

# **CLAUDEMIR CASA**

## **DETECTANDO PARTES HUMANAS EM AMBIENTES NÃO CONTROLADOS**

*(versão pré-defesa, compilada em 29 de março de 2021)*

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Inteligência Computacional*.

Orientador: Luciano Silva.

Coorientador: Olga Regina Pereira Bellon.

**CURITIBA PR  
2021**

# Resumo

O objetivo deste trabalho é apresentar um modelo de Rede Neural para a detecção e segmentação de partes do corpo humano. É disponibilizado um modelo multiplataforma de tempo real que pode ser executado em computadores comuns, assim como em dispositivos móveis e sistemas embarcados. Ele caracteriza-se por ser compacto, e por explorar uma parte da detecção de objetos ainda pouco utilizada. Uma das características marcantes é sua capacidade de reconhecer partes do corpo humano mesmo em ambientes não controlados, devido a utilização de um subset aleatório da base pública da Google (Open Images Dataset) que contém imagens com objetos nos mais variados tamanhos, posições, condições de iluminação e de oclusão. O modelo final possui a capacidade de detectar 18 classes distintas, incluindo classificação de gênero e de pessoas. A proposta principal é entregar algo que possa ser utilizado para a resolução de problemas específicos que necessitem da detecção e extração de partes do corpo humano, assim como aqueles de autenticação de usuários que necessitam extrair certas regiões para aumentar sua precisão.

**Palavras-chave:** Detecção de partes do corpo humano, modelo compacto, modelo de tempo real, redes neurais.

# Abstract

The objective of this work is to present a Neural Network model for the detection and segmentation of human's body parts. A real-time multiplatform model is available and can be run on standard computers, as well as mobile devices and embedded systems. It is characterized by being compact and exploiting a part of object detection that is still little used. One of the striking features is the ability to recognize parts of the human body even in uncontrolled environments, due to the use of a random subset of Google's public database (Open Images Dataset) that contains images with objects of the most varied sizes, positions, work conditions, lighting and occlusion. The final model has the capacity to detect 18 different classes, including gender and people classification. The main proposal is to deliver something that can be used to solve specific problems that require the detection and extraction of the human's body parts, as well as to authenticate users who need to extract certain regions to increase their accuracy.

**Keywords:** human's body parts detection, compact model, runtime model, neural networks.

# **Lista de Figuras**

1.1	Estimativa de pose baseada em partes do corpo humano. . . . .	10
2.1	Visão simplificada de neurônio artificial. . . . .	12
2.2	Rede neural comum versus de múltiplas camadas. . . . .	13
2.3	Visão de serviços cloud para visão computacional . . . . .	17
3.1	Exemplos de anotações do Google Open Images Dataset . . . . .	23
3.2	Fluxo de conversão do modelo HPNet . . . . .	24
3.3	Exemplos de resultados obtidos utilizando o modelo proposto. . . . .	25
3.4	Exemplos de resultados para mãos ao volante e detecção de pedestres . . . . .	28
A.1	Exemplos de imagens contendo oclusões. . . . .	36
A.2	Exemplos de imagens contendo oclusões. . . . .	37
A.3	Exemplos de imagens contendo variações de iluminação . . . . .	38
A.4	Exemplos de imagens contendo outros tipos de variações . . . . .	38
B.1	Resultados de predições obtidas com o HPNet . . . . .	39
B.2	Resultados de predições obtidas com o HPNet . . . . .	40
B.3	Gráfico de resultados por classe para o mAP . . . . .	40
B.4	Gráfico de resultados de detecção por classe . . . . .	41
B.5	Gráfico de ground-truth por classe . . . . .	42
C.1	Resultados de detecção de pedestres obtidos com o HPNet . . . . .	43
C.2	Resultados de detecção de pedestres obtidos com o HPNet . . . . .	44
C.3	Resultados de detecção de mãos ao volante obtidos com o HPNet. . . . .	44

# **Lista de Tabelas**

2.1	Arquitetura padrão YOLO . . . . .	15
2.2	Comparações de tempo de inferência. . . . .	16
3.1	HPNet arquitetura original . . . . .	22
3.2	Comparação entre abordagens compactas para detecção em computadores de baixo custo (sem uso de GPU's) . . . . .	25

# **Lista de acrônimos**

IA	Inteligência Artificial
CNN	Convolutional Neural Networks
ANN	Artificial Neural Networks
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ML	Machine Learning
PC	Personal Computer

# Sumário

<b>1</b>	<b>Introdução . . . . .</b>	<b>8</b>
1.1	Motivações . . . . .	9
1.2	Trabalhos relacionados . . . . .	10
1.3	Contribuições . . . . .	10
1.4	Limitações . . . . .	11
<b>2</b>	<b>Redes Neurais Artificiais . . . . .</b>	<b>12</b>
2.1	Redes Neurais Profundas . . . . .	13
2.1.1	Estado da arte . . . . .	14
2.1.2	Métodos e ferramentas . . . . .	15
<b>3</b>	<b>Proposta . . . . .</b>	<b>20</b>
3.0.1	Detalhes de implementação . . . . .	21
3.0.2	Métodos alternativos . . . . .	25
3.0.3	Aplicabilidade . . . . .	26
<b>4</b>	<b>Considerações finais . . . . .</b>	<b>29</b>
4.0.1	Otimização e melhorias futuras . . . . .	29
4.0.2	Conclusão . . . . .	30
	<b>Referências . . . . .</b>	<b>31</b>
	<b>Apêndice A: Exemplos de imagens obtidas do Google Open Images Dataset . . . . .</b>	<b>36</b>
A.1	Imagens contendo oclusões . . . . .	36
A.2	Variações de iluminação . . . . .	37
A.3	Outros tipos de variações . . . . .	38
	<b>Apêndice B: Exemplos de resultados obtidos utilizando o HPNet . . . . .</b>	<b>39</b>
B.1	Exemplos de predições . . . . .	39
	<b>Apêndice C: Estudo de caso . . . . .</b>	<b>43</b>
C.1	Detecção de pedestres e condução consciente . . . . .	43

# 1 Introdução

As redes neurais têm sido constantemente citadas quando se procura a solução para problemas específicos de detecção de objetos, interpretação inteligente de imagens, de áudios e conteúdos textuais. Praticamente há uma associação implícita entre o termo *inteligência artificial* e *redes neurais*, isso devido a grande utilização desse tipo de abordagem para resolver problemas relacionados a I.A. Um dos problemas em que mais se utilizam redes neurais atualmente, é na detecção de objetos. Geralmente para esse tipo de problema comumente utiliza-se uma arquitetura específica chamada "*Convolutional Neural Networks*" ou "*Redes Neurais Convolucionais*".

Em uma busca rápida pelo termo "*object detection*" através de ferramentas indexadoras de publicações científicas é possível encontrar muitas citações de diferentes abordagens relacionadas a detecção de objetos. São comuns citações a modelos/arquiteturas de redes neurais para detecção de objetos assim como R-CNN (Girshick et al. (2013)), R-FCNN (Dai et al. (2016)), FAST-RCNN (Girshick (2015)), FASTER-RCNN (Ren et al. (2015b)), MASK-RCNN (He et al. (2017a)), MS-CNN (Cai et al. (2016)), YOLO (Redmon e Farhadi (2018)), dentre outros.

De forma geral, apesar do grande número de publicações relacionadas a CNN's, existem poucas diferenças de grande impacto entre as abordagens apresentadas. Geralmente a arquitetura dessas redes é formada por uma sequência de camadas vários neurônios artificiais. Algumas possuem características particulares, assim como o MASK-RCNN que em parte de seu processo extrai uma máscara do objeto detectado, ou assim como o YOLO, se distinguem pelo número de classes de objetos e tempo de predição. Mas na grande maioria, o que realmente existe de maior relevância é a otimização do modelo, ou seja, a velocidade em que os objetos são detectados utilizando o menor número de recursos possíveis.

Uma das vantagens do grande interesse em CNN's é que temos a disposição várias ferramentas que nos permitem realizar a prototipagem, implementação e testes das mesmas. Muitas dessas ferramentas são disponibilizadas por grandes instituições, que utilizam técnicas algorítmicas consideradas como "estado da arte". Isso proporciona uma boa margem para obter bons resultados na implementação de novos modelos. Há também aquelas ferramentas na nuvem<sup>1</sup>, que aceleram ainda mais o processo de treino para os novos modelos.

É possível encontrar várias aplicações práticas utilizando-se de redes neurais para a detecção de objetos. Há alguns exemplos no setor automobilístico de carros autônomos, por exemplo, que utilizam tais técnicas para identificar pedestres, objetos de sinalização e também

---

<sup>1</sup>São ferramentas executadas em servidores com desempenho suficiente para acelerar o processo de treino.

para predizer situações inesperadas durante o trajeto (Wu et al. (2017); Kato et al. (2015)). Existem também aplicações mais específicas, como aquelas que utilizam redes neurais para realizar a contagem de objetos (Oñoro-Rubio e López-Sastre (2016)) e detecção de símbolos textuais (Zhang et al. (2017)).

Algumas metodologias utilizam CNN's para a autenticação de usuários de modo a proporcionar soluções mais seguras. Como exemplo, é válido citar aquelas de detecção de partes do rosto humano baseando-se em seus atributos<sup>2</sup>(Samangouei et al. (2017); Yang et al. (2015)). Esse tipo de metodologia utiliza a segmentação de partes do rosto para aumentar a precisão no momento da autenticação.

Com base em busca na literatura atual, é possível observar que a detecção de partes do corpo humano em ambientes não controlados utilizando técnicas de detecção de objetos é algo explorado com cautela. A maior parte das pesquisas está direcionada à detecção de objetos comuns. É importante ressaltar a grande dificuldade em detectar partes do corpo humano em ambientes com muitas variações utilizando técnicas comuns de detecção de objetos. As partes do corpo humano não possuem formas tão definidas como os objetos comuns, por exemplo, uma maçã possui forma bem definida enquanto o cabelo humano possui muitas variações.

Com base nas informações apresentadas sobre CNN's, é proposto neste trabalho um modelo de Rede Neural para a detecção de partes do corpo humano em ambientes não controlados. A proposta está direcionada a entregar um modelo que possa ser utilizado em aplicações que necessitem detectar e segmentar tais partes, isso para qualquer problema dentro desse escopo. A intenção não é oferecer o modelo mais preciso e sim aquele que possa ser utilizado na resolução da maioria dos problemas desse escopo, bem como aquele com a maior velocidade de resposta e que possa ser utilizado na maioria das plataformas atuais<sup>3</sup>.

## 1.1 Motivações

Dentre os motivos para a elaboração deste trabalho, aqueles de maior destaque estão relacionados a otimização e entrega de um modelo específico para a defecção de partes do corpo humano. É possível encontrar vários exemplos de implementações de modelos e abordagens diferentes para a detecção de objetos comuns (Ren et al. (2015a); Redmon et al. (2016); Lin et al. (2017)), mas com base nas pesquisas realizadas, até o momento, nenhum tão específico como o apresentado.

Também é importante destacar que a criação de um modelo para ser executado em ambientes não controlados se enquadra nas motivações. Devido a grande quantidade de imagens da base de dados gerada e das variações encontradas nessa mesma base, isso despertou um grande interesse em visualizar como o resultado final se comportaria e em como ele poderia ser utilizado em outros escopos.

---

<sup>2</sup>Características como cor dos olhos, distância entre partes do rosto, forma das partes, etc.

<sup>3</sup>Dispositivos computacionais assim como computadores de mesa, smartphones, dispositivos embarcados, etc.

Quanto a otimização citada anteriormente, ela está diretamente relacionada em tornar o modelo o mais rápido e compacto possível, além de torná-lo executável na maioria das plataformas atuais. Existem muitas abordagens que realmente são otimizadas, porém devido a complexidade da arquitetura do modelo, ora não podem ser executadas em tempo real sem auxílio de hardware específico, ora não podem ser executados em mais de uma plataforma.

## 1.2 Trabalhos relacionados

Nas pesquisas realizadas encontramos alguns trabalhos relevantes e que devem ser citados.

Yang ( Yang et al. (2015, 2016)) apresenta, respectivamente uma abordagem para a detecção facial baseando-se em partes do rosto para aumentar a precisão, e uma abordagem para estimar poses humanas se baseando na utilização de partes do corpo humana e o relacionamento entre elas. Um outro exemplo é o trabalho de Tian Y. (Tian et al. (2015)) que utiliza também partes do corpo humano para aumentar a precisão na detecção de pedestres. A. Jalal (Jalal et al. (2019)) apresenta uma proposta para estimar a pose em esportes físicos utilizando partes do corpo humano, vide Figura 1.1. Hai-Wen Chen (Chen e McGurr (2016)) propôs um método baseado na intensidade das cores para estimar atividades humanas, padrões de caminhada e reconhecimento facial.

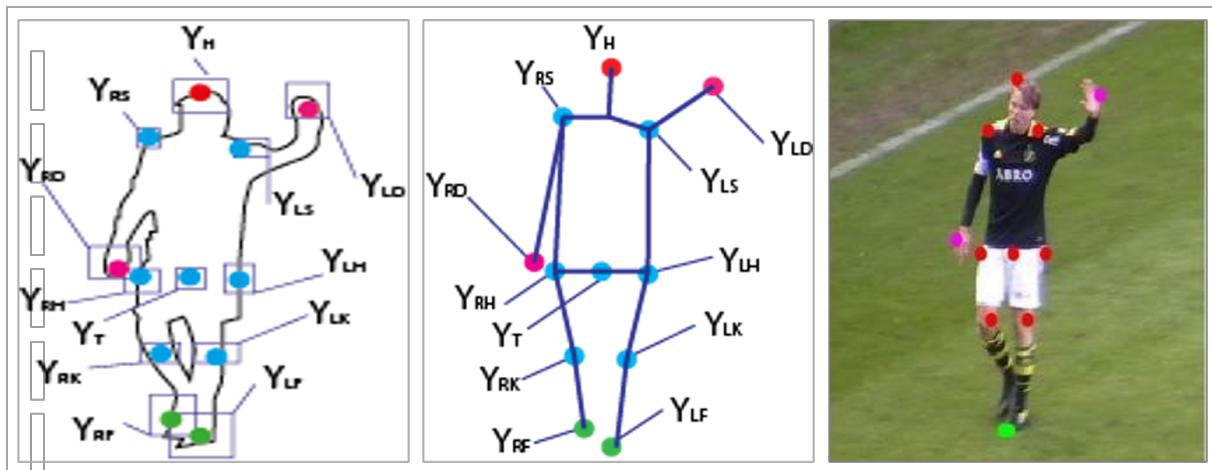


Figura 1.1: Estimativa de pose baseada em partes do corpo humano.

Fonte: Artigo do autor

## 1.3 Contribuições

Entre as contribuições de maior relevância, é válido citar:

- Um modelo específico para a detecção de partes do corpo humano.

- Um modelo compacto.
- Um modelo que pode ser executado em tempo real sem a utilização de Unidades graficas de processamento (GPU's).
- Um modelo que pode ser executado sem qualquer alteração em computadores de mesa, dispositivos móveis, embarcados e também em navegadores.
- Um modelo que contempla 14 classes <sup>4</sup> diferentes do corpo humano e classificação de gênero.

## 1.4 Limitações

Algumas das limitações encontradas durante o desenvolvimento e que remetem diretamente no resultado final, são o número limitado de recursos (físicos) para o treino do modelo e o tempo necessário entre análise e ajustes de parâmetros (O grande número de imagens na base de dados demanda um grande tempo e poder computacional para processá-las). Existe também as limitações de precisão do modelo devido a sua propriedade compacta. Sendo um modelo compacto, logicamente a arquitetura/topologia<sup>5</sup> da rede neural tiveram que ser reduzidas, impactando diretamente na precisão do mesmo. Mas mesmo assim, comparado ao modelo base, o modelo proposto possui uma precisão e tempo de resposta superiores.

---

<sup>4</sup>Pessoas, Homem, Mulher, Garoto, Garota, Cabeça, Face, Olhos, Sombrancelhas, Nariz, Boca, Orelhas, Cabelo, Barba, Pernas, Braços, Pés e Mão.

<sup>5</sup>Refere-se a como os neurônios artificiais foram organizados nas camadas da rede neural.

## 2 Redes Neurais Artificiais

Para que seja possível compreender de forma simples e precisa as Redes Neurais Artificiais (ANN's), é necessário abordar sua origem e respectivamente o conceito de neurônios artificiais. Com base nessas informações será possível uma completa compreensão das arquiteturas de redes neurais e as redes profundas.

Apesar do crescente interesse na inteligência artificial e em redes neurais nas últimas décadas, o conceito é bem antigo e foi apresentado pela primeira vez por volta de 1943 pelos neurofisiologistas Warren McCulloch e o matemático Walter Pitts (McCulloch e Pitts (1943)). Eles desenvolveram o primeiro modelo de neurônio artificial baseando-se nos neurônios humanos, utilizando artifícios matemáticos e algoritmos chamados de lógica de limiar (Hampel e Winder (1971))<sup>1</sup>.

O conceito de neurônio artificial é bem simples, ele foi inteiramente baseado nos neurônios reais. Assim como nos neurônios reais, os dentritos são representados pelas entradas nos neurônios artificiais. o corpo do neurônio realiza uma soma ponderada das entradas pelos pesos em cada entrada. O valor excitatório ou inibitório é definido por um limiar(threshold), e em seguida uma função de ativação define a saída do neurônio. A figura 2.1 apresenta uma visão simplificada de um neurônio artificial.

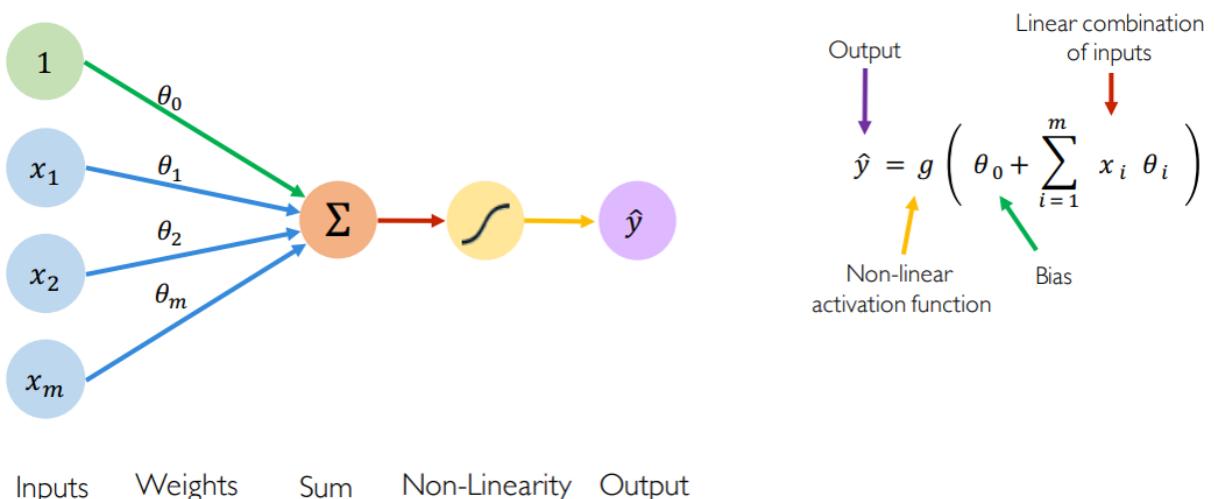


Figura 2.1: Visão simplificada de neurônio artificial.

Fonte: Google

<sup>1</sup>Aplicam um limiar (threshold) a uma entrada binária com objetivo de filtrar uma saída, também binária.

Olhando para apenas um neurônio artificial, aparentemente ele não possui nada de muito surpreendente. Porém se forem realizadas a união de vários neurônios distribuídos em camadas, é possível criar arquiteturas de redes para resolver diversos tipos de problemas. E se a arquitetura for programada para ajustar seus pesos de forma automática, o resultado é um dispositivo inteligente extremamente eficiente. Na verdade é exatamente assim que funcionam as redes neurais, elas recebem uma grande quantia de dados e realizam os ajustes dos pesos com base nas entradas, tornando possível prever um resultado mesmo para valores diferentes.

A combinação de neurônios artificiais em quantidades e camadas diferentes, proporcionou ao longo dos anos, a criação de diferentes arquiteturas de redes neurais. Através dessas arquiteturas é possível resolver problemas diferentes, como por exemplo: reconhecimento facial (Cheng et al. (2019)), reconhecimento de fala e de escrita (Hori et al. (2017)), criação de obras de arte digitais (Nikolić et al. (2018)), controle de carros autônomos (Chishti et al. (2018)), dentre outros.

## 2.1 Redes Neurais Profundas

As redes neurais profundas utilizam mais de uma camada interna (oculta) em sua arquitetura, podendo conter um número variável de neurônios artificiais dependendo da arquitetura utilizada, assim como mostra a figura 2.2. Os primeiros estudos relacionados a redes neurais de múltiplas camadas (ou redes neurais profundas), estão relacionados ao matemático soviético Alexey Grigorevich Ivakhnenko Valentin Grigor'evich Lapa (autor de Cybernetics and Forecasting Techniques) em 1965 (Ivakhnenko e Lapa (1967)).

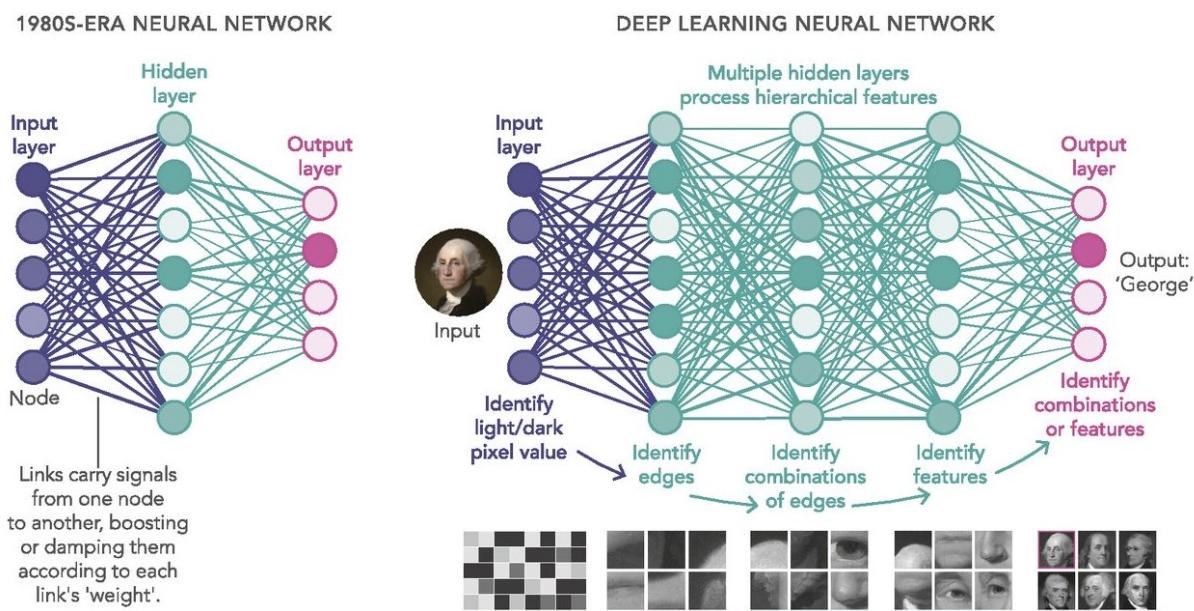


Figura 2.2: Rede neural comum versus de múltiplas camadas.

Fonte: Google

A primeira rede neural profunda utilizava funções de ativação polinomial que foram analisadas manualmente através de métodos estatísticos. As camadas da rede eram analisadas separadamente selecionando os melhores recursos para serem passados adiante para as camadas posteriores. A técnica de backpropagation<sup>2</sup> ainda não havia sido descoberta, tornando o processo lento e totalmente manual.

Em 1989, um cientista da computação chamado Yann LeCun, conseguiu demonstrar de forma funcional uma aplicação do backpropagation para ler dígitos manuscritos em cheques no Bell Labs (LeCun et al. (1989)). Durante os anos seguintes a utilização de redes neurais de múltiplas camadas não teve avanço significativo, isso devido ao exagero no potencial imediato das redes neurais e as limitações de *hardware*. Porém em 1999 iniciou-se a era das Unidades Gráficas de Processamento (GPU's), possibilitando assim um enorme salto no poder computacional e auxiliando na proliferação das redes neurais.

Durante os anos seguintes foi possível observar um grande avanço na elaboração e implementação de redes neurais de múltiplas camadas. Vários estudos direcionados ao desenvolvimento de arquiteturas de redes diferentes e suas funções de ativação, tonaram possível que redes neurais de múltiplas camadas reconheçam objetos, fala, expressões faciais, a até mesmo sejam capazes de pilotar um veículo. Dentre as várias arquiteturas e implementações de algoritmos de redes neurais de múltiplas camadas, os de maior interesse são aqueles para a detecção de objetos.

### 2.1.1 Estado da arte

Existem várias abordagens diferentes no contexto da detecção de objetos implementadas utilizando redes neurais de múltiplas camadas (*deep learning*). O *Mask R-CNN*(He et al. (2017a)), por exemplo, além de entregar a localização do objeto (*bounding box*) e sua classe, também entrega a máscara (a forma) do objeto na saída da rede. Apesar de existirem abordagens diferentes na detecção de objetos, o modelo de rede neural com a maior precisão e menor tempo na predição (constituindo o estado da arte na detecção de objetos) é o YOLO (Redmon e Farhadi (2018)).

O YOLO (*You Look Once*) utiliza uma abordagem que consiste em dividir a imagem de entrada em uma grade de 13x13 células, sendo que cada uma dessas células é responsável em predizer 5 regiões (*bounding box*). O modelo é formado por apenas uma rede neural que é aplicada a imagem inteira para predizer objetos em cada célula, e ao final, os objetos são filtrados pelas probabilidades das predições (Redmon et al. (2016)). O modelo apresentado por Joseph Redmon, é uma rede neural profunda (*deep learning*) contendo 19 camadas de convoluções, assim como apresentado na tabela 2.1.

A versão mais recente do YOLO difere das demais em sua arquitetura, pois utiliza 53 camadas de convoluções e predições em 3 escalas diferentes, além de outras particularidades. Essas alterações na arquitetura trouxeram um ganho significativo ao modelo final, tornando-o

---

<sup>2</sup>O backpropagation é um algoritmo utilizado para treinar uma rede neural. Ele atualiza os pesos da rede calculando o gradiente da função de perda na camada final ((Vemuri, 2019, p. 241)).

(a)				
	Type	Filters	Size	Output
1x	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
2x	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64x64
	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
8x	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
8x	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
	Convolutional	256		
	Convolutional	512		
4x	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
Residual				
8 x 8				
Avgpool				
Connected				
Softmax				

Tabela 2.1: Arquitetura padrão YOLO

Fonte: Artigo do autor

assim estado da arte na detecção de objetos. A tabela 2.2 apresenta o tempo de inferência comparando com outras metodologias. É possível observar que o YOLO possui uma grande vantagem em relação as outras metodologias, principalmente para uma entrada de 320 *pixels*.

### 2.1.2 Métodos e ferramentas

As redes neurais de múltiplas camadas geralmente funcionam de forma semelhante quanto ao treino e validação. O primeiro passo é escolher uma ferramenta para criar a arquitetura da rede. A maioria dos trabalhos sobre redes neurais utilizam ferramentas de terceiros para implementar suas arquiteturas, pois existem grandes empresas que investem continuamente na elaboração dessas ferramentas, e essas empresas possuem maturidade suficiente para entregar ferramentas precisas e confiáveis.

Após escolher as ferramentas adequadas de arquitetura, é preciso preparar o *dataset*<sup>3</sup> desejado, configurar os parâmetros iniciais e realizar o treino da rede. Após o treino é preciso

<sup>3</sup>O banco de dados contendo as informações desejadas (Chu et al. (2016)).

Method	mAP-50	Time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	<b>59.1</b>	172
RetinaNET-50-500	50.9	73
RetinaNET-101-500	53.1	90
RetinaNET-101-800	57.5	198
<b>YOLOv3-320</b>	51.5	<b>22</b>
<b>YOLOv3-416</b>	55.3	29
<b>YOLOv3-608</b>	57.9	51

Tabela 2.2: Comparações de tempo de inferência

Fonte: Artigo do autor

reajustar os parâmetros se necessário, entregando ao término um modelo que resolve o problema de estudo (Druzhkov e Kustikova (2016)).

Um modelo de rede neural é um arquivo binário contendo a estrutura da rede, ou seja, como os neurônios estão dispostos em suas camadas, e os pesos ajustados de todos os neurônios (Liu et al. (2017); Zhang e Zhu (2018)). O tamanho desse arquivo está diretamente relacionado a quantia de neurônios e camadas utilizadas e o tamanho do *dataset* (quanto maiores os números, maior será o modelo).

É importante compreender a relação de tamanho de um modelo, pois o seu tamanho tem impacto direto na execução. Quanto maior, mais complexa sua distribuição, e consequentemente, maior será o tempo de carregamento e execução. É importante ressaltar que um modelo complexo (com muitos neurônios e camadas) também demanda um maior tempo de treino, pois uma quantidade maior de pesos precisam ser ajustados.

### *Ferramentas na nuvem*

Uma das maiores dificuldades encontradas no treinamento de uma rede neural, independente do contexto, esta relacionada aos recursos de *hardware* necessários. Atualmente são utilizadas GPU's para acelerar o processo de treino e predição, isso devido ao alto poder de processamento desse tipo de dispositivo. Mas para aqueles usuários que não possuem acesso aos requisitos mínimos, a tarefa de treino e predição torna-se algo complexo e custoso. Porém, é possível utilizar soluções na nuvem para sanar essas dificuldades (Feng et al. (2018)).

Grandes empresas como Google, Microsoft e Amazon, disponibilizam ferramentas em nuvem para a configuração e treino de redes neurais, e se for necessário apenas recursos de hardware, é possível alocar poder de processamento nessas plataformas. O Google, por

exemplo, disponibiliza a opção de treino de redes neurais utilizando o TensorFlow<sup>4</sup> em seu serviço chamado "*Google Cloud*". A Microsoft disponibiliza o "*Microsoft Cognitive Services*" em seu serviço chamado Azure, e que já possui conversão nativa entre modelos, ferramenta de anotação de imagens e serviços de estatísticas. A Amazon não possui uma ferramenta própria para redes neurais, mas da suporte a várias ferramentas de terceiros. A Figura 2.3 apresenta uma visão simplificada do fluxo de operação do serviço de visão computacional da Microsoft.

Os serviços em nuvem tem ganhado tanto destaque, que agora não existem apenas soluções para treino e predição de Redes Neurais, mas também modelos distribuídos sendo explorados em soluções na nuvem (Li et al. (2012); Teerapittayanon et al. (2017)). Esse tipo de solução é muito eficiente mas complexa de se gerenciar. Nessa abordagem camadas e/ou neurônios estão distribuídos em diferentes computadores na nuvem e requerem *softwares* específicos para realizar seu gerenciamento.

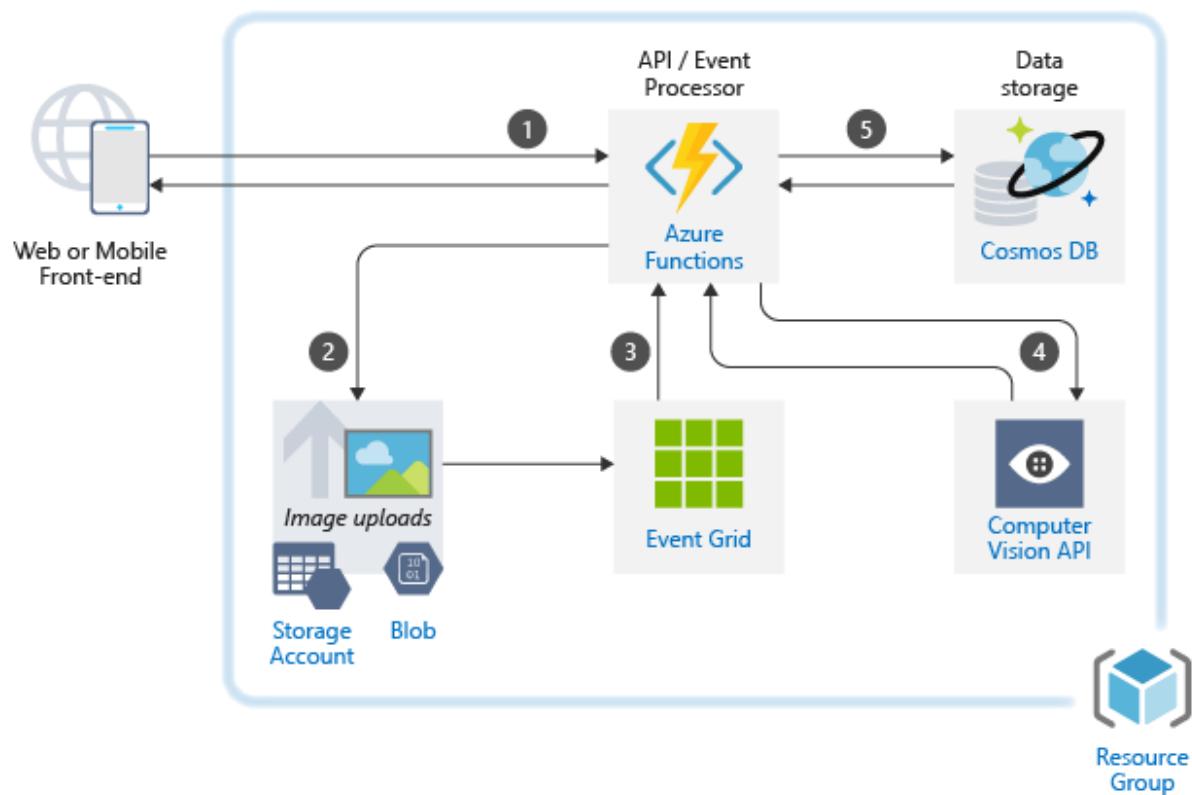


Figura 2.3: Visão de serviços cloud para visão computacional

Fonte: Google

### *Ferramentas locais*

Existem também ferramentas que podem ser executadas diretamente em computadores locais dos usuários que dispõem dos recursos necessários. Dentre as ferramentas locais mais utilizadas, aquela de maior destaque é o já citado TensorFlow. Ela possui um grande número

<sup>4</sup>Framework desenvolvido pela Google que implementa vários algoritmos de *Machine Learning*.

de recursos, uma implementação estável e é altamente aceita pela comunidade. Mas além do TensorFlow, existem várias outras que podem ser utilizadas localmente, cada uma com suas particularidades. Dentre elas podemos citar:

**Chainer** É uma ferramenta de código aberto com suporte a várias arquiteturas de rede, podendo ser utilizada para processamento em CPU's e GPU's.

**Cognitive Toolkit** Desenvolvido pela Microsoft para as linguagens C#, C++ e Python. Também possui suporte nativo ao formato ONNX.

**mxnet** Desenvolvido pela Apache para 8 linguagens diferentes, sendo possível a utilização de treino distribuído.

**PyTorch** Biblioteca desenvolvida para a linguagem python e amplamente utilizada pela comunidade.

**PaddlePaddle** Ferramenta distribuída para treino em múltiplas máquinas (paralelo).

**Matlab** Modulo de *ML (Machine Learning)* para a ferramenta matemática Matlab.

**SAS** Módulo de *ML (Machine Learning)* em python para o SAS.

**Neural Network Libraries** Biblioteca de *ML (Machine Learning)* implementada e mantida pela Sony.

Há também o *framework* Darknet, que serviu de base para esse trabalho. Esse conjunto de ferramentas foi escrito na linguagem C, destacando-se pelo grande desempenho e podendo ser executado em múltiplos CPU's e GPU's. Ele também foi utilizado para implementar o modelo YOLO de detecção de objetos.

O Darknet é configurado via arquivos \*.cfg, que servem, além de definir parâmetros e configurações, para construir a arquitetura da rede. Além desses arquivos, existe o executável binário que apresenta informações sobre os parâmetros aceitos através da linha de comando. Também é possível executar exemplos utilizando bases de dados como coco (Lin et al. (2014)), imagenet (Deng et al. (2009)) e Pascal VOC (Everingham et al. (2015)). Mais detalhes de implementação do Darknet podem ser encontrados no repositório público do autor em <https://github.com/pjreddie/darknet>.

### *Representações intermediárias de modelos*

Uma representação intermediária torna possível expressar modelos de redes neurais de forma mais simples. A maior vantagem em se utilizar uma representação desse tipo, é que através dela pode ser realizada uma conversão entre diferentes tipos de modelos, além de tornar possível sua execução de forma universal em vários dispositivos. De forma simples, podemos concluir que a representação intermediaria atua entre o modelo inicial e seu executor.

Duas das maiores empresas de tecnologia da atualidade, a Microsoft e o Facebook, decidiram unir seus conhecimentos para criar um *framework* com objetivo de representar, de forma universal, modelos de redes neurais. Com isso surgiu o "*Open Neural Network Exchange*" ou apenas ONNX, que é uma ferramenta que permite a representação de modelos de forma intermediaria através de operadores e tipos de dados universais. Um operador pode ser considerado um filtro que atua sobre a entrada, por exemplo, o uma camada de soma na rede nada mais é do que um operador de soma. O interessante é que o ONNX permite que os desenvolvedores criem seus próprios operadores e tipos de dados. Assim, quando um novo operador é desenvolvido pela comunidade científica, ele pode ser implementado no *framework*.

Através do ONNX é possível, por exemplo, treinar um modelo no TensorFlow e converte-lo para o Cognitive Toolkit, ou para qualquer ferramenta citada anteriormente. O grande potencial do ONNX, é que torna mais fácil compartilhar modelos entre diferentes tipos de projetos, o que antes não era possível, era preciso rescrever a rede neural para cada ferramenta. Os modelos exportados no formato intermediário ainda podem ser executados de forma nativa em um grande número de ferramentas de tempo real, dentre as quais podemos citar:

**NVIDIA TensorRT** Criada pela NVIDIA para acelerar e otimizar modelos de redes neurais.

**Qualcomm** Otimização para arquitetura mobile.

**Bitmain** *Hardware* específica para a aceleração de modelos de redes neurais.

**Tencent** *Framework* de alta performance otimizado para dispositivos móveis.

**Vespa** Framework para Big Data.

**ONNXRuntime** Otimizador para a plataforma Windows.

**synopsys** Conjunto de ferramentas otimizados para embarcados.

**CEVA** Compilador otimizado para CEVA-XM Vision DSPs e processadores NeuPro AI.

**MACE** Criado pela Xiaomi para dispositivos móveis.

**Habana** *Hardware* de processamento gráfico proprietário.

Um dos pontos negativos do ONNX é que não há uma ferramenta oficial para a conversão de modelos escritos com o Darknet, sendo necessário converter para alguma outra ferramenta intermediária, como por exemplo, o TensorFlow, para que a conversão do modelo seja possível. Há também uma limitação na conversão entre versões, e dependendo dos operadores contidos na arquitetura, a conversão torna-se inviável.

### 3 Proposta

Baseado nas informações apresentadas, este trabalho apresenta como proposta um modelo de rede neural com múltiplas camadas (deeplearning) para detectar e segmentar partes do corpo humano em ambientes não controlados. Nossa modelo foi projetado para detectar 18 classes diferentes relacionadas ao ser humano e pode ser executado em qualquer plataforma (com ou sem auxilio de GPU's) em tempo real sem a necessidade de modificações. As classes do modelo estão divididas da seguinte forma:

**Humanos** : Detecção de pessoas (corpo inteiro)<sup>1</sup>.

**Gênero** : Detecção de 4 gêneros (adultos e crianças)<sup>2</sup>.

**Partes do corpo** : Detecção das partes do corpo humano<sup>3</sup>.

A proposta para esse modelo surgiu devido ao pequeno interesse por parte da comunidade na elaboração de uma rede neural para esse escopo em específico (detecção de distintas partes do corpo humano em ambientes não controlados utilizando técnicas de detecção de objetos comuns). Existe um grande interesse na utilização de modelos para a detecção de objetos, porém eles não são voltados a partes do corpo. Há também algumas pequenas citações utilizando métodos mais simples como *Histograms of Oriented Gradients* (Dalal e Triggs (2005)) para detectar algumas partes específicas do corpo, ou *Haar Cascades* (Wilson e Fernandez (2006)) para algumas partes da face, porém nenhum modelo completo de rede neural para a detecção completa de todas as partes visíveis do corpo humano.

A princípio, o modelo final é disponibilizado sendo uma rede de um único estágio<sup>4</sup> capaz de detectar e segmentar partes do corpo humano. Os esforços também foram direcionados em disponibilizar um modelo otimizado e universal para ser executado em múltiplas plataformas, pois a maioria das arquiteturas de redes neurais são desenvolvidas para executar em computadores contendo um grande poder computacional (o que não é a realidade da maioria das aplicações que são executadas em ambiente mobile/embarcados e seus usuários).

O modelo também foi configurado e treinado de forma a conseguir resultados semelhantes mesmo para aquelas entradas com muitas variações, como por exemplo, vindas de uma *webcam*.

---

<sup>1</sup>Person

<sup>2</sup>Man, Woman, Boy e Girl

<sup>3</sup>\*Human head, face, eye, eyebrow, nose, mouth, ear, hair, beard, leg, arm, foot e hand

<sup>4</sup>A detecção e segmentação são realizadas em uma única rede neural.

Isso foi possível devido a grande variação no *dataset* de imagens e pela combinação de alguns parâmetros da rede.

### 3.0.1 Detalhes de implementação

A primeira versão do modelo foi treinada utilizando as ferramentas em nuvem da Microsoft (*Computer Vision*). O processo de treino dentro dessa plataforma é bem simples, bastando apenas enviar as imagens rotuladas ou realizar a marcação dentro da própria ferramenta. Porém não estão disponíveis detalhes da arquitetura utilizada dentro da ferramenta e também não é possível o ajuste manual de parâmetros. Um outro problema encontrado nessa abordagem é a necessidade de escrever um programa para enviar as imagens rotuladas em lotes. Há também um limite de imagens que podem ser enviadas simultaneamente, tornando o processo de upload lento e trabalhoso para grandes *datasets*.

As soluções em nuvem também não são uma opção de baixo custo em alguns casos, se levarmos em consideração o treino para a base de dados utilizadas em nosso modelo, gastamos aproximadamente \$2.000 dólares em treino para chegar em um resultado aceitável. O conjunto de pontos apresentados inviabilizou a utilização dessas ferramentas em nuvem para o treino de nosso modelo. Isso não exclui a possibilidade de utilizar esse tipo de abordagem em outros tipos de situações.

A decisão tomada foi em utilizar um treino local utilizando uma variação do YOLO em sua versão mais recente. Foram levados em consideração pontos como os resultados obtidos utilizando o YOLO em outros trabalhos, a facilidade em utilizar a ferramenta e sua documentação bem estruturada e completa. Nossa modelo é uma variação compacta contendo três camadas do YOLO, sendo que os parâmetros estão ajustados para que seja possível a predição mesmo nas imagens com variações em nosso dataset. Devido as escolhas utilizadas na elaboração da arquitetura da rede, o modelo é capaz de realizar predições em imagens com variações de tamanho, iluminação, oclusão, dentre outros.

Além da arquitetura possuir otimizações nos parâmetros de treino<sup>5</sup>, ela foi construída utilizando três camadas de excitação manual (*Assisted Excitation of Activations*) (Derakhshani et al. (2019)). Esse tipo de técnica permite excitar certas camadas que possuem maior relevância na rede de forma manual, trazendo um ganho significativo as predições finais. A descrição da arquitetura original para o modelo é apresentada na Tabela 3.1.

A base de dados utilizada para o treino do modelo é composta por mais de 200.000 imagens que foram extraídas aleatoriamente do "*Open Images Dataset*" (Kuznetsova et al. (2020); Benenson et al. (2019); Pont-Tuset et al. (2020a); Krasin et al. (2017)). Essa base de dados foi elaborada pelo Google, contém em torno de 9 milhões de imagens e é utilizada em vários de seus modelos de redes neurais, além de disponibilizar um grande número de imagens anotadas e verificadas por humanos ao invés de robôs automatizados. Entre algumas das vantagens em

---

<sup>5</sup>Detalhes de implementação e otimizações podem ser encontrados no repositório oficial do trabalho.

<b>Layer</b>	<b>Filters</b>	<b>Size</b>	<b>Input</b>	<b>Output</b>
0 conv		3x3/1	608x608x3	608x608x16
1 max	16	2x2/2	608x608x16	304x304x16
2 conv		3x3/1	304x304x16	304x304x32
3 max	32	2x2/2	304x304x32	152x152x32
4 conv		3x3/1	152x152x32	152x152x64
5 max	64	2x2/2	152x152x64	76x76x64
<b>6 ae_conv</b>	128	3x3/1	76x76x64	76x76x128
7 max		2x2/2	76x76x128	38x38x128
8 conv		3x3/1	38x38x128	38x38x256
9 max	256	2x2/2	38x38x256	19x19x256
10 conv		3x3/1	19x19x256	19x19x512
11 max	512	2x2/2	19x19x512	19x19x512
12 conv	1024	3x3/1	19x19x512	19x19x1024
13 conv	256	1x1/1	19x19x1024	19x19x256
14 conv	512	3x3/1	19x19x256	19x19x512
15 conv	69	1x1/1	19x19x512	19x19x69
16 yolo				
17 route	13			19x19x256
18 conv	128	1x1/1	19x19x256	19x19x128
19 upsample		2x	19x19x128	38x38x128
20 route	19, 8			38x38x384
21 conv	256	3x3/1	38x38x384	38x38x256
22 conv	69	1x1/1	38x38x256	38x38x69
23 yolo				
24 route	21			38x38x256
25 conv	128	1x1/1	38x38x256	38x38x128
26 upsample		2x	38x38x128	76x76x128
27 route	26, 6			76x76x256
28 ae_conv	128	3x3/1	76x76x256	76x76x128
29 ae_conv	69	1x1/1	76x76x128	76x76x69
30 yolo				

Tabela 3.1: HPNet arquitetura original

utilizar um subset de imagens do dataset público do Google, é que há variações extremas nas imagens, é possível utilizar utilizar tipos variados de anotações<sup>6</sup>. Existe também a possibilidade de extrair um subset apenas de imagens anotadas por humanos, garantindo uma maior precisão e confiabilidade nas anotações. O dataset é dividido da seguinte maneira: 50% das imagens são utilizadas para treino (sendo que 10% dessas são utilizadas para a validação) e 50% das imagens são anotações de classes que não estão contidas no modelo para uma melhor normalização da rede nas previsões em imagens distintas. A Figura 3.1, apresentam exemplos de anotações realizadas no dataset OID. Também é possível visualizar uma lista mais detalhada de imagens no Apêndice A.

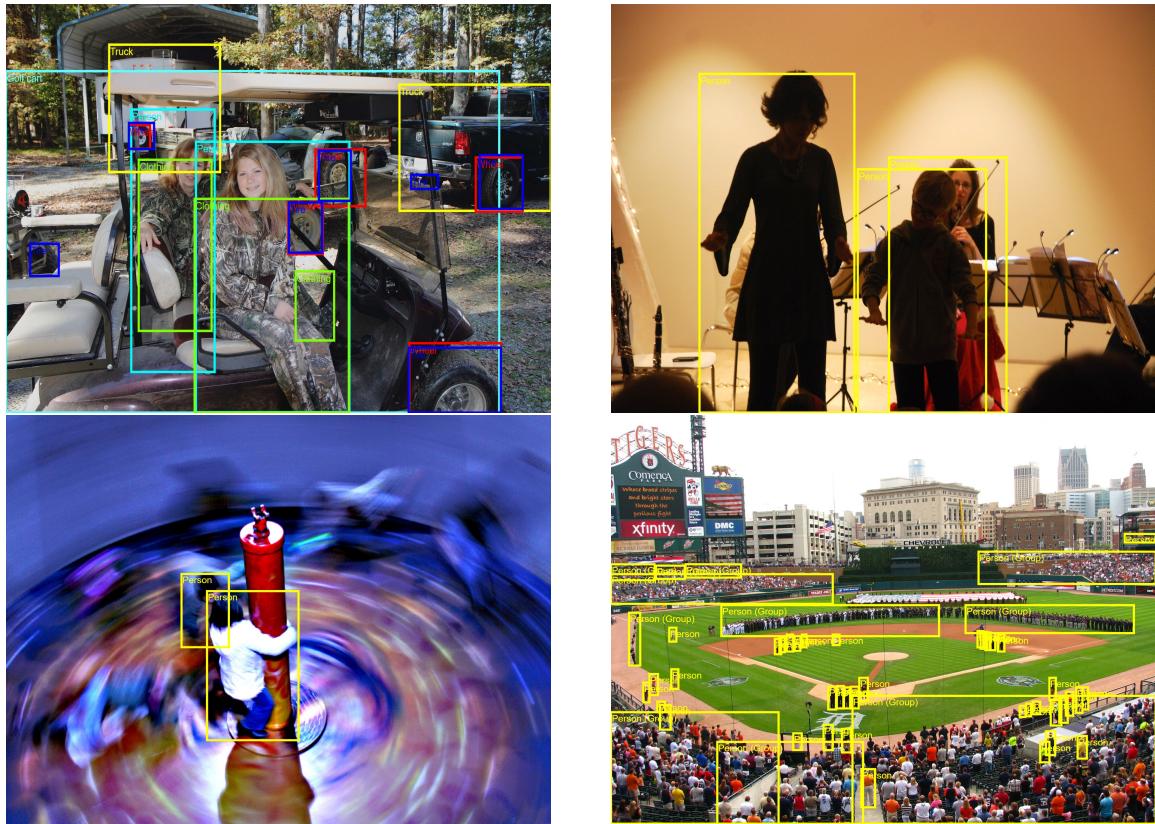


Figura 3.1: Exemplos de anotações do Google Open Images Dataset

Fonte: Google

Para a representação do modelo (arquitetura e binário) foi utilizado o ONNX (*Open Neural Network Exchange*) já apresentado anteriormente. Esse padrão de representação permite que o modelo seja executado de forma universal e sem alterações em diversos dispositivos diferentes. O binário inicial treinado utilizando o Darknet não exportado nativamente para ONNX, é preciso realizar uma conversão utilizando as ferramentas adequadas.

Logo após o treino no Darknet é necessário executar a conversão para uma representação aceita no framework PyTorch. Essa etapa é essencial pois a ferramenta Pytorch é a única que da suporte a conversão entre as representações do Darknet para outras de forma precisa e funcional.

<sup>6</sup>Detecção, segmentação, relações entre objetos e narrativas localizadas (Pont-Tuset et al. (2020b))

Na última etapa é necessário realizar uma outra conversão (agora para ONNX) utilizando a ferramentas yolov5 da Ultralytics<sup>7</sup>. Após obter o binário já no formato universal, é executada então uma otimização de operadores utilizando as ferramentas do próprio ONNX para eliminar operações desnecessária e tornar o modelo mais rápido e preciso. O fluxo simplificado do processo pode ser visualizado na Figura 3.2.

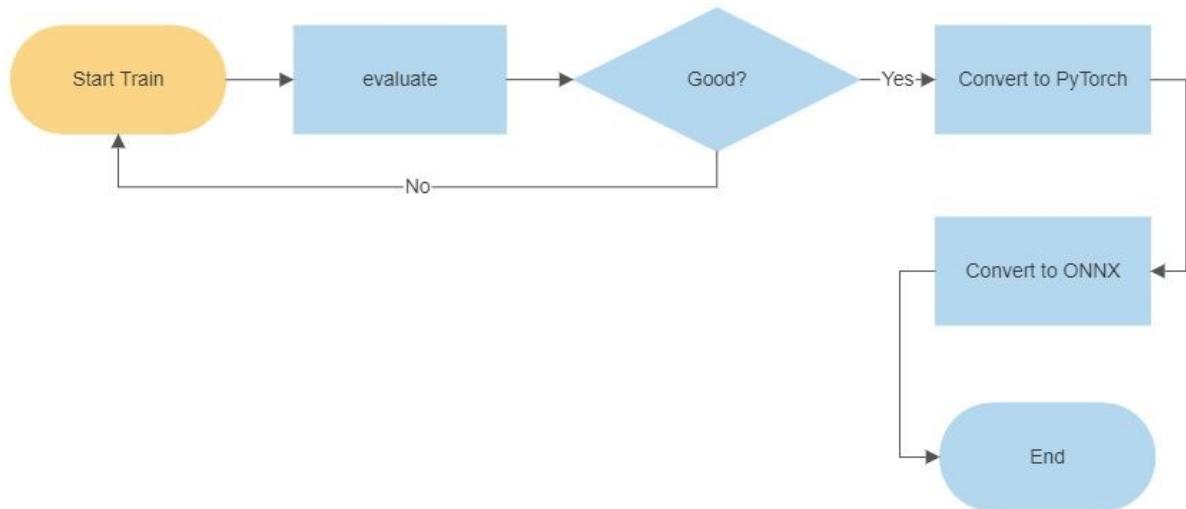


Figura 3.2: Fluxo de conversão do modelo HPNet

Os resultados obtidos nos primeiros experimentos foram condizentes com os primeiros treinos. Foi alcançado um percentual de **20%** de precisão nas previsões, quando a precisão encontrada no estado da arte para modelos compactos é de **33%**. Após uma análise mais crítica e otimização dos parâmetros, um novo resultado de **37%** foi obtido, superando assim o modelo base<sup>8</sup>. Apesar do percentual não ser elevado como o do modelo completo para uso em computadores com grande poder de processamento, é preciso considerar a perda pela otimização no tamanho e desempenho da arquitetura da rede. Por se tratar de um modelo compacto, o número de parâmetros e operações é drasticamente reduzido. Um outro fator importante para o maior percentual de precisão nos modelos completos (não compactos), é que eles realizam a previsão de objetos com formas bem definidas e mais simples de classificar (comparado as formas detectadas no modelo apresentado nesse trabalho, que possuem muitas variações). A Tabela 3.2 apresenta uma comparação de resultados entre os modelos mais utilizados e o apresentado nesse trabalho.

Para compreender a real dificuldade em detectar formas pouco definidas citadas anteriormente, veja como exemplo a detecção de uma parte do corpo humano em específico, a barba em pessoas do sexo masculino. A barba não tem forma bem definida quanto um carro por exemplo, é bem possível que uma barba possa ser confundida com uma árvore com folhagens

<sup>7</sup>A ferramenta pode ser encontrada em <https://github.com/ultralytics/yolov5>

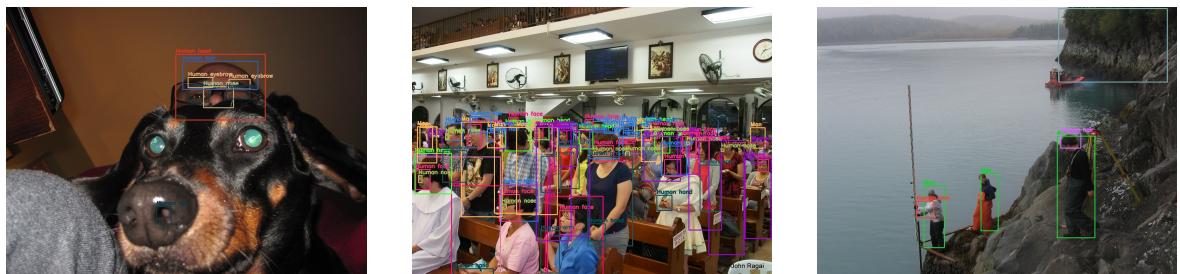
<sup>8</sup>Os detalhes de implementação e históricos de alterações no código-fonte podem ser encontrados na documentação oficial em <https://claudemircasa.github.io/hptiny/>

Model	Train	Test	mAP	FPS	Classes	Context
<b>Tiny YOLO</b>	OID	test-dev	23.7	~4	80	Generic
<b>YOLOv3-tiny</b>	OID	test-dev	33.1	~4	80	Generic
<b>HPTiny (ours)</b>	OID	test-dev	37.0	~16	18	Specific
<b>SqueezeDet Wu et al. (2016)</b>	OID	test-dev	32.4	~9	18	Specific
<b>YOLO nano Wong et al. (2019)</b>	OID	test-dev	35.8	~12	18	Specific

Tabela 3.2: Comparação entre abordagens compactas para detecção em computadores de baixo custo (sem uso de GPU's)

longas inclinadas para o solo. Esse é apenas um exemplo de como partes do corpo humano são mais fáceis de ser confundidas em predições na detecção de objetos.

Uma outra característica que diminui a precisão do modelo em comparação aos modelos base, é a variação no dataset de imagens. Existem muitas variações nas imagens utilizadas para o treino da rede neural, as imagens não são normalizadas, isso para atender um número maior de situações de entrada. Essas variações de ambiente e posicionamento das formas alteram a precisão do modelo. Nossa estratégia para as variações no dataset foi adicionar tantas imagens de classes não predizíveis quanto aquelas utilizadas para o treino (predizíveis), para que fosse possível separar bem as classes de treino na distribuição dos dados. Essa abordagem, juntamente as otimizações de parâmetros e código-fonte, tornaram possível o aumento de 20% para 37% (um ganho de 17%) na acurácia e precisão. A Figura 3.3 apresenta alguns dos resultados obtidos na execução do modelo proposto. Vários outros exemplos também estão disponíveis no Apêndice B.



(a) Exemplo com objetos contendo oclusão  
(b) Exemplo contendo variações de tamanho  
(c) Exemplo contendo variação de iluminação

Figura 3.3: Exemplos de resultados obtidos utilizando o modelo proposto

Fonte: Do autor

### 3.0.2 Métodos alternativos

Não encontramos na literatura um método completamente similar (existem abordagens relacionadas já citadas em seção própria) para a comparação com as mesmas características que propomos: 1) Um modelo de rede neural profunda de uma passagem; 2) Um modelo compacto (tanto em arquitetura quanto em tamanho); 3) Um modelo de detecção e execução em tempo real para partes do corpo humano e gêneros; 4) Um modelo universal que pode ser executado em

qualquer dispositivo em tempo real sem alterações prévias. Porém existem métodos que podem ser utilizados para a segmentação de partes do corpo humano como uma alternativa ao proposto.

Esses métodos não detectam as partes do corpo, é necessário um processamento posterior para que seja possível a segmentação. Um exemplo seria utilizar landmarks<sup>9</sup> (He et al. (2017b); Ranjan et al. (2017)) ou então utilizar classificadores existentes como o haar cascades. O grande problema desse tipo de abordagem é a necessidade de utilizar uma grande quantidade de heurísticas para poder estimar a localização das partes. Isso gera um código-fonte muito grande e complexo.

Existem também alguns outros métodos que são um tanto distintos em implementação mas que devem ser mencionados. Esses métodos estão mais relacionados a forma de geração de datasets com partes do corpo humano do que na predição. Um desses métodos seria a utilização de dados sintéticos<sup>10</sup> para o treinamento e posterior segmentação de partes do corpo humano (Lin et al. (2019); Varol et al. (2017)). Outro exemplo interessante é a utilização da estimativa da posse para a geração de datasets de partes do corpo (Fang et al. (2018)).

De forma geral é possível utilizar os métodos alternativos já citados, porém com maior dificuldade. A utilização de *landmarks* é um exemplo viável para algumas partes, mas não para todas. Eles também são extraídos utilizando redes neurais, porém não existem um detector que entregue todos os pontos necessários para extrair tantas partes quanto as do modelo proposto. Para utilizar essa técnica é necessário realizar a leitura das coordenadas dos pontos de interesse e utilizar heurísticas para estimar a área da região que se deseja segmentar. Por exemplo, se desejar extrair a região do olho esquerdo, é necessário percorrer o vetor de pontos para esse olho e calcular o ponto  $p1(x1, y1)$  sendo o par de coordenadas onde,  $x1$  é o ponto mais à esquerda e  $y1$  o mais acima, assim como o ponto  $p2(x2, y2)$  sendo, o par de coordenadas onde  $x2$  é o ponto mais à direita e  $y2$  é o ponto mais a baixo. Esses cálculos são apenas para o sistema de coordenadas partindo da esquerda para a direita e de cima para baixo.

Ao utilizar classificadores existentes como o haar cascades o procedimento é praticamente o mesmo, o que fazemos é estimar através de heurísticas a área de certas regiões baseando-se em pontos específicos. Esse tipo de metodologia é extremamente veloz, porém possui uma alta taxa de erro em grandes variações de movimentos, qualidade da entrada e de ambiente. Como a proposta específica um modelo para ambientes não controlados ela não se adequa aos nossos requisitos.

### 3.0.3 Aplicabilidade

Uma das vantagens de utilizar um modelo universal de uma passagem é que não é necessário implementar um processamento posterior à rede, basta apenas utilizar as predições. Esse tipo de modelo também pode ser utilizado em quaisquer dispositivos sem alterações, isso

---

<sup>9</sup>São pontos de referência em partes do corpo humano.

<sup>10</sup>Imagens geradas de modo artificial por programas de computador

significa que apenas o código-fonte para carregar o modelo é alterado de acordo com a plataforma, mas o modelo em si é o mesmo.

Uma das possíveis aplicações seria na autenticação de usuários, por exemplo. Existem algumas abordagens novas de autenticação que utilizam partes do rosto de forma separada para uma maior precisão. Essas partes são segmentadas e posteriormente são enviadas a outra rede que realiza a identificação do usuário realizando a soma das previsões de cada parte, diminuindo assim sistemas fraudulentos. Em um sistema assim, cada parte reconhecida possui um peso em específico, ou podem possuir pesos similares.

Também é possível aumentar a detecção facial utilizando um modelo como o apresentado. Os métodos convencionais treinam a rede para extrair características da face inteira, e não de partes separadas. É possível representar uma face como um conjunto de partes onde todas as partes do rosto estão contidas no conjunto da face, e assim ponderar cada parte detectada extrapolando a região das partes detectadas para que seja possível obter a face por completo. A precisão poderia ser determinada pela soma dos pesos de cada parte detectada, sendo que algumas partes podem ser consideradas mais importantes do que outras.

Utilizando o mesmo método de detectar partes separadas e utilizar extração de regiões, é possível criar um detector de pedestres. Um pedestre também pode ser considerado como um conjunto de partes do corpo humano, o que torna possível sua detecção sem a necessidade da identificação de todas as partes do corpo. Esse tipo de abordagem é extremamente promissora, pois existe a possibilidade de introduzir o mesmo modelo em várias aplicações utilizando apenas heurísticas. Um dos pontos mais interessantes é que, onde algoritmos convencionais falham devido a variação de pose e ambiente, assim como na detecção facial, utilizando essa heurística de partes, o modelo proposto funcionaria perfeitamente.

Existem muitas outras possibilidades além das mencionadas. Para exemplificar e apresentar resultados visualmente comprovados é proposto um estudo de caso em duas situações distintas. A primeira é no rastreamento de pedestres, situação muito utilizada atualmente para o controle de tráfego. Uma outra situação seria na detecção de presença das mãos ao volante para identificar um condutor responsável. Essas duas situações foram postas a prova e são apresentadas nas Figuras 3.4(a), 3.4(b) e com mais detalhes no Apêndice C. É disponibilizada uma lista completa de testes em vídeo através do endereço <https://www.youtube.com/playlist?list=PLE1kmKX4LWV1fzbB4ustbO4G5bV-bW4Z>.



(a) Exemplo de detecção de pedestres

(b) Exemplo de condutor com mãos ao volante

Figura 3.4: Exemplos de resultados para mãos ao volante e detecção de pedestres

Fonte: Do autor

## 4 Considerações finais

### 4.0.1 Otimização e melhorias futuras

O modelo proposto não é perfeito, é preciso trabalhar na precisão, acurácia e no tamanho da rede para melhorar a performance. Reduzir o tamanho (parâmetros e operações) da rede, resultaria em previsões mais rápidas e eficientes. Porém é preciso determinar quais pontos da rede podem ser reduzidos sem perder os resultados já obtidos.

Uma forma viável para aumentar a precisão seria utilizando heurísticas, relacionadas ao posicionamento e agrupamento das partes do corpo humano. Utilizar esse tipo de técnica não afeta o desempenho pois elas são implementadas logicamente através do código-fonte do executor. Para que isso seja possível é preciso se basear nas propriedades espaciais das regiões do corpo humano, essas regiões são praticamente imutáveis e bem definidas. Sabemos que um olho, por exemplo, naturalmente não vai estar no lugar da boca, ou que dois olhos estarão conectados um ao outro.

Utilizar heurísticas com características espaciais pode não funcionar em muitos casos, mas para esse caso em específico elas funcionariam muito bem. É possível aplicar pesos para as partes do corpo de acordo com seus relacionamentos. Ao final somamos todas as previsões que calculamos utilizando as heurísticas e teremos uma porcentagem real de uma boca ser de fato uma boca, baseando-se nas partes vizinhas detectadas.

Para que isso se torne mais claro, imagine que é necessário aumentar a precisão na detecção da boca humana, junto com a boca foram obtidas também as previsões dos olhos e nariz. É possível inferir que, se existe um nariz e ele está posicionado logo acima da boca as chances de a boca estar no local correto aumentam. Assim também, se existem dois olhos e eles estão posicionados logo acima do nariz e opostos um ao outro, as chances de a boca estar no local correto aumentam ainda mais. A lógica utilizando heurísticas é exatamente essa, ela é bem simples, porém extremamente funcional.

Um dos problemas em utilizar heurísticas como essas é que elas são difíceis de escrever e validar. Existe também o problema da recursão, devemos utilizar essa técnica pois algumas heurísticas estão contidas em outras e isso a torna ainda mais difíceis de escrever. Mas como existe um número finito de partes do corpo humano, e ele é um número pequeno, a abordagem é usual.

Existem outros pontos que seriam interessantes se incluídos no modelo. A segmentação utilizando máscaras (Juntamente com a segmentação de máscaras, também seria viável segmentar outras regiões do corpo além daquelas já existentes) é um desses pontos. Esse tipo de segmentação entrega maiores detalhes sobre a forma do objeto em relação ao um simples retângulo, como é o caso da detecção de objetos comuns. Uma outra funcionalidade que seria interessante ao modelo, é a sua execução em ambientes web (navegadores). Na verdade essa funcionalidade já é suportada pelo ONNX, porém ainda não foi implementada no código-fonte proposto devido à algumas restrições de versão<sup>1</sup>.

#### 4.0.2 Conclusão

Foram apresentados nesse trabalho, um modelo compacto para detecção e segmentação de partes do corpo humano em ambientes não controlados, sujeitos aos mais diversos tipos de variações. A abordagem apresentada também cobre cenários em que é necessária a execução em tempo real e de forma universal na maioria dos dispositivos atuais como PC's, Smartphones e Embarcados. Foram demonstradas também, algumas aplicações reais utilizando o HPNet com o mínimo de alterações possíveis no código-fonte, e que os resultados obtidos ultrapassaram aqueles do modelo base.

De modo geral, o modelo apresentou bons resultados. Não possui a mesma precisão de um modelo completo, como aqueles utilizados para detectar objetos comuns, mas está dentro do previsto para a categoria compacta. A proposta oferece um modelo universal para a detecção de partes do corpo humano, que pode ser utilizada como base para futuras melhorias ou comparações no trabalho de outros pesquisadores. Foi abordado que no futuro, além da segmentação utilizando máscaras, a ideia seria expandir o modelo para oferecer suporte não apenas às partes do corpo, mas também a regiões. O modelo pode ser acessado através do repositório oficial disponível em <https://claudemircasa.github.io/hptiny/>.

---

<sup>1</sup>No momento em que o modelo foi compilado, ele foi gerado em uma versão ONNX anterior aquela disponibilizada atualmente.

# Referências

- Benenson, R., Popov, S. e Ferrari, V. (2019). Large-scale interactive object segmentation with human annotators. Em *CVPR*.
- Cai, Z., Fan, Q., Feris, R. S. e Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. *CoRR*, abs/1607.07155.
- Chen, H.-W. e McGurr, M. (2016). Moving human full body and body parts detection, tracking, and applications on human activity estimation, walking pattern and face recognition. Em Sadjadi, F. A. e Mahalanobis, A., editores, *Automatic Target Recognition XXVI*, volume 9844, páginas 213 – 246. International Society for Optics and Photonics, SPIE.
- Cheng, E.-J., Chou, K.-P., Rajora, S., Jin, B.-H., Tanveer, M., Lin, C.-T., Young, K.-Y., Lin, W.-C. e Prasad, M. (2019). Deep sparse representation classifier for facial recognition and detection system. *Pattern Recognition Letters*, 125:71 – 77.
- Chishti, S. O., Riaz, S., BilalZaib, M. e Nauman, M. (2018). Self-driving cars using cnn and q-learning. Em *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, páginas 1–7. IEEE.
- Chu, B., Madhavan, V., Beijbom, O., Hoffman, J. e Darrell, T. (2016). Best practices for fine-tuning visual classifiers to new domains. Em *European conference on computer vision*, páginas 435–442. Springer.
- Dai, J., Li, Y., He, K. e Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. Em *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, páginas 379–387, USA. Curran Associates Inc.
- Dalal, N. e Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. Em Schmid, C., Soatto, S. e Tomasi, C., editores, *International Conference on Computer Vision & Pattern Recognition (CVPR ’05)*, volume 1, páginas 886–893, San Diego, United States. IEEE Computer Society.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. e Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Em *CVPR09*.
- Derakhshani, M. M., Masoudnia, S., Shaker, A. H., Mersa, O., Sadeghi, M. A., Rastegari, M. e Araabi, B. N. (2019). Assisted excitation of activations: A learning technique to improve

- object detectors. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Druzhkov, P. N. e Kustikova, V. D. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15.
- Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J. e Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.
- Fang, H.-S., Lu, G., Fang, X., Xie, J., Tai, Y.-W. e Lu, C. (2018). Weakly and semi supervised human body part parsing via pose-guided knowledge transfer. *arXiv preprint arXiv:1805.04310*.
- Feng, L., Kudva, P., Da Silva, D. e Hu, J. (2018). Exploring serverless computing for neural network training. Em *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, páginas 334–341.
- Girshick, R. (2015). Fast r-cnn. Em *The IEEE International Conference on Computer Vision (ICCV)*.
- Girshick, R. B., Donahue, J., Darrell, T. e Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.
- Hampel, D. e Winder, R. O. (1971). Threshold logic. *IEEE Spectrum*, 8(5):32–39.
- He, K., Gkioxari, G., Dollár, P. e Girshick, R. B. (2017a). Mask R-CNN. *CoRR*, abs/1703.06870.
- He, Z., Kan, M., Zhang, J., Chen, X. e Shan, S. (2017b). A fully end-to-end cascaded cnn for facial landmark detection. Em *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, páginas 200–207. IEEE.
- Hori, T., Watanabe, S., Zhang, Y. e Chan, W. (2017). Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. *arXiv preprint arXiv:1706.02737*.
- Ivakhnenko, A. G. e Lapa, V. G. (1967). Cybernetics and forecasting techniques. *CERN Document Server*.
- Jalal, A., Nadeem, A. e Bobasu, S. (2019). Human body parts estimation and detection for physical sports movements. Em *2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE)*, páginas 104–109.
- Kato, S., Takeuchi, E., Ishiguro, Y., Ninomiya, Y., Takeda, K. e Hamada, T. (2015). An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68.

- Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Malloci, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D. e Murphy, K. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from <https://storage.googleapis.com/openimages/web/index.html>.*
- Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T. e Ferrari, V. (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.
- Li, Z., Sun, W. e Wang, L. (2012). A neural network based distributed intrusion detection system on cloud platform. Em *2012 IEEE 2nd international conference on Cloud Computing and Intelligence Systems*, volume 1, páginas 75–79. IEEE.
- Lin, K., Wang, L., Luo, K., Chen, Y., Liu, Z. e Sun, M.-T. (2019). Cross-domain complementary learning with synthetic data for multi-person part segmentation. *arXiv preprint arXiv:1907.05193*.
- Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P. e Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.
- Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B. e Belongie, S. (2017). Feature pyramid networks for object detection. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. e Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11 – 26.
- McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- Nikolić, P. K., Yang, H., Chen, J. e Stankevich, G. P. (2018). Syntropic counterpoints: Art of ai sense or machine made context art. Em *ACM SIGGRAPH 2018 Posters*, SIGGRAPH ’18, páginas 18:1–18:2, New York, NY, USA. ACM.
- Oñoro-Rubio, D. e López-Sastre, R. J. (2016). Towards perspective-free object counting with deep learning. Em Leibe, B., Matas, J., Sebe, N. e Welling, M., editores, *Computer Vision – ECCV 2016*, páginas 615–629, Cham. Springer International Publishing.

- Pont-Tuset, J., Uijlings, J., Changpinyo, S., Soricut, R. e Ferrari, V. (2020a). Connecting vision and language with localized narratives. Em *ECCV*.
- Pont-Tuset, J., Uijlings, J., Changpinyo, S., Soricut, R. e Ferrari, V. (2020b). Connecting vision and language with localized narratives. Em *ECCV*.
- Ranjan, R., Patel, V. M. e Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):121–135.
- Redmon, J., Divvala, S., Girshick, R. e Farhadi, A. (2016). You only look once: Unified, real-time object detection. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Redmon, J. e Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.
- Ren, S., He, K., Girshick, R. e Sun, J. (2015a). Faster r-cnn: Towards real-time object detection with region proposal networks. Em Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. e Garnett, R., editores, *Advances in Neural Information Processing Systems 28*, páginas 91–99. Curran Associates, Inc.
- Ren, S., He, K., Girshick, R. B. e Sun, J. (2015b). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.
- Samangouei, P., Patel, V. M. e Chellappa, R. (2017). Facial attributes for active authentication on mobile devices. *Image and Vision Computing*, 58:181 – 192.
- Teerapittayanon, S., McDanel, B. e Kung, H.-T. (2017). Distributed deep neural networks over the cloud, the edge and end devices. Em *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, páginas 328–339. IEEE.
- Tian, Y., Luo, P., Wang, X. e Tang, X. (2015). Deep learning strong parts for pedestrian detection. Em *The IEEE International Conference on Computer Vision (ICCV)*.
- Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I. e Schmid, C. (2017). Learning from synthetic humans. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 109–117.
- Vemuri, V. K. (2019). Pattern recognition and machine learning.
- Wilson, P. I. e Fernandez, J. (2006). Facial feature detection using haar classifiers. *Journal of Computing Sciences in Colleges*, 21(4):127–133.
- Wong, A., Famouri, M., Shafiee, M. J., Li, F., Chwyl, B. e Chung, J. (2019). YOLO nano: a highly compact you only look once convolutional neural network for object detection. *CoRR*, abs/1910.01271.

- Wu, B., Iandola, F., Jin, P. H. e Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Wu, B., Iandola, F. N., Jin, P. H. e Keutzer, K. (2016). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *CoRR*, abs/1612.01051.
- Yang, S., Luo, P., Loy, C.-C. e Tang, X. (2015). From facial parts responses to face detection: A deep learning approach. Em *The IEEE International Conference on Computer Vision (ICCV)*.
- Yang, W., Ouyang, W., Li, H. e Wang, X. (2016). End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Zhang, Q.-s. e Zhu, S.-c. (2018). Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39.
- Zhang, Y., Gueguen, L., Zharkov, I., Zhang, P., Seifert, K. e Kadlec, B. (2017). Uber-text: A large-scale dataset for optical character recognition from street-level imagery. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

# Apêndice A: Exemplos de imagens obtidas do Google Open Images Dataset

Aqui são apresentadas algumas das imagens do *Dataset* público do Google selecionadas manualmente. Essas imagens foram selecionadas com a intenção de apresentar a diversidade contida no banco de imagens e suas particularidades.

## A.1 Imagens contendo oclusões

Nesse tipo de situações é possível encontrar objetos que cobrem outros objetos. Para a detecção, isso a torna mais difícil, pois os descritores (características) do objeto não estão todos visíveis. Para a rede neural, objetos com muita oclusão são difíceis de interpretar, e se o banco imagens conter apenas amostras oclusas para uma determinada classe, isso afetaria diretamente a predição quando a entrada for completa. Para as amostras seguintes, é utilizada a classe *Person* como base.

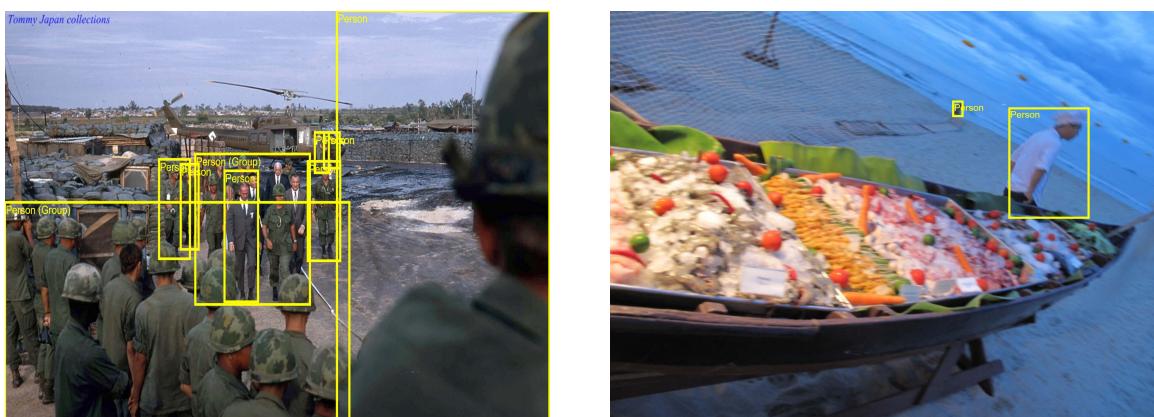


Figura A.1: Exemplos de imagens contendo oclusões

Fonte: Google



Figura A.2: Exemplos de imagens contendo oclusões

Fonte: Google

## A.2 Variações de iluminação

A iluminação também é um fator de influência direta no treino e predição de redes neurais. Objetos em ambientes pouco iluminados não possuem suas formas bem definidas, dificultando sua identificação pela rede. O ideal é escrever a rede para conseguir lidar com esse tipo de variação, porém isso pode aumentar significativamente o seu tamanho. Podem existir também casos em que não há variações de iluminação na entrada, fazendo com que não seja necessária a adaptação da rede.

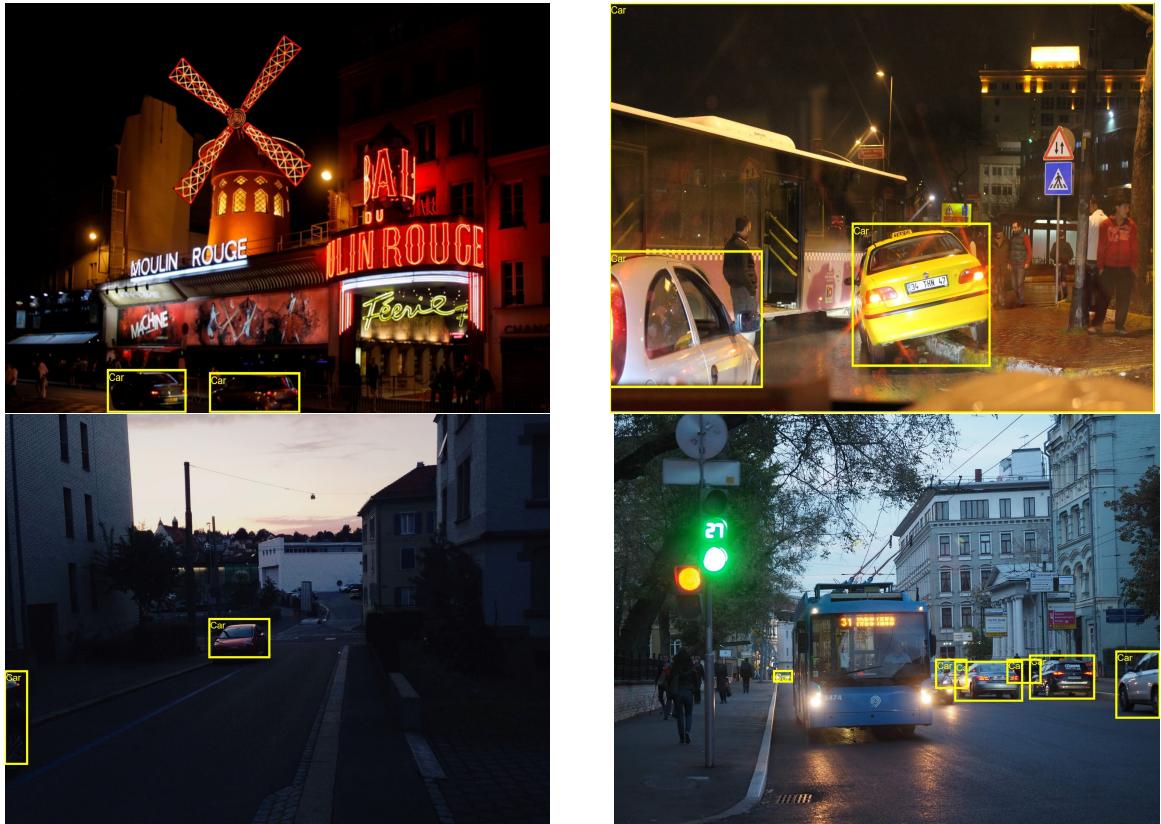
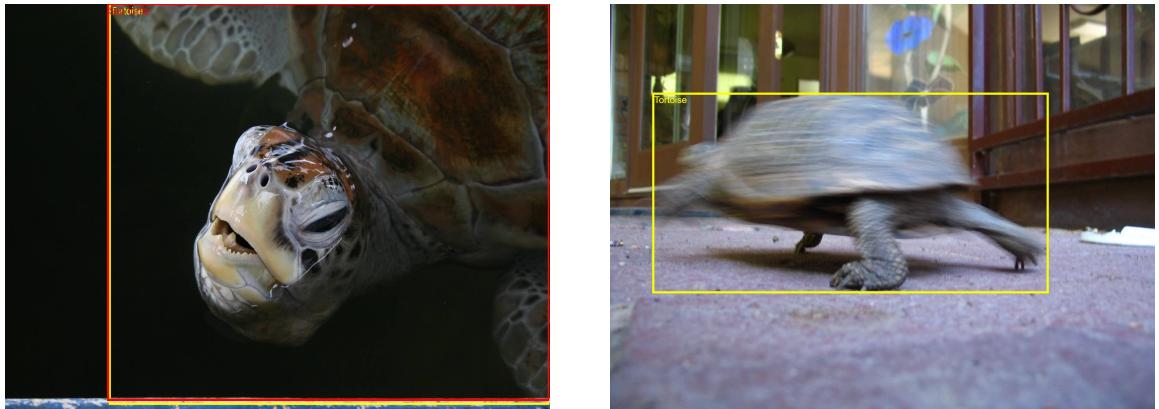


Figura A.3: Exemplos de imagens contendo variações de iluminação

Fonte: Google

### A.3 Outros tipos de variações

Existem também outros tipos de variações que possuem grande influência no treino e predição, exemplos são as de tamanho dos objetos, objetos borrados ou com recorte. Assim como as variações citadas anteriormente, essas acabam causando os mesmos problemas.



(a) Exemplo de corte

(b) Exemplo de borão

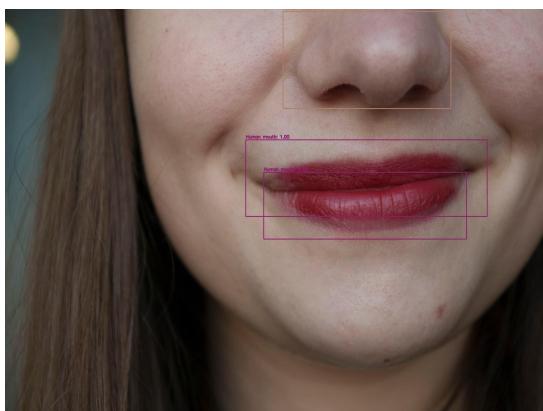
Figura A.4: Exemplos de imagens contendo outros tipos de variações

Fonte: Google

# Apêndice B: Exemplos de resultados obtidos utilizando o HPNet

## B.1 Exemplos de previsões

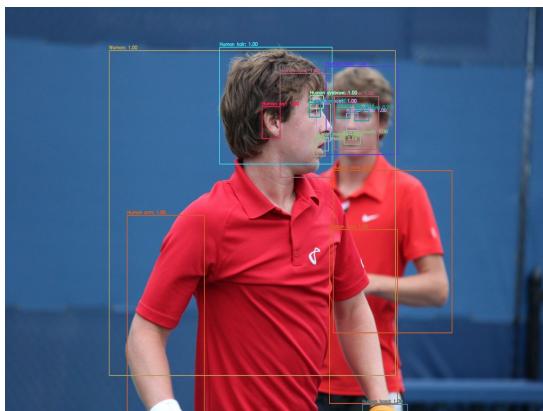
Esta seção apresenta alguns exemplos de previsões obtidas utilizando o HPNet. É possível observar visualmente a precisão do modelo, porém, ele não é perfeito. Existem alguns casos de comportamento inadequado, como na Figura B.1(c) em que é aparece a figura de um homem, porém a previsão o caracteriza como mulher. Esse caso em particular pode acontecer devido a classificação de gênero, pois também existem mulheres com cabelos curtos e com traços masculinos. Os Gráficos B.3, B.4 e B.5 apresentam, respectivamente, o mAP, resultados de detecção e de ground-truth por classe para o *dataset* de validação.



(a) Face cortada



(b) Grupo de garotos



(c) Jovens praticando esporte



(d) Grupo de pessoas

Figura B.1: Resultados de previsões obtidas com o HPNet

Fonte: Do autor

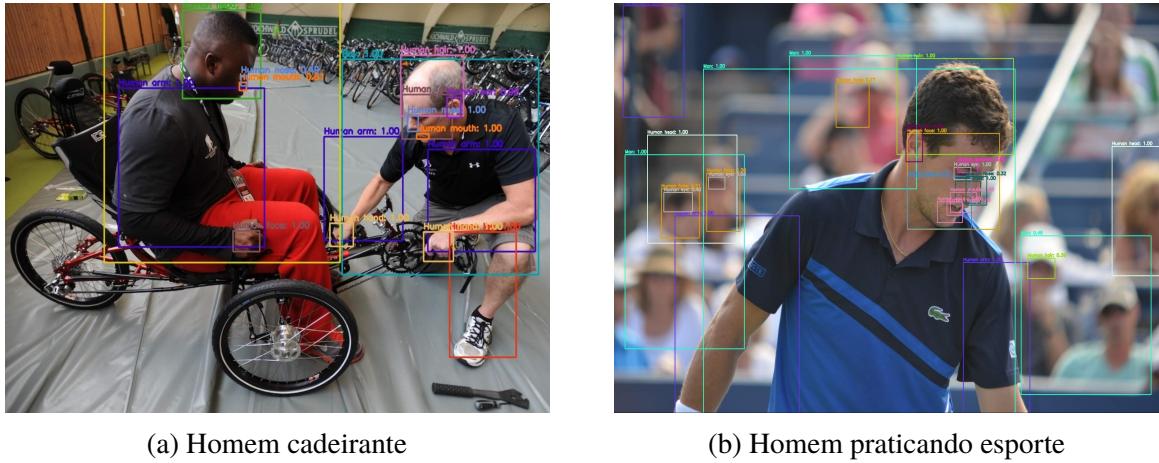


Figura B.2: Resultados de predições obtidas com o HPNet

Fonte: Do autor

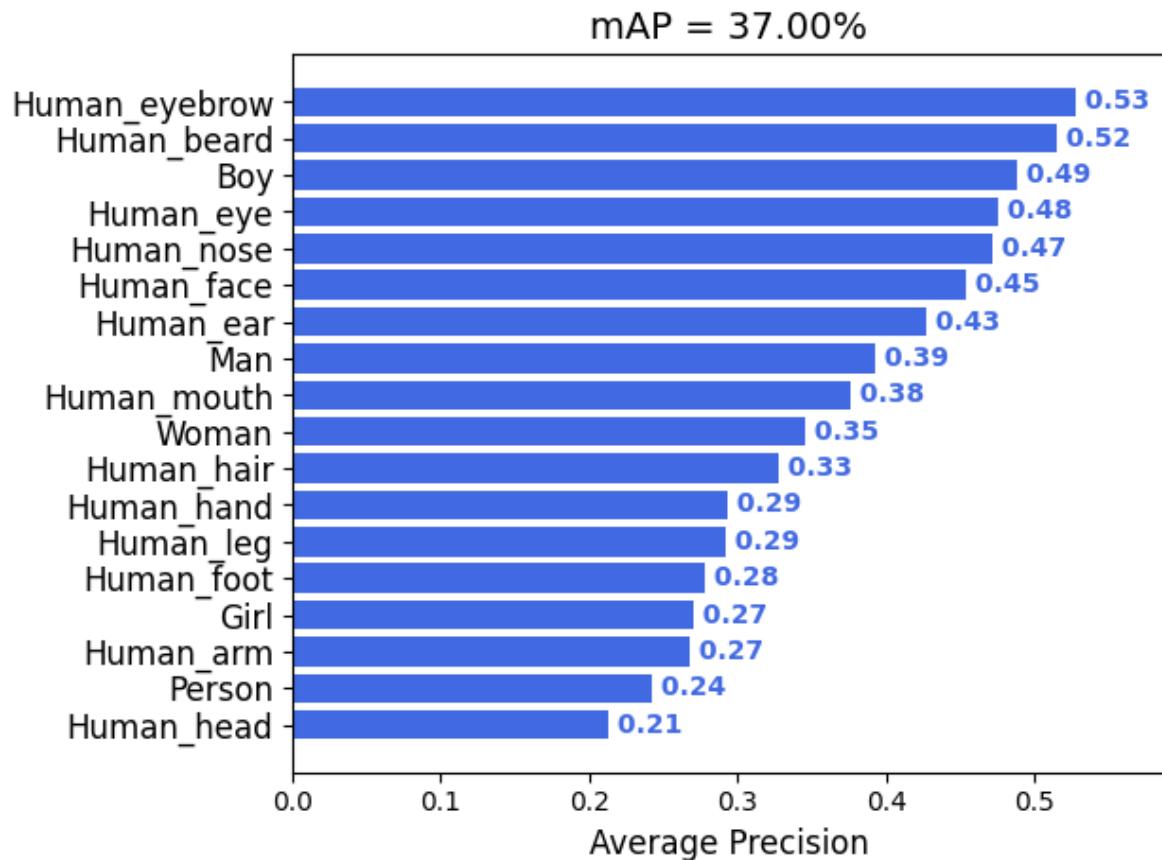


Figura B.3: Gráfico de resultados por classe para o mAP

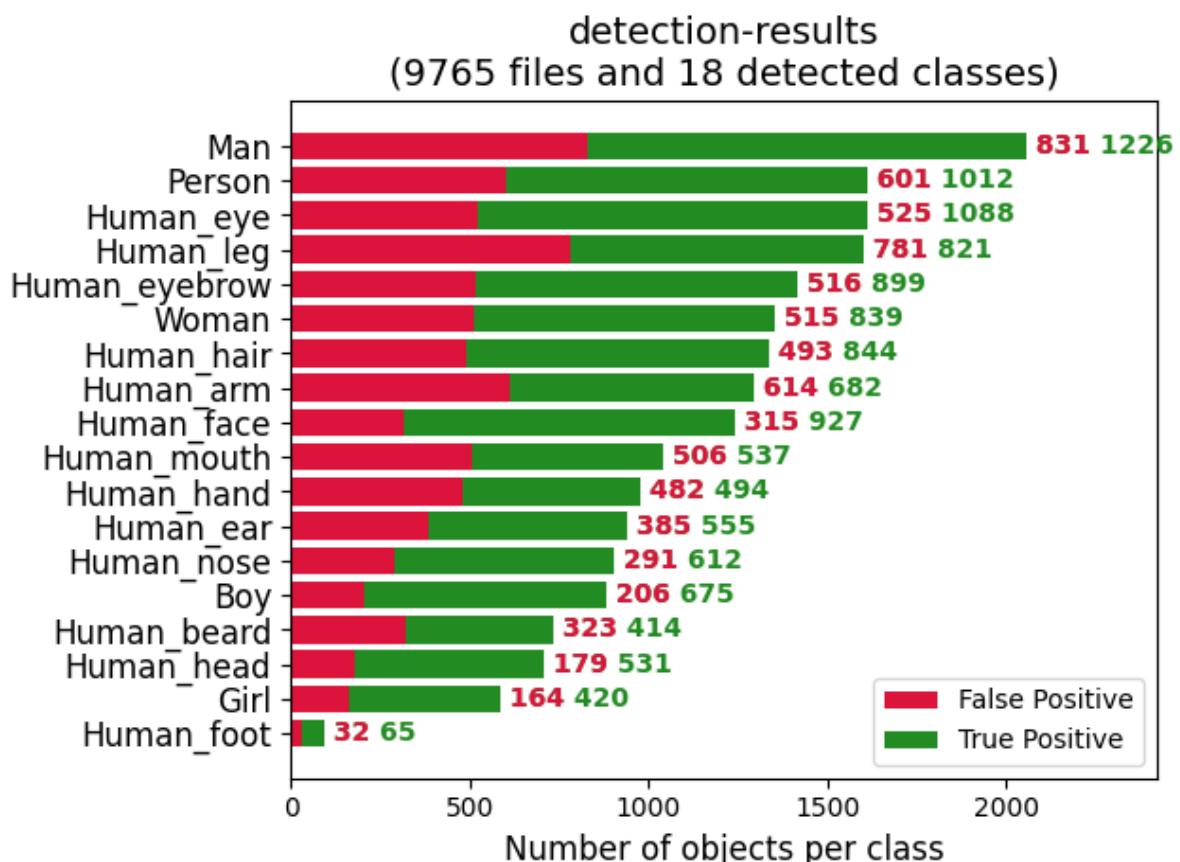


Figura B.4: Gráfico de resultados de detecção por classe

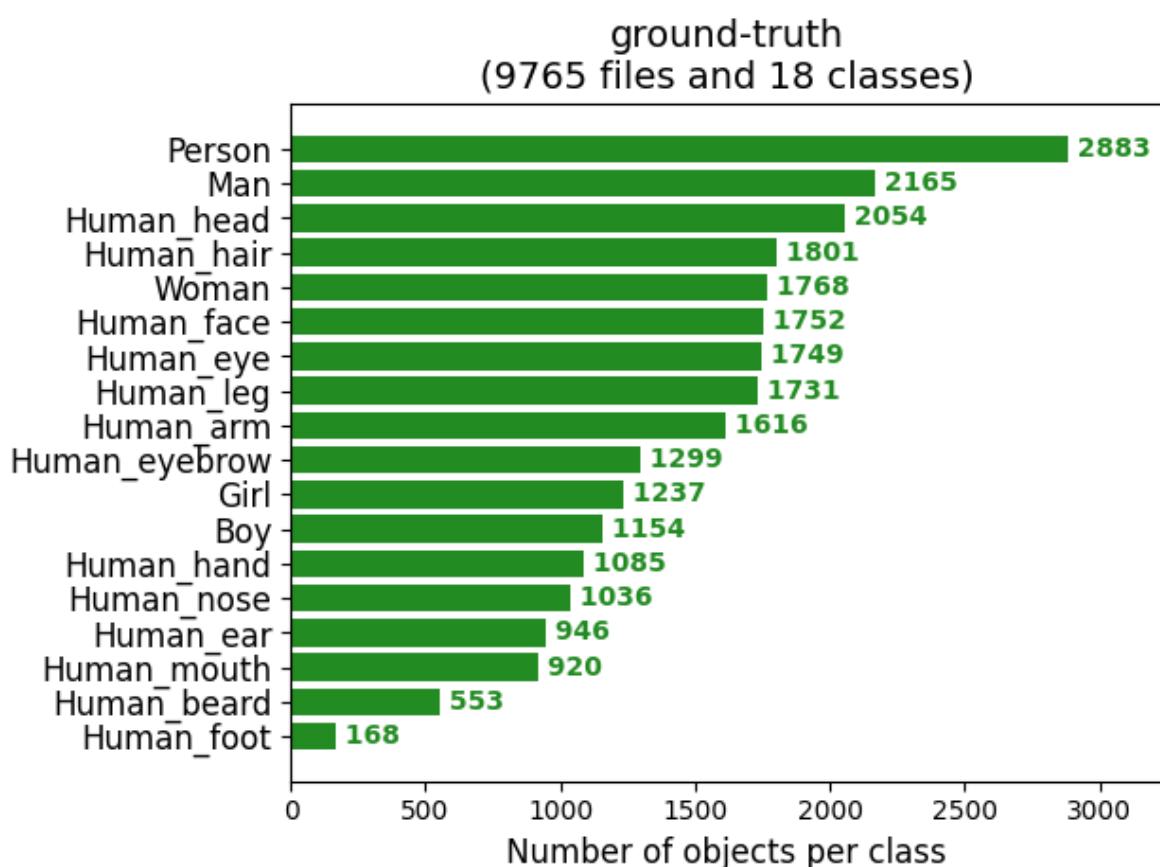


Figura B.5: Gráfico de ground-truth por classe

# Apêndice C: Estudo de caso

## C.1 Detecção de pedestres e condução consciente

Foram colocadas a prova duas situações com a premissa de utilizar o mesmo modelo sem alterações e com o mínimo de alterações no código-fonte do executor. Na primeira situação foi necessário utilizar o HPNet para realizar a detecção de pedestres, função semelhante aquela encontrada em carros autônomos. A detecção de pedestres foi executada sem problemas, como apresentada nas Figuras C.1 e C.2. Bastou apenas uma pequena alteração no código-fonte para apresentar um retângulo vermelho de alerta quando pedestres fossem detectados, o restante foi configurado pela própria linha de comando através de parâmetros.

O mesmo é válido para o caso da detecção de mãos ao volante. Bastou alterar apenas um trecho do código-fonte para apresentar o retângulo vermelho quando as mãos estão fora do volante e realizar uma configuração manual para que a imagem de entrada da rede seja apenas a região em que se encontra o motorista. Os demais parâmetros são passados através da linha de comando. A Figura C.3 apresenta os exemplos para a detecção de mãos ao volante.



Figura C.1: Resultados de detecção de pedestres obtidos com o HPNet

Fonte: Do autor



Figura C.2: Resultados de detecção de pedestres obtidos com o HPNet

Fonte: Do autor



Figura C.3: Resultados de detecção de mãos ao volante obtidos com o HPNet

Fonte: Do autor