados computadores disponíveis no laboratório VIROS e CISNE da UFES. Estes computadores possuem a seguinte configuração: (*i*) sistema operacional Linux, distribuição Ubuntu Server 16.04; (*ii*) processador Intel Core i7-7700, 3.60GHz com 4 núcleos físicos; (*iii*) memória RAM de 16 GB; (*iv*) unidade de armazenamento de 1TB (disco rígido); (*v*) placa de vídeo Nvidia Geforce GTX 1080, com 8 GB de memória dedicada.

Base de dados UA-DETRAC A base de dados utilizada no trabalho, para treino e validação do detector de objetos implementado baseado em CNN, foi a base de dados

[1] <https://research.google.com/teams/brain/>

DETRAC, apresentado por (WEN et al., 2015). O DETRAC é composto por 10 horas de vídeo gravados por câmeras de monitoramento de trânsito localizadas em 24 pontos diferentes das cidades de Pequim e Tanjin, China. Os vídeos foram gravados a uma taxa de 25 FPS. Assim, a base de dados possui mais de 140.000 *frames* com resolução de 960×540 pixels, totalizando $1,21$ milhões de BB de objetos rotulados e 12 Gb de dados. A Figura 13 apresenta imagens que compõem o banco de dados.

Figura 13 – Exemplo de imagens pertencente ao banco de dados



Fonte: WEN et al.,2015.

4.3 Implementação

Esta seção é dedicada a fornecer explicações mais detalhadas da implementação da arquitetura aplicada ao problema específico de detecção de veículos. As subseções são divididas de forma a ressaltar as particularidades encontradas nas etapas de implementação relacionadas a: (*i*) características intrínsecas do banco de dados utilizado, (*ii*) problemas específicos de implementação para o treinamento de uma CNN a partir de um banco de dados grande, e (*iii*) técnicas e critérios utilizados na etapa de refinamento das regiões classificadas.

4.3.1 Particularidades do Banco de Dados

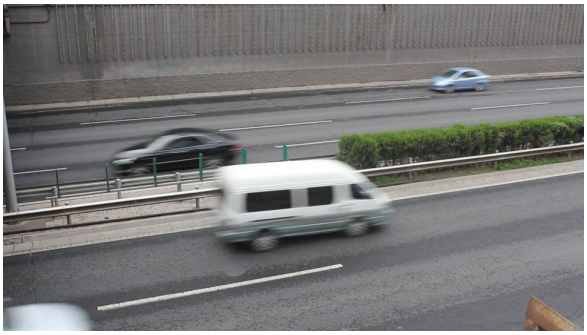
4.3.1.1 Escala

A Figura 14 apresenta dois exemplos de imagens presentes no banco de dados, ressaltando a diferença das imagens obtidas devido ao posicionamento das câmeras em relação às rodovias serem diferentes.

Figura 14 – Imagens geradas por câmeras em posições diferentes

(a) Paralela à rodovia

(b) Perpendicular à rodovia



Fonte: WEN et al.,2015.

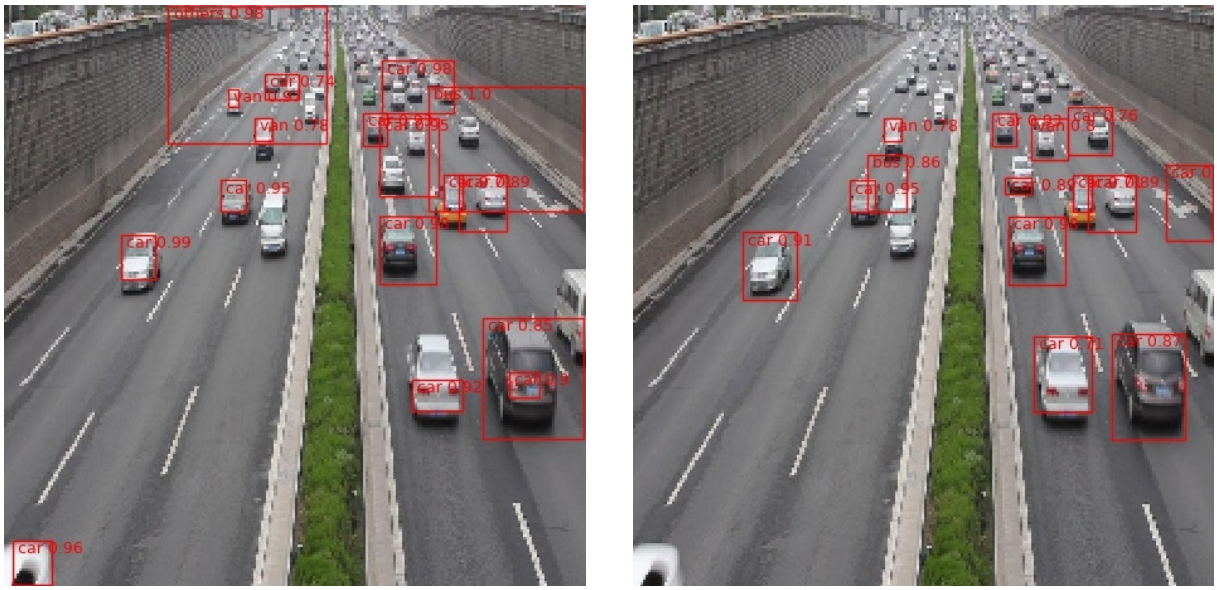
A existência de uma diferença de escala é perceptível nas imagens geradas por câmeras paralelas à rodovia. Veículos próximos à câmera são representados por BBs maiores e veículos distantes à câmera são representados por BBs menores. Dentro de uma mesma classe de veículo podem existir exemplos positivos com escalas diferentes. As imagens geradas por câmeras perpendiculares acentuam ainda o problema da diferença de escala, uma vez que produzem exemplos positivos com BB maiores que os de uma câmera paralela e alteram a relação entre largura e altura dos BB observada para imagens geradas por câmeras paralelas.

Em suma, o problema de escala influencia diretamente em dois pontos: (*i*) definição dos parâmetros a serem utilizados pelo algoritmo *Selective Search* e (*ii*) filtragem de regiões propostas.

A filtragem das regiões propostas fornecidas pelo *Selective Search* é realizada com base nas características intrínsecas das imagens disponíveis no banco de dados. Desta forma, regiões com relação de altura e largura que superem o padrão observado entre os veículos são descartadas, uma vez que o objetivo é encontrar apenas os veículos presentes na imagem. Além disso, considerando o padrão observado nas imagens geradas por câmeras paralelas à rodovia, o tamanho máximo que uma região proposta válida pode ter é proporcional à sua posição na imagem, ou seja, regiões localizadas na parte superior da imagem possuem um tamanho máximo aceitável inferior à regiões localizadas na parte inferior da imagem. A Figura 15 ilustra a influência dessa filtragem.

Figura 15 – Exemplo da influência do filtro de tamanho em relação à localização do BB em uma imagem

(a) Sem o filtro (b) Com o filtro



Fonte: Produção do próprio autor.

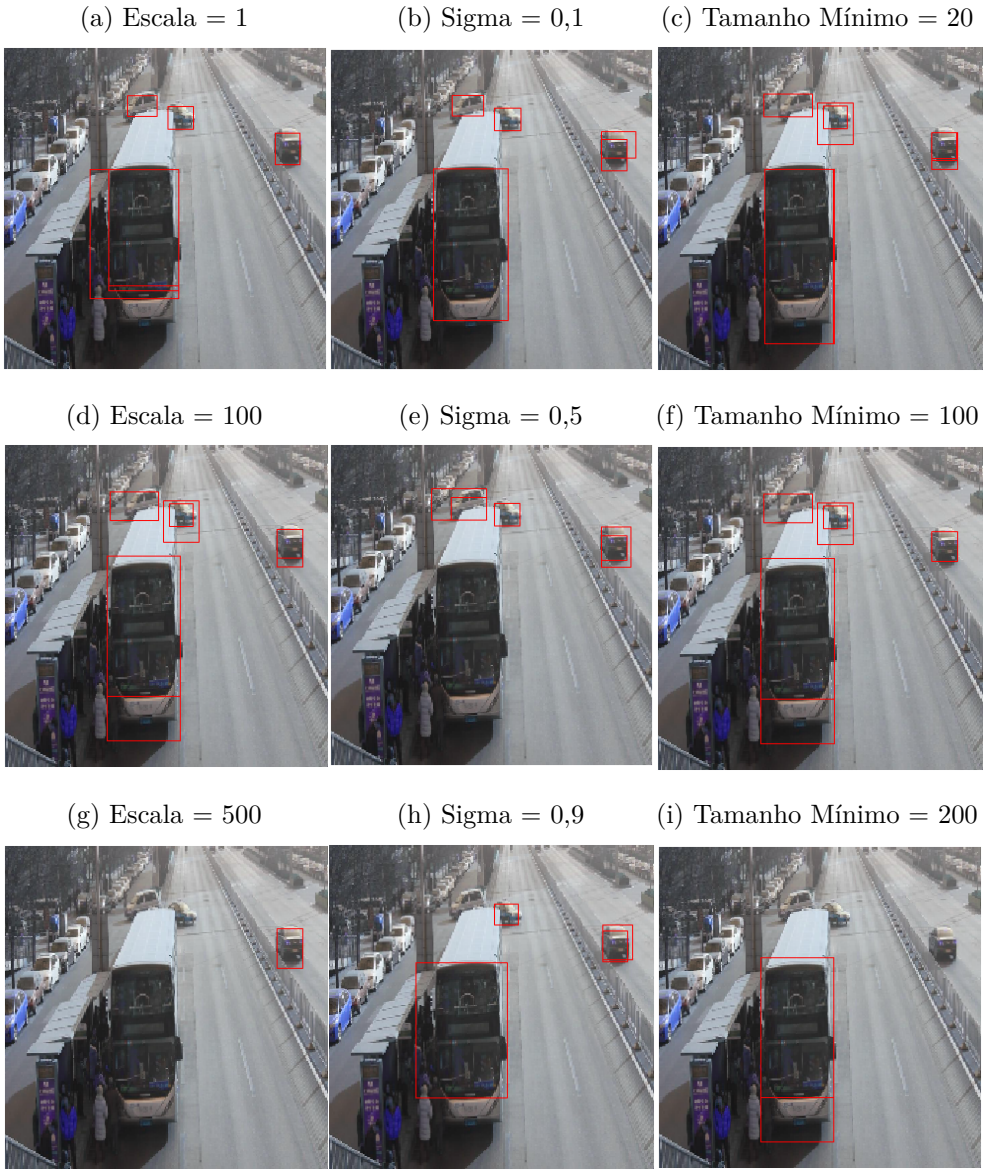
A utilização do *Selective Search* requer que três parâmetros sejam previamente definidos: escala, sigma e tamanho mínimo das regiões propostas. Combinações diferentes desses parâmetros resultarão em conjuntos de regiões propostas diferentes.

Todos três parâmetros são descritos em (FELZENSZWALB; HUTTENLOCHER, 2004). O tamanho mínimo define o menor valor de área, em pixels, de uma região para que ela seja uma região proposta válida. O parâmetro sigma corresponde à largura do filtro Gaussiano utilizado pela técnica de segmentação. Por fim, o parâmetro de escala define o tamanho

máximo possível dos grupos de pixels para as regiões propostas, ou seja, quanto maior este parâmetro, maior serão estes grupos.

A definição desses parâmetros é arbitrária, desta forma, variou-se cada um deles isoladamente em várias imagens para que os valores fossem definidos de forma que veículos de diferentes escalas pudessem ser abrangidos pela maior quantidade de regiões propostas. A Figura 16 ilustra a variação feita em cada um parâmetros.

Figura 16 – Exemplo de variação isolada dos parâmetros do *Selective Search*



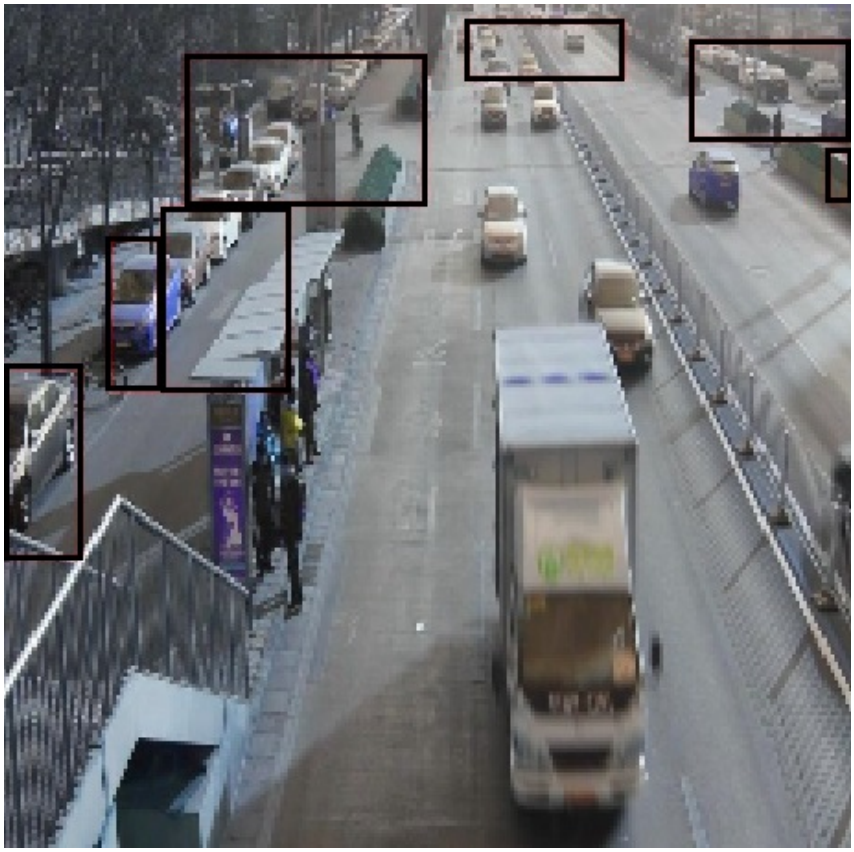
Fonte: Produção do próprio autor.

Analisando a variação dos parâmetros isoladamente para o conjunto de imagens escolhido, foram definidos os seguintes valores finais: Escala = 90, Sigma = 0,9 e Tamanho Mínimo = 30.

4.3.1.2 Regiões Ignoradas

O banco de dados define algumas zonas em cada imagem que não são consideradas, contendo carros estacionados ou carros em dimensões pequenas na imagem. Desta forma, utilizar regiões que incluam partes das zonas ignoradas como amostras negativas de veículos poderia inviabilizar o treinamento da CNN, uma vez que essas zonas podem conter veículos. A Figura 17 apresenta um exemplo de zonas ignoradas pelo DETRAC.

Figura 17 – Exemplo de zonas ignoradas pelo DETRAC.

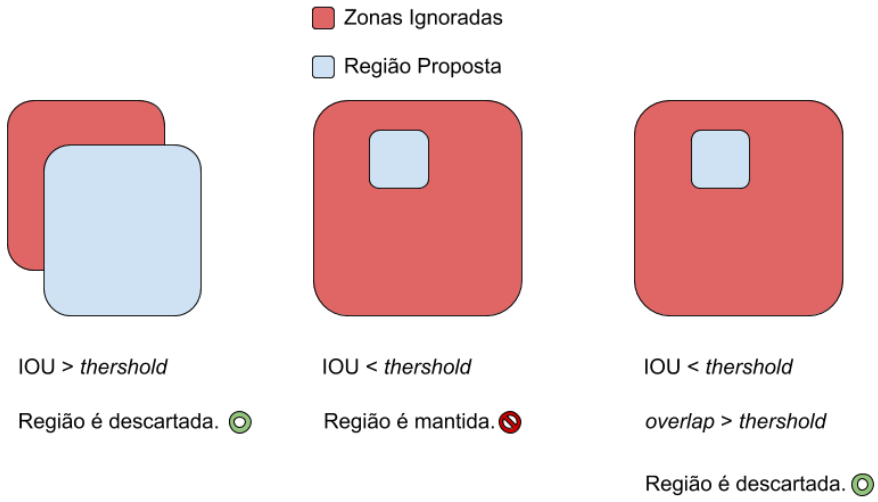


Fonte: Produção do próprio autor.

As zonas ignoradas impactam diretamente a etapa de pré-processamento, uma vez que regiões propostas pelo *Selective Search* que incluam tais zonas não podem ser utilizadas como amostras de *background*, já que podem existir veículos pertencentes às regiões ignoradas (observe a Figura 17). Por este motivo, além do critério do IOU, adicionou-se um critério de *overlap*, ou seja, regiões propostas não podem ter mais de 80% da sua área contida em uma zona ignorada nem uma zona ignorada pode ter mais de 80% de sua área contida em uma região proposta. O critério de *overlap* é necessário para solucionar casos particulares onde uma das regiões, proposta ou ignorada, é muito maior que a outra. Permitindo que uma das duas esteja inteiramente inserida na outra, fazendo com que apenas a medida do IOU não seja suficiente para evitar regiões propostas contidas em

zonas ignoradas. As Figura 18 e 19 ilustram o caso particular em que apenas a utilização do IOU não seria suficiente, a primeira apresenta situação de forma genérica enquanto a segunda apresenta o problema em uma imagem do banco de dados utilizado.

Figura 18 – Descarte de regiões propostas: IOU x IOU e *Overlap*



Fonte: Produção do próprio autor.

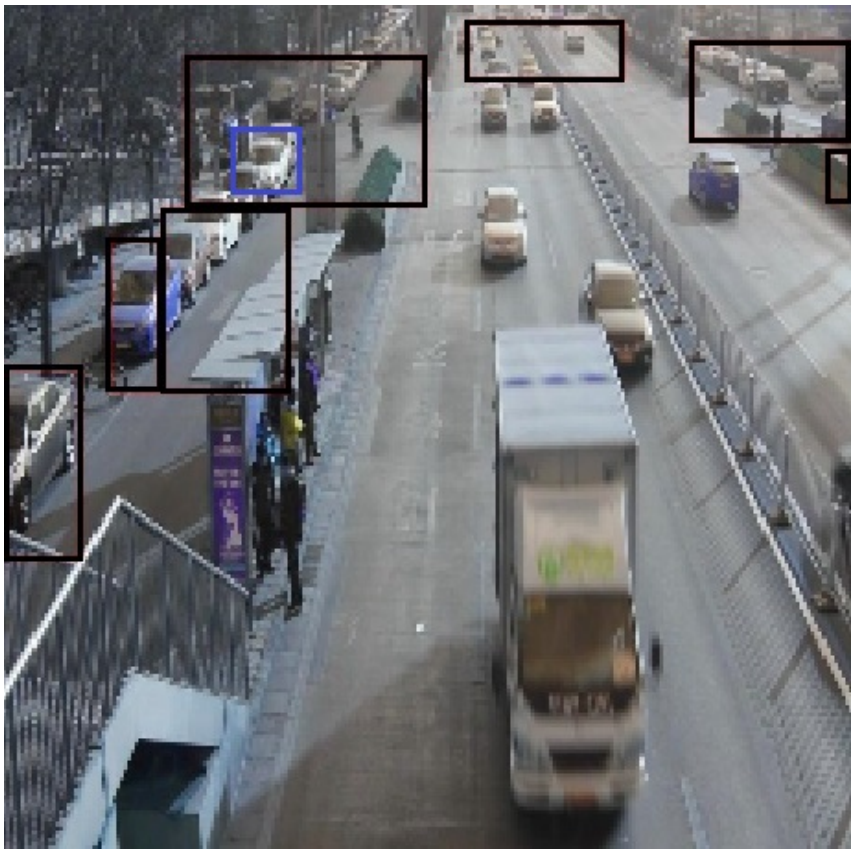
4.3.2 Particularadidades do Treinamento da CNN

O DETRAC, como apresentado na seção 4.2, possui mais de 140.000 imagens. Desta forma, pontos importantes devem ser observados ao utilizar um banco de dados dessa dimensão. As subseções a seguir detalham particularidades encontradas na utilização de um banco de dados grande para o treinamento da CNN.

4.3.2.1 Hiperparâmetros

O tamanho do banco de dados influencia diretamente na escolha dos hiperparâmetros. O tempo estimado de treinamento utilizando *batchs* de 32 imagens superava 24h por época. Por este motivo, e a fim de reduzir o tempo de treinamento, foram utilizados valores de hiperparâmetros baseados no artigo original da arquitetura RCNN (GIRSHICK et al., 2014). Desta forma, foram utilizados *batchs* de 256 imagens, que fornecem estimativas de tempo de treinamento para uma época de 3h. Além disso, o treinamento foi realizado por 20 épocas, atingindo o patamar em que a função de custo para o conjunto de validação pára de reduzir.

Figura 19 – Zonas Ignoradas (em preto) e exemplo de Região Proposta (em azul) não descartada, incorretamente, pelo critério de IOU



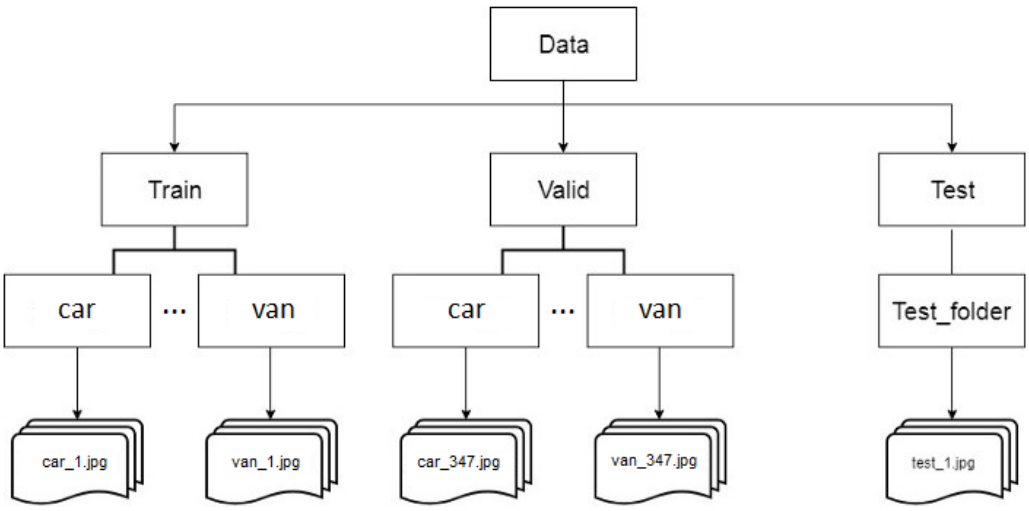
Fonte: Produção do próprio autor.

4.3.2.2 Generators

A grande quantidade de imagens presentes no banco de dados impossibilita o carregamento de todas elas na memória RAM para realizar o treinamento da CNN. Este problema é resolvido pela utilização de *generators* disponibilizado pelo módulo *keras* (CHOLLET et al., 2015).

O objetivo dos *generators* é permitir que a CNN seja treinada diretamente pelos arquivos das imagens do banco de dados, ou seja, treinada diretamente por arquivos salvos no disco rígido, resolvendo o problema de estouro da memória RAM para grandes bancos de dados (CHOLLET et al., 2015). A partir de uma estrutura de pastas definida, em que cada pasta deve ser nomeada com o nome da classe de imagens que ela contém, o *keras* se encarrega da leitura e utilização dos dados das imagens *on the fly*, ou seja, simultaneamente com o processo de treinamento da CNN. A Figura 20 apresenta a estrutura de diretórios necessária para correta utilização dos *generators*.

Figura 20 – Estrutura de diretórios para treinamento, validação e teste da CNN via *generators*



Fonte: Produção do próprio autor.

Considerando que o próprio *keras* se encarrega da leitura dos arquivos de imagem, ao realizar predições com o modelo treinado, é necessária a utilização de dados codificados da mesma forma que os utilizados no treinamento, caso contrário, as predições não serão coerentes. Por exemplo, o *keras* utiliza o módulo PIL, que codifica as imagens lidas no formato RGB por padrão, enquanto outro módulo bastante comum, o OpenCV, utiliza uma codificação BGR por padrão, ou seja, inserir uma imagem lida pelo OpenCV sem atentar para a diferença de codificação forneceria predições equivocadas para a imagem.

4.3.3 Particularadidades do Pós-Processamento

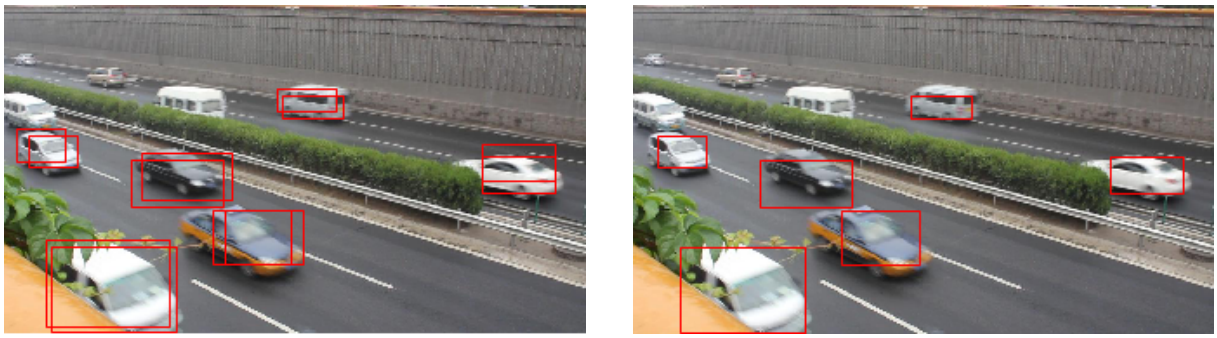
Ao final da etapa de classificação das regiões propostas pelo *Selective Search*, é obtido um conjunto muito amplo de regiões detectadas como um veículo. Entretanto, nem todas essas regiões foram corretamente classificadas e, mesmo quando o foram, várias regiões podem apontar para o mesmo veículo. Desta forma, o *Non-Maximum Suppression* é utilizado para exclusão de BBs redundantes, contudo, é necessário um critério que defina quais BBs devem ser excluídos e quais BBs devem ser mantidos. Além disso, é necessário que a quantidade de BBs classificados como veículos incorretamente seja minimizada. As subseções a seguir explicam os processos adotados para solucionar estes problemas.

4.3.3.1 Critérios de exclusão de BBs redundantes

O critério utilizado é baseado na probabilidade atribuída a cada região pela CNN. Para cada região, a CNN distribui entre as classes de veículo a probabilidade que essa região tem de pertencer a cada uma das classes, totalizando 100%. Desta forma, ao executar o *Non-Maximum Suppression*, o algoritmo prioriza regiões que tenham probabilidades mais altas de pertencer a uma classe e descarta regiões com probabilidades inferiores, ou seja, apenas a região com maior probabilidade dentre as regiões que se sobrepõem restará ao final. A Figura 21 ilustra as regiões propostas antes da execução do *Non-Maximum Suppression* e após a execução.

Figura 21 – Remoção de BB via *Non-Maximum Suppression*

(a) Antes do NMS (b) Depois do NMS



Fonte: Produção do próprio autor.

4.3.3.2 Limiares de probabilidade e tamanho das regiões

O problema de regiões classificadas como veículos de maneira incorreta (falsos positivos) é minimizado pela utilização de limiares que validarão ou não a classificação positiva de determinadas regiões.

O primeiro limiar utilizado é o de probabilidade. Regiões classificadas positivamente como um veículo, ou seja, não sendo um *background*, só serão apresentadas como uma detecção verdadeira caso a probabilidade associada a classe do veículo atribuida à região seja superior ao valor pré-estabelecido de limiar. Desta forma, regiões que a CNN atribuiu uma baixa probabilidade são descartadas.

Além do limiar de probabilidade, foi utilizado um limiar de tamanho máximo e mínimo para as regiões propostas dependendo da sua posição na imagem. Devido ao posicionamento das câmeras ao gerar as imagens, percebe-se que veículos da parte superior da imagem tendem

a ser menores, enquanto veículos da parte inferior da imagem tendem a ser maiores. A Figura 22 exemplifica essa tendência.

Figura 22 – Diferença de tamanho entre veículos da parte superior e da parte inferior da imagem

Fonte: Produção do próprio autor.

Desta forma, regiões propostas localizadas na parte superior da imagem que possuam uma área superior a área máxima permitida são descartadas. Da mesma maneira, regiões propostas localizadas na parte inferior da imagem que possuam área inferior a área mínima permitida também são descartadas.

4.4 Avaliação de Desempenho

A avaliação de desempenho do detector de veículos será realizada em duas etapas: (*i*) análise do desempenho da CNN na classificação de regiões e (*ii*) análise da desempenho do detector de veículos considerando os BB esperados e obtidos ao final do processo de detecção. As métricas utilizadas para a avaliação da detecção serão apresentadas simultaneamente aos resultados obtidos.

4.4.1 CNN

O treinamento da CNN foi realizado a partir do conjunto de imagens gerados após a etapa de pré-processamento de dados. O treinamento durou 42h e foi realizado via container do *Docker* (NICKOLOFF, 2016). A Tabela 1 apresenta a divisão dos conjuntos de treino e

validação, evidenciando a quantidade de amostras presentes de cada classe em cada um dos conjuntos.

Tabela 1 – Divisão das amostras das classes no conjunto de treino e validação

-	background	bus	car	others	van
Treino	741.166	69.273	438.643	4.960	59.461
Validação	78.578	789	59.732	1.442	10.655
TOTAL	819.744	70.062	498.375	6.402	70.116

Fonte – Produção do próprio autor

A Figura 23, mostra a evolução dos valores de acurácia e da função de perda, tanto para o conjunto de treino quanto para o conjunto de validação, durante o treinamento da CNN.

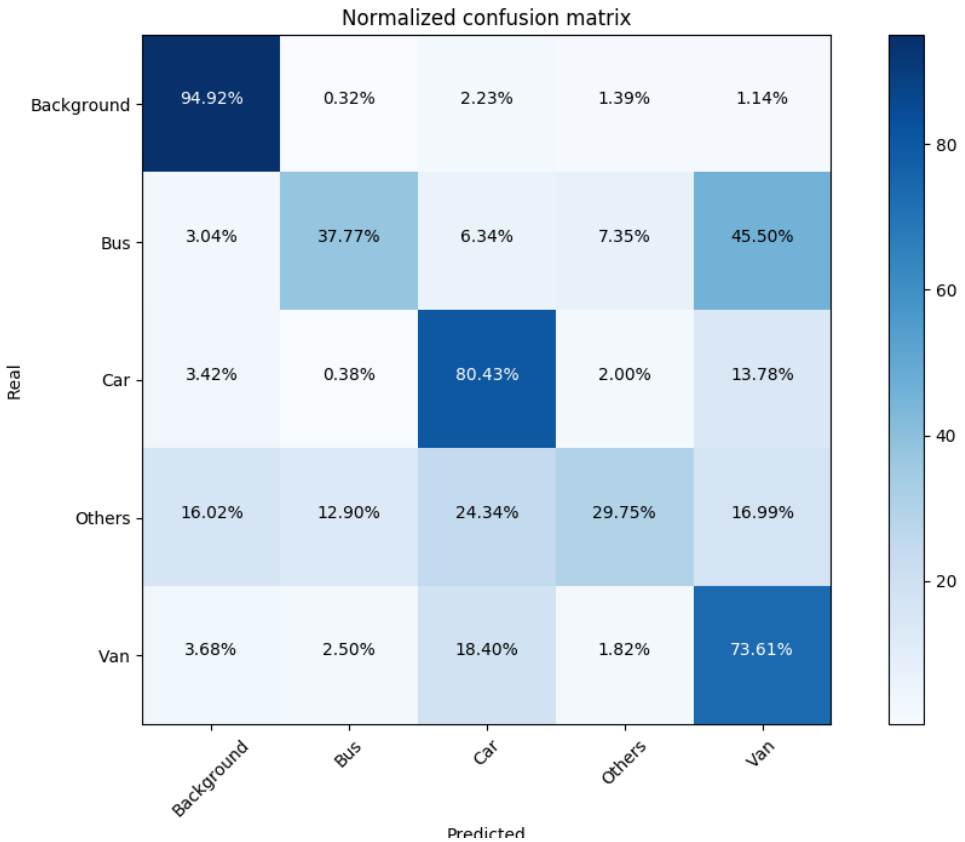
Figura 23 – Gráficos da precisão e função de custo durante treinamento da CNN

Fonte: Produção do próprio autor.

Analisando a evolução dos indicadores apresentados durante o treinamento da CNN, observa-se que a partir da terceira época a função de perda para o conjunto de validação para de diminuir e passa a crescer. Além disso, a acurácia para o conjunto de validação passa a oscilar enquanto a acurácia para o conjunto de treino continua a crescer, porém, a uma taxa inferior à taxa observada anteriormente. Desta forma, é possível perceber o início do processo de *overfitting*, em que a rede tende a ficar especializada no conjunto de treino e perde a generalidade necessária para um bom desempenho no conjunto de validação.

A fim de evitar a utilização de um modelo influenciado pelo processo de *overfitting*, são utilizados os pesos do modelo treinado até a terceira época, com acurácia de 86% para o conjunto de validação. As predições nas imagens presentes no conjunto de validação são realizadas e a matriz de confusão apresentada na Figura 24 é obtida.

Figura 24 – Matriz de Confusão gerada a partir do conjunto de validação



Fonte: Produção do próprio autor.

A matriz de confusão permite que se faça uma análise mais profunda do desempenho da CNN do que o valor final da acurácia. A CNN é mais eficiente em identificar entre '*background*' ou veículo, uma vez que 94, 92% das regiões de '*background*' foram corretamente

classificadas. Entretanto, entre as classes existentes de veículos, a CNN teve menos eficiência.

Observando as classes '*car*' e '*van*', é perceptível que elas são classificadas corretamente em mais de 70% dos casos e que em praticamente todo o restante de casos de classificação incorreta, a CNN tende a confundir essas duas classes entre si. Esse fenômeno pode ser influenciado pela semelhança desses dois tipos de veículos ao avaliar regiões das partes dianteira e traseira de ambos.

Figura 25 – Semelhança entre carros e vans

Fonte: Produção do próprio autor.

Observando a classe '*others*', a dificuldade que a CNN tem em classificar corretamente imagens pertencentes a esta classe é notória. Essa situação é agravada por dois fatores: (*i*) pouca quantidade de amostras dessa classe, como pode ser observado na Tabela 1, e (*ii*) falta de um padrão bem definido para esta classe, uma vez que são classificados como '*others*' tanto caminhões como motocicletas e bicicletas.

Por fim, a classe '*bus*' é analisada. É observado um baixo nível de classificações corretas e que a CNN confunde '*bus*' com '*van*'. Os fatores que podem influenciar esse comportamento são: (*i*) IOU $0,5$ pode ser insuficiente para coleta de boas amostras de ônibus, uma vez que *Ground Truth* para ônibus tendem a ser grandes, permitindo que partes significativas de '*background*' e outros veículos estejam presente nas amostras e (*ii*) tetos de ônibus e vans são muito semelhantes e ambos são considerados *Ground Truth* pelo UA-DETRAC. As Figuras 26 e 27 ilustram esses dois fatores, respectivamente.

4.4.2 *Object Detector*

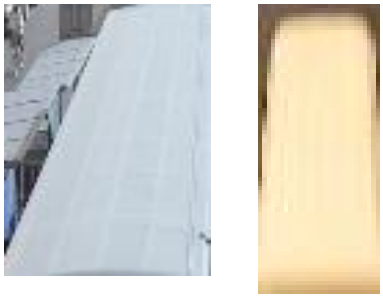
O desempenho do *Object Detector*, ou seja, a avaliação das detecções previstas pela arquitetura implementada deve ser avaliada com base nas saídas esperadas e as saídas

Figura 26 – Amostras de ônibus precárias para um IOU de 0,5

Fonte: Produção do próprio autor.

Figura 27 – Semelhança entre tetos de ônibus e vans

(a) Ônibus (b) Van



Fonte: Produção do próprio autor.

obtidas para as imagens do banco de dados. A Figura 28 apresenta algumas das imagens geradas após a execução da detecção de veículos. BBs classificados como '*car*', '*van*', '*bus*' e '*others*' são representados pelas cores vermelho, azul, amarelo e verde, respectivamente.

Analisando as imagens, é possível observar que o detector é capaz de localizar e classificar veículos presentes na imagem. Todavia, também é possível observar a existência de erros nas detecções. A seguir, é feita uma análise dos erros apresentados pelo detector.

Observando as imagens (c), (e), (g) e (l), é perceptível o erro de classificação para as classes '*bus*' e '*others*' (observe os valores da matriz de confusão para essas classes na Figura 24), que acarreta não só na classificação incorreta da região proposta como também na não

detecção de outros veículos. Vale ressaltar que erros de classificação também ocorrem para as outras classes, como pode ser observado nas imagens (b), (i) e (k).

Além disso, as imagens (a), (d), (e) e (f) mostram uma divisão de um único veículo em vários BBs, fazendo com que múltiplas detecções sejam geradas para um mesmo veículo, ainda que o *Non-Maximum Suppression* tenha sido aplicado anteriormente. Essa situação está possivelmente atrelada ao tamanho dos veículos, uma vez que os parâmetros escolhidos para o *Selective Search* tendem a propor regiões de tamanhos intermediários.

Outro tipo de erro observado são os erros de localização. Existem BBs que representam uma detecção correta, ou seja, indicam um veículo corretamente. Contudo, esses BBs podem englobar apenas parcialmente o veículo associado (observe a imagem (j)).

Por fim, são observados veículos existentes nas imagens que não são detectados. Esta situação está possivelmente relacionada a dois fatores: *(i)* nenhuma região proposta via *Selective Search* engloba esses veículos e *(ii)* as regiões propostas para esses veículos são descartadas pelo filtro de tamanho, proporcional à localização do BB na imagem. Por exemplo, as imagens (d) e (h), geradas por câmeras não paralelas à rodovia, apresentam BBs maiores na parte superior da imagem que o esperado, uma vez que os veículos estão mais próximos à câmera. Desta forma, mesmo que existam regiões propostas para esses veículos, elas serão descartadas.

Além da análise das detecções, a avaliação do desempenho do detector pode ser realizada a partir dos valores de *Precision* e *Recall*. Os valores são obtidos a partir da variação do limiar de sobreposição entre os BBs detectados e os *Ground Truths* presentes na imagem, gerando valores diferentes para cada limiar utilizado. Os valores de *Precision* e *Recall* são calculados a partir das equações 4.1 e 4.2, respectivamente.

$$P = \frac{VP}{VP + FP} \tag{4.1}$$

$$R = \frac{VP}{VP + FN} \tag{4.2}$$

Nas equações acimas são utilizados os seguintes parâmetros: *(i)* VP são os verdadeiros positivos (número de objetos corretamente detectados), *(ii)* FP são os falsos positivos (número de objetos detectados incorretamente) e *(iii)* FN são falsos negativos (número de objetos não detectados). A Tabela 2 apresenta os resultados obtidos.

Tabela 2 – Valores de *Precision* e *Recall* do Detector para limiares de sobreposição diferentes

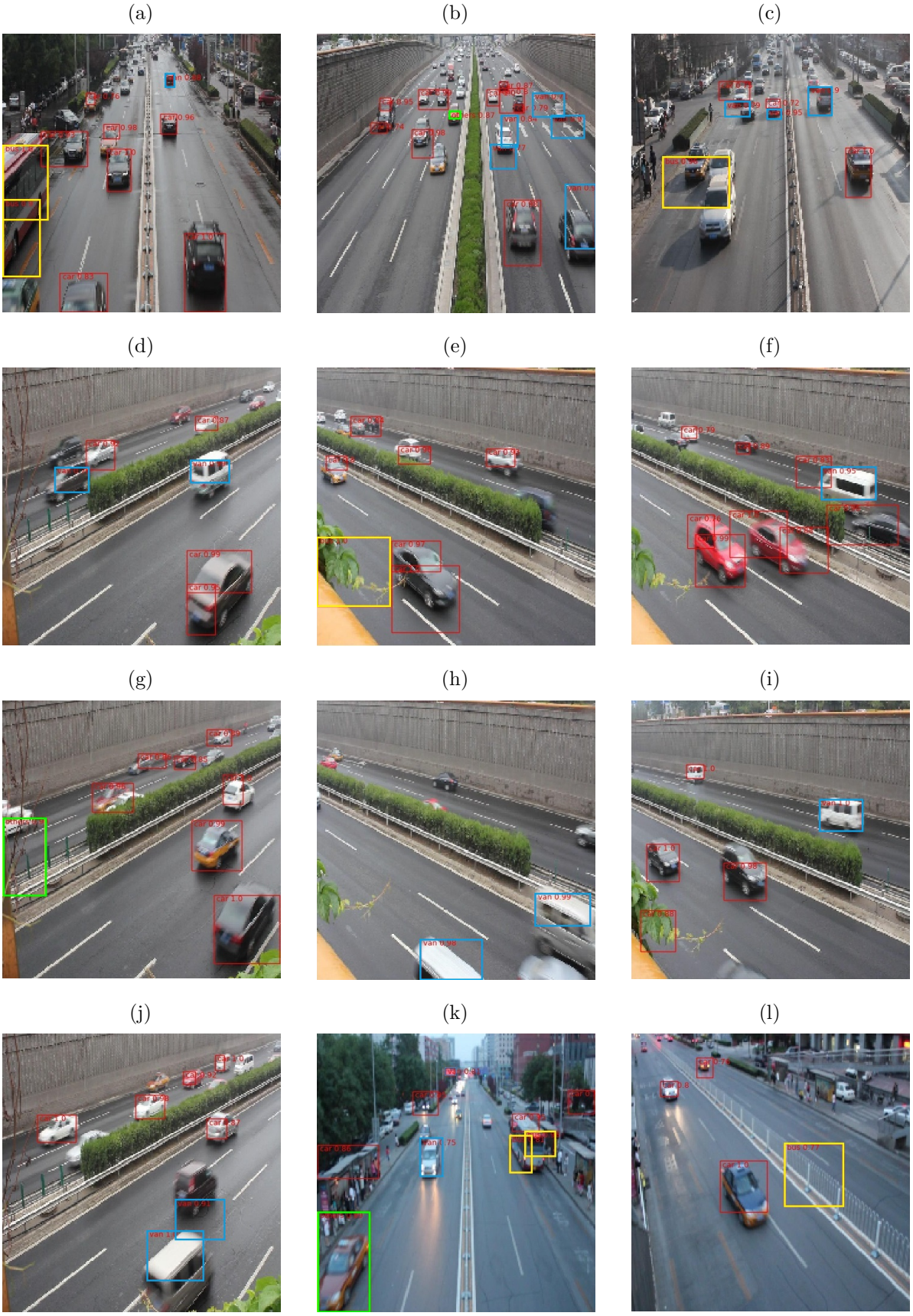
-	**IoU = 0,1**	**IoU = 0,5**	**IoU = 0,7**
Precision	$0,828$	$0,341$	$0,08$
Recall	$0,671$	$0,283$	$0,065$

Fonte – Produção do próprio autor

4.5 Resumo

Neste capítulo foram apresentados os detalhes de implementação do detector de objetos assim como os resultados obtidos. O capítulo seguinte aborda as conclusões e possíveis temas a serem pesquisados futuramente a partir deste trabalho.

Figura 28 – Exemplos de detecções



Fonte: Produção do próprio autor.

5 CONCLUSÕES E PROJETOS FUTUROS

5.1 Conclusões

O objetivo principal deste trabalho foi implementar a arquitetura R-CNN para o problema de detecção de objetos. Para isso, seguiu-se o *pipeline* para detecção de objetos descrito em (GIRSHICK et al., 2014): (*i*) utilizou-se o algoritmo *Selective Search* para gerar regiões com alta probabilidade de conter objetos; (*ii*) cada região proposta foi classificada entre uma das classes de interesse por uma CNN treinada para essa tarefa (aplicou-se a técnica de *Transfer Learning* usando como modelo base a rede VGG16); (*iii*) regiões redundantes foram exlcuídas via o algoritmo *Non-Maximum Suppression* e obtendo-se o resultado da detecção de objetos para a imagem de entrada.

A arquitetura implementada foi avaliada usando o banco de dados DETRAC (WEN et al., 2015), que consiste em imagens de câmeras de monitoramento de rodovias, para a detecção de veículos (carros, ônibus, vans e outros). O trabalho permitiu: *(i)* comprovar a viabilidade da arquitetura R-CNN para o problema de detecção de veículos e *(ii)* avaliar as particularidades da detecção de veículos.

5.2 Temas a serem pesquisados

Como trabalhos futuros:

- Realizar o treinamento da rede utilizando a resolução original das imagens (540×960);
- Utilizar outras redes pré-treinadas como *feature extractors* no lugar da VGG16, por exemplo, a ResNet.
- Aplicar outras arquiteturas de detecção de objetos ao problema de detecção de veículos, por exemplo, a Fast-RCNN e Faster-RCNN e comparar os resultados obtidos.

REFERÊNCIAS

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M. et al. Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). [S.l.: s.n.], 2016. p. 265–283. Citado na página 30.

ALEXE, B.; DESELAERS, T.; FERRARI, V. Measuring the objectness of image windows. IEEE transactions on pattern analysis and machine intelligence, IEEE, v. 34, n. 11, p. 2189–2202, 2012. Citado na página 20.

AMMOUR, N.; ALHICHRI, H.; BAZI, Y.; BENJDIRA, B.; ALAJLAN, N.; ZUAIR, M. Deep learning approach for car detection in uav imagery. Remote Sensing, Multidisciplinary Digital Publishing Institute, v. 9, n. 4, p. 312, 2017. Citado na página 13.

BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In: 27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL. [S.l.: s.n.], 2010. Citado na página 17.

CHOLLET, F. et al. Keras. 2015. <https://keras.io>. Citado na página 37.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. [S.l.], 2005. v. 1, p. 886–893. Citado na página 19.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. [S.l.: s.n.], 2009. Citado na página 28.

FELZENSZWALB, P. F.; GIRSHICK, R. B.; MCALLESTER, D.; RAMANAN, D. Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence, IEEE, v. 32, n. 9, p. 1627–1645, 2010. Citado 2 vezes nas páginas 19 e 22.

FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient graph-based image segmentation. International journal of computer vision, Springer, v. 59, n. 2, p. 167–181, 2004. Citado 2 vezes nas páginas 20 e 33.

GIRSHICK, R. Fast r-cnn. In: IEEE. Computer Vision (ICCV), 2015 IEEE International Conference on. [S.l.], 2015. p. 1440–1448. Citado 2 vezes nas páginas 19 e 21.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2014. p. 580–587. Citado 7 vezes nas páginas 11, 12, 19, 21, 23, 36 e 48.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2016. p. 770–778. Citado na página 27.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. The Journal of Physiology, v. 195, n. 1, p. 215–243, mar 1968. ISSN 0022-3751. Citado na página 15.

HUVAL, B.; WANG, T.; TANDON, S.; KISKE, J.; SONG, W.; PAZHAYAMPALLIL, J.; ANDRILUKA, M.; RAJPURKAR, P.; MIGIMATSU, T.; CHENG-YUE, R. et al. An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716, 2015. Citado 2 vezes nas páginas 12 e 13.

KARPATHY, A. et al. Cs231n convolutional neural networks for visual recognition. Neural networks, v. 1, 2016. Citado 3 vezes nas páginas 16, 17 e 18.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances In Neural Information Processing Systems, p. 1–9, 2012. Citado na página 17.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. Nature, v. 521, n. 7553, p. 436–444, 2015. ISSN 14764687. Citado na página 10.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Comput., MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, 1989. ISSN 0899-7667. Citado na página 18.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, v. 86, n. 11, p. 2278–2323, 1998. ISSN 00189219. Citado 2 vezes nas páginas 10 e 15.

LECUN, Y.; BOTTOU, L.; ORR, G. B.; MÜLLER, K. R. Efficient backprop. In: _____. Neural Networks: Tricks of the Trade. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 9–50. Citado na página 17.

LI, B.; ZHANG, T.; XIA, T. Vehicle detection from 3d lidar using fully convolutional network. arXiv preprint arXiv:1608.07916, 2016. Citado na página 13.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. European conference on computer vision. [S.l.], 2014. p. 740–755. Citado na página 10.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125–160, 2009. Citado na página 10.

MATHWORKS. MATLAB Computer Vision Toolbox. 2019. Citado na página 22.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. USA: Omnipress, 2010. (ICML'10), p. 807–814. Citado na página 17.

NICKOLOFF, J. Docker in action. [S.l.]: Manning Publications Co., 2016. Citado na página 40.

NIELSEN, M. A. Neural networks and deep learning. [S.l.]: Determination Press, 2015. Citado na página 17.

NIXON, M.; AGUADO, A. S. Feature extraction and image processing for computer vision. [S.l.]: Academic Press, 2012. Citado na página 10.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. [S.l.: s.n.], 2015. p. 91–99. Citado na página 19.

RUMELHART, D.; HINTON, G.; WILLIAMS, R.; CALIFORNIA, S. D. I. f. C. S. University of. Learning Internal Representations by Error Propagation. [S.l.]: Institute for Cognitive Science, University of California, San Diego, 1985. (ICS report). Citado na página 18.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M. S.; BERG, A. C.; LI, F.-F. ImageNet Large Scale Visual Recognition Challenge. CoRR, abs/1409.0575, 2014. Citado na página 11.

SCHWARZ, B. Lidar: Mapping the world in 3d. Nature Photonics, Nature Publishing Group, v. 4, n. 7, p. 429, 2010. Citado na página 13.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. Citado 2 vezes nas páginas 27 e 28.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. p. 1–9. Citado na página 27.

TÖRN, A.; ŽILINSKAS, A. Global optimization. [S.l.]: Springer, 1989. v. 350. Citado na página 19.

UIJLINGS, J. R.; SANDE, K. E. V. D.; GEVERS, T.; SMEULDERS, A. W. Selective search for object recognition. International journal of computer vision, Springer, v. 104, n. 2, p. 154–171, 2013. Citado 3 vezes nas páginas 20, 21 e 24.

WAN, L.; ZEILER, M.; ZHANG, S.; CUN, Y. L.; FERGUS, R. Regularization of neural networks using dropconnect. In: International conference on machine learning. [S.l.: s.n.], 2013. p. 1058–1066. Citado na página 11.

WANG, L.; LU, Y.; WANG, H.; ZHENG, Y.; YE, H.; XUE, X. Evolving boxes for fast vehicle detection. In: IEEE. Multimedia and Expo (ICME), 2017 IEEE International Conference on. [S.l.], 2017. p. 1135–1140. Citado 2 vezes nas páginas 12 e 13.

WANG, T.; WU, D. J.; COATES, A.; NG, A. Y. End-to-end text recognition with convolutional neural networks. In: Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). [S.l.: s.n.], 2012. p. 3304–3308. Citado na página 17.

WEN, L.; DU, D.; CAI, Z.; LEI, Z.; CHANG, M.; QI, H.; LIM, J.; YANG, M.; LYU, S. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. arXiv CoRR, abs/1511.04136, 2015. Citado 5 vezes nas páginas 11, 24, 31, 32 e 48.

WIJNHOVEN, R. G. J.; WITH, P. H. N. de. Fast training of object detection using stochastic gradient descent. In: 2010 20th International Conference on Pattern Recognition. [S.l.: s.n.], 2010. p. 424–427. Citado na página 19.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? In: Advances in neural information processing systems. [S.l.: s.n.], 2014. p. 3320–3328. Citado 2 vezes nas páginas 18 e 27.

ZINKEVICH, M. A.; WEIMER, M.; SMOLA, A.; LI, L. Parallelized stochastic gradient descent. In: Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2. USA: Curran Associates Inc., 2010. (NIPS'10), p. 2595–2603. Citado na página 19.