Contagem do Fluxo de Pessoas Utilizando Aprendizado Profundo

Pedro Henrique de Moraes

Contagem do Fluxo de Pessoas Utilizando Aprendizado Profundo

Pedro Henrique de Moraes

Orientador: *Prof. Dr. Edson Takashi Matsubara*

Dissertação apresentada ao programa de Pós Graduação, Mestrado Acadêmico em Ciência da Computação, da Universidade Federal de Mato Grosso do Sul, para a defesa do Mestrado.

UFMS - Campo Grande

Agosto/2021

Aos meus pais,
Claudia e Sandro,

A minha irmã,
Helena,

Aos meus tios,
Vanessa e Fabrício.
A minha namorada.

Agradecimentos

Primeiramente, ao meu orientador o professor Edson Takashi, que me ensinou não só a pesquisa, mas também sobre lições de vida; as professoras Patrícia Matsubara e Jane Eleutério que sempre me atenderam com dedicação desde a graduação e assim compartilharam seus conhecimentos.
Aos meus pais Sandro e Claudia, por todo o apoio. Minha irmã Helena, menina de luz.
Agradeço também ao Fabrício, Vanessa, Gabrielly e demais membros da família pela compreensão, companheirismo, apoio e amor.
Aos professores do programa e de outras instituições que disponibilizaram tempo e conhecimento para me ensinar.
Aos amigos do laboratório da pós graduação: André, Lucas, Cleiton, Rogério, Adriano, Fabrício.
Ao grupo do LIA, especialmente por me proporcionar amigos como o Carlos.
Aos funcionários e servidores da UFMS e FACOM.
Ao Saul, Barbueno, Erik, Élder e Fábio pelo apoio nas coletas de dados e configuração da rede na FACOM.
À Before TI pelo apoio financeiro e por acreditar no projeto, principalmente na figura do Marcelo.
Ao Hudson e o Guilherme, pelas caronas durante os dias de chuva e noites da FACOM até em casa.
Aos demais amigos pela compreensão e paciência.

Ninguém ignora tudo.
Ninguém sabe tudo.
Todos nós sabemos alguma coisa.
Todos nós ignoramos alguma coisa.
Por isso aprendemos sempre
(Paulo Freire)

Abstract

The customer's flow count is the counting of the number of people entering the stores. This count allows you to extract different business metrics such as conversion rate of marketing actions, time of visit and people's traffic. The main objective of this dissertation is to propose, develop and evaluate a solution to count customers using security cameras. The proposal is to combine deep learning algorithms for counting people. Additionally, identify people who should not be counted, such as employees and collaborators. Videos were collected in two different places that were manually written, the Real count was defined. They were later submitted to Yolov5 to define the count by People and RetinaFace by Face. The counting performed by the proposal was compared with the manual counting using a significance test. According to the test, there was no significant difference between the Pessoa and Real scores. Thus, the technique of the proposed solution was validated and the economic feasibility presented a cost of 24.4USD per month, considering 10 hours of video per day for cloud processing. The proposed solution does not require the insertion of specific hardware and modifications in the store's spaces, being a promising alternative for this problem.

Resumo

A contagem do fluxo de clientes é a contabilização da quantidade de pessoas que entram no estabelecimento. Esta contagem permite extrair diferentes métricas do negócio como taxa de conversão das ações de marketing, duração das visitas e tráfego de pessoas. O objetivo principal desta dissertação é propor, desenvolver e avaliar uma solução para contar clientes utilizando câmeras de segurança. A proposta consiste em combinar algoritmos de aprendizado profundo para contagem das pessoas. Adicionalmente identificar pessoas que não devem entrar na contagem, como funcionários e colaboradores. A partir de vídeos coletados em dois locais distintos que foram anotados manualmente foi definido a contagem Real. Posteriormente foram submetidos à Yolov5 para a definição da contagem por Pessoas e RetinaFace por Face. A contagem realizada pela proposta foi comparada com a contagem manual utilizando teste de significância. Segundo o teste houve não diferença significativa entre as contagens de Pessoa e Real. Sendo assim a técnica da solução proposta foi validada e a viabilidade econômica apresentou custo de 24,4USD por mês, considerando 10 horas de vídeo diárias para processamento em nuvem. A solução proposta não necessita de inserção de hardware específico e modificações nos espaços dos lojistas, sendo uma alternativa promissora para esse o problema.

Sumário

Lista de Figuras

CAPÍTULO

1

Introdução

O fluxo de clientes pode ser definido como o número de clientes que passam por um caminho com um determinado comprimento através de uma área do varejo (Shumsky et al., 2021). Na proposta desse trabalho é definido pelo número de pessoas que atravessam uma linha ou região em direção a um estabelecimento. Serão contabilizados os clientes de acordo com sua orientação de movimento e o destino final dado pela última aparição desse objeto. A quantidade de clientes, quantos desses clientes são a mesma pessoa, podem ser utilizados para verificar se, por exemplo, os esforços de propaganda tiveram impacto nas vendas (Tretyak and Sloev, 2013). Em shoppings é possível usar essa métrica para definir preços de aluguéis de lojas em determinados corredores ou em tempos de pandemia avaliar o impacto sofrido pelo lojista em seu fluxo de clientes (Shumsky et al., 2021). Dessa forma, isso pode auxiliar os vendedores a adaptarem suas vitrines de forma dinâmica, modificando sua vitrine de maneiras distintas e analisar o impacto dessas mudanças com a quantidade de conversão de clientes que entram em suas lojas. Em um Shopping Center se em um corredor qualquer, através das contagens dos clientes nas lojas, foi identificado que estas possuem quantidades maiores de conversão terão seus aluguéis ajustados de acordo com essa informação.

Informações sobre o fluxo podem ser obtidas através de aparelhos como câ-

meras 3D, sensores e analisadores de redes Wi-Fi. Porém a aquisição destes aparelhos pode ser uma solução financeiramente custosa, sendo em alguns casos inviável a pequenos varejistas. Um investimento muito comum no Brasil é a implantação de câmeras de segurança, para identificar roubos, comportamentos indesejados e prevenção de perdas. É uma pratica crescente e câmeras IPs, por exemplo, tem se proliferado bastante nesse segmento (Gantz and Reinsel, 2012). Essas câmeras provêm imagens em tempo real, com armazenamento local ou em serviços de nuvem. Com o avanço das tecnologias de processamento de imagens, é possível dar novos significados à utilização dessas câmeras. Uma delas pode ser o processamento dessas imagens por algoritmos de Aprendizado de Máquina, como, detecção de pessoas e reconhecimento delas. Dessa maneira o equipamento disponível do lojista pode ser utilizado para outros propósitos sem perder sua ideia original de fornecer segurança e necessidade de aquisição de outros equipamentos. A proposta deste trabalho está alinhada com a ideia de aproveitar a implantação ou já existência de câmeras de segurança e agregar uma nova e importante funcionalidade de contagem de fluxo de pessoas.

O problema aparentemente trivial que consiste na combinação de algoritmos de detecção de objetos e rastreamento, ao ser confrontado com cenários reais de câmeras de segurança em lojas brasileiras, diversos são os desafios que a literatura ainda não explora adequadamente como: baixa qualidade das imagens, os ângulos e alturas das câmeras de segurança inadequados para contagem, taxas de FPS computacionalmente intensivas para processamento e resoluções inadequadas para identificação por face. O comportamento humano é outro ponto que deve ser levado em consideração, pois é cada vez mais comum encontrar pessoas caminhando e olhando para o celular simultaneamente, o que dificulta a visualização da face das pessoas por câmeras de segurança. As limitações foram analisadas nessa proposta através do desenvolvimento e testes com algoritmos de detecção e classificação de objetos.

1.1 Objetivo

O objetivo dessa dissertação consiste em desenvolver e avaliar o desempenho de algoritmos de Visão Computacional e Redes Neurais Convolucionais para detectar o cliente, rastreá-lo, fornecer métricas de estimativa do fluxo de clientes

e por último distinguir se este é um funcionário do estabelecimento. Com isso foram estipuladas duas hipóteses:

1. (Viabilidade Técnica) É possível estimar fluxo de pessoas em estabelecimentos com a utilização de câmeras de segurança a partir de algoritmos de visão computacional e aprendizado de máquina?

2. (Viabilidade Econômica) Caso seja possível estimar o fluxo de pessoas com base na hipótese anterior, seria possível realizar tal tarefa com baixo custo para lojistas ao levar em conta as soluções existentes e mantendo uma contagem satisfatória de clientes?

1.2 Impactos Esperados

Para a comunidade científica são esperados impactos pela avaliação experimental criteriosa de algoritmos já existentes e também suas combinações que possibilitem obter melhores taxas de assertividade e menor custo computacional. Principalmente o impacto do reconhecimento facial ao serem feitas detecções em menor qualidade com extração de características do quadro original.

Para a sociedade é esperado uma solução de baixo custo que faça proveito da infraestrutura de câmeras de segurança já existentes a fim de auxiliar a tomada de decisões dos lojistas com impacto no crescimento de seus negócios bem como um melhor direcionamento de produtos para os clientes.

A dissertação está descrita em 5 capítulos. No segundo capítulo, são descritos os fundamentos teóricos para a realização deste trabalho. No terceiro capítulo estão os trabalhos da literatura que possuem relação com o tema proposto. A proposta está detalhada no quarto capítulo. Por fim, no quinto capítulo, estão disponíveis as considerações finais contendo os pontos fracos, dificuldades, as contribuições e possíveis trabalhos para pesquisas futuras.

CAPÍTULO

2

Referencial Teórico

Neste capítulo são apresentados conceitos que tornaram seu entendimento necessário para o desenvolvimento da pesquisa.

Na Seção 2.1 serão apresentados conceitos referentes a área de visão computacional, dos quais auxiliam no processamento bem como a interpretação computacional das imagens de maneira semelhante a visão dos seres humanos. Na Seção 2.2 é introduzida a área da computação de Aprendizado de Máquina, área esta que está relacionada com o treinamento de algoritmos para fins específicos ou gerais. Na Seção 2.3 discutido o conceito de aprendizado profundo, desde a sua invenção, transição de raso para profundo, redes neurais, funções matemáticas relacionadas a área, exemplos de aplicações e Redes Neurais convolucionais com suas especificidades. Na Seção 2.4 é introduzido o rastreamento de objetos através de imagens, a ideia por trás, suas utilizações, bem como o estado da arte nessa área.

2.1 Visão Computacional

Visão computacional tem como objetivo extrair informações de imagens através de computadores, imitando a visão humana (Prince, 2012). Separar objetos específicos dos demais, contar pessoas, segmentar partes específicas de imagens

e até mesmo reconstruir faces ou objetos 3D são exemplos de problemas que podem ser resolvidos através do uso da Visão Computacional. Porém quando máquinas processam imagens, diferente da visão humana, são mais suscetíveis a erros por influência de limitações tecnológicas na coleta de imagens que podem vir com baixa resolução, tamanhos diferentes e deformações mas também a capacidade de generalização que ainda não são comparáveis ao ser humano. Para o computador os vídeos são representados por conjuntos de imagens em sequência, que podem ser de câmeras fotográficas, câmeras de segurança, internet ou celulares.

Desde a obtenção dos dados até o treinamento de algoritmos, existem diversas tarefas que são comumente realizadas e que compõe uma espécie de *pipeline* ilustrado na Figura 2.1. É composto por quatro etapas (O'Mahony et al., 2019; Szeliski, 2010):

- obtenção dos dados;
- pré-processamento;
- extração de características;
- aprendizado de máquina;

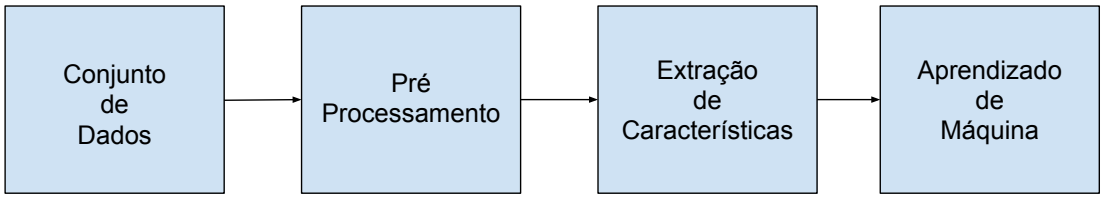


Figura 2.1: Pipeline de visão computacional

Obtenção dos Dados

A obtenção dos dados pode ser realizada por meio de câmeras, drones ou imagens obtidas pela da internet. Conjuntos de dados como LFW (Huang et al., 2007), CIFAR (Krizhevsky et al., 2009) são exemplos de conjuntos de dados compostos por imagens retiradas e pré processadas da internet. Estes conjuntos são

utilizados para facilitar a comparação de novas propostas de arquitetura e metodologias de processamento de imagens. Um exemplo importante de conjunto de dados é ImageNet que possui mais de 14 milhões de imagens, divididas em aproximadamente 22 mil sub categorias das quais podem estar divididas em até 27 categorias mais específicas (Russakovsky et al., 2015a). Alex Krizhevesky utilizou destes dados para o treinamento e avaliação do desempenho, do que seria a primeira rede neural convolucional a obter performance de destaque frente as abordagens tradicionais no desafio ILSVRC (Russakovsky et al., 2015b).

Uma imagem pode ter um objeto de interesse oculto, ter um ambiente com iluminação insuficiente, ter um objeto com muita luz ou reflexo. A visão humana muitas vezes é eficaz contra esses problemas. Para as máquinas, estes problemas podem dificultar o processamento. Para isso, a escolha de parâmetros como tipo, abertura e velocidade das lentes podem fazer diferença na captura de imagens. Também pode fazer parte do processo a escolha desses parâmetros.

Um bom conjunto de dados pode refletir em um bom desempenho do modelo. Pela Lei dos Grandes Números uma amostra grande tem maior probabilidade de refletir o todo. Dessa forma o erro de Generalização, dado pela distância dos erros do treino e teste, tende a ser menor (Vapnik, 1998).

Pré-Processamento e Extração de Características

O pré-processamento padroniza as imagens ou elimina imagens ruidosas (Krig, 2016). A falta de brilho é um exemplo de ruído pode tornar objetos irreconhecíveis e pode ser resolvido pelo pré-processamento das imagens. Várias técnicas podem ser aplicadas no pré-processamento, por exemplo, transformações pixel a pixel, caracterização de pontos de interesse em imagens, descrições visuais e redução de dimensionalidade (Prince, 2012). Cada técnica produz resultados distintos e são aplicados de acordo com a necessidade do problema.

Imagens podem ser representadas por várias estruturas de dados computacionalmente. A estrutura de matriz é comum em bibliotecas como OpenCV em que cada posição equivale a um pixel da imagem (Bradski and Kaehler, 2008). Transformações pixel a pixel modificam o valor de cada pixel da imagem por meio de soma ou aplicações de funções complexas. O branqueamento é, um exemplo de aplicação, obtido pela subtração da média dos pixeis divididos pelo desvio padrão para solucionar problemas como reflexão ou intensidade extrema da luz. Outro

exemplo de tratamento, é o efeito de borrar a imagem, obtido pela convolução da imagem com uma Gaussiana 2D. Após uma filtragem nos dados, é possível usa-los como entrada para extratores de características (Guyon and Elisseeff, 2006).

Filtros de Gabor (Jain and Farrokhnia, 1990) são exemplos de extratores de características que utilizam Gaussiana 2D porém com auxílio de uma senoide. Estes filtros possibilitam detecção de bordas em imagens a partir do produto entre o mapa de frequência obtido pela gaussiana e o mapa de orientação resultante da senóide, um exemplo pode ser observado na Figura 2.2. A configuração do ângulo da orientação da senoide reflete em diferentes resultados. Uma configuração de 0 graus pode ressaltar melhor traços na vertical, com 90 graus pode ressaltar melhor objetos com traços verticais, podendo variar de acordo com a escolha do ângulo do filtro. Mapa de textura, filtros de Haar ou Laplacianos, são exemplos de outras técnicas que atuam pixel a pixel (Prince, 2012).

Figura 2.2: Aplicação do filtro de Gabor

Por outro lado operações como a de Harris (SI and Stephens, 1988) fazem a caracterização de bordas, cantos e regiões planas através da análise de transição dos valores dos pixels. Se a transição ocorre em ambas as direções, o ponto de interesse é um canto, se ocorrer para ambos os lados é considerado borda e por último se a variação ocorrer para todos os lados é uma região plana. Dessa maneira o intuito é encontrar cantos que definem um objeto e podem ser observados na Figura 2.3. Na figura a imagem foi processada iniciando com a conversão de RGB para escala de cinza e então aplicando um kernel de dimensões 21x21 parametrizado na imagem original, a fim de obter o resultado filtrado por esse kernel.

A partir da coleta, o pré-processamento e a extração de características os

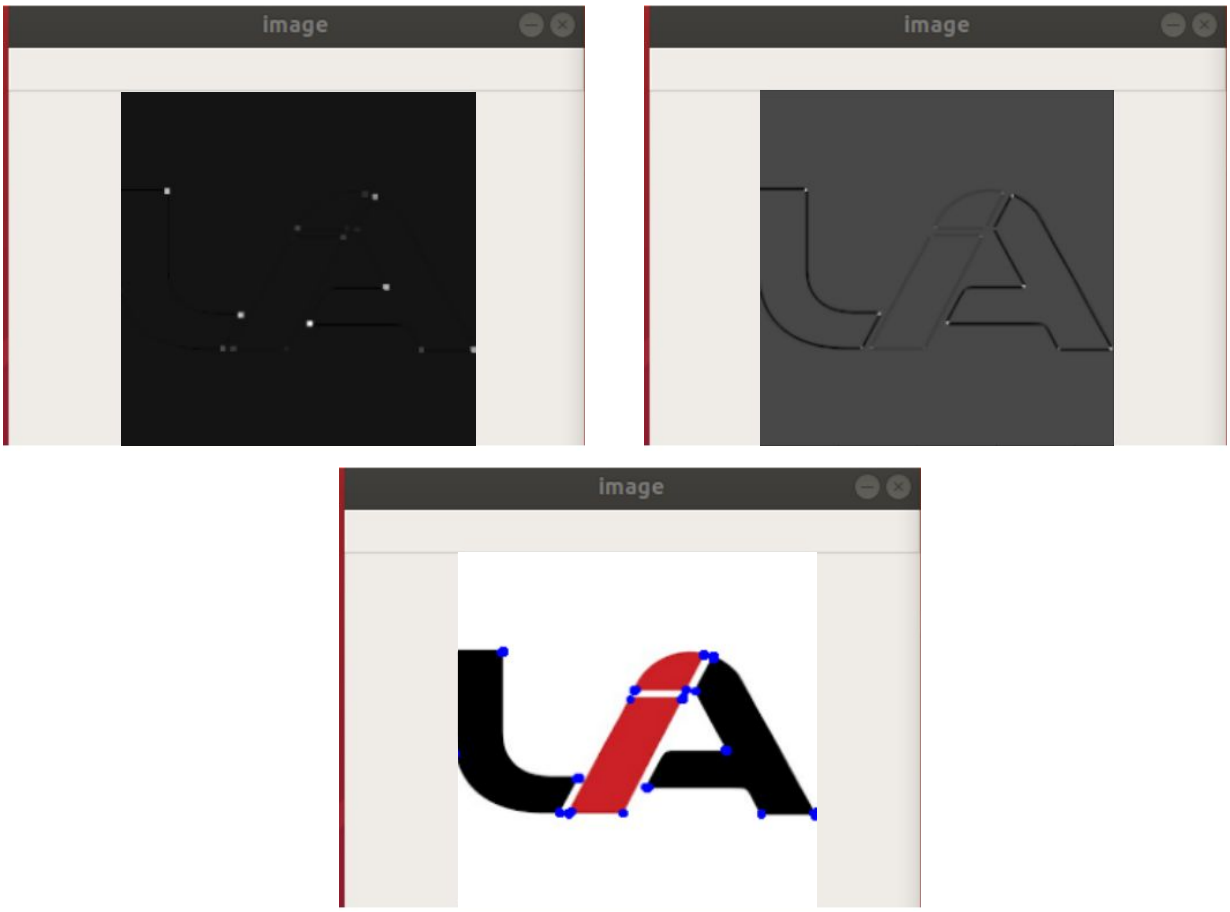


Figura 2.3: Uso do operador Harris para detecção de bordas

dados podem servir de entrada para um modelo de aprendizado de máquina, que processará os padrões e características extraídas transformando esses dados em informações que podem descrever formas, objetos e situações.

Aprendizado de Máquina

Aprendizado de máquina é o estudo de algoritmos que melhoram automaticamente com a experiência (Mitchell et al., 1997). Em aprendizado de máquina, em referência ao próprio nome, são fornecidos dados para que estes funcionem como experiência e consequentemente venham a gerar conhecimento.

A estatística compõe uma das bases do aprendizado de máquina, ambas se misturam para prover a análise de dados (Witten and Frank, 2002). Enquanto a estatística ampara a parte teórica com foco em modelos e seus ajustes, o aprendizado de máquina busca dar poder de decisão com base nesses modelos implementados por algoritmos. Desta maneira, semelhante a capacidade humana,

esses algoritmos são capazes de reconhecer padrões e podem tomar decisões em determinados contextos.

Para desenvolver uma máquina capaz de aprender, é necessário fornecer situações a fim de que os modelos estatísticos a partir do indutor treinado tenha um classificador como resultado. Um algoritmo pode aprender com um determinado conjunto de dados, para isso é necessário que possa treinar a partir desses dados.

Seres humanos conseguem generalizar com poucos exemplos, as máquinas precisam de uma maior quantidade de dados. Os modelos possuem várias estratégias de aprendizado como humanos, a dedução, a analogia ou até mesmo por indução. A última é mais trabalhosa, mas se dominada pode ser determinante para a aquisição de conhecimento pelas máquinas (Michalski, 1983). O resultado da indução é uma hipótese que deve posteriormente ser validada.

Para obtenção de hipóteses é utilizado um conjunto de dados como estímulo para o aprendizado, que pode em alguns casos estar disposto em tabelas semelhantes às de um bancos de dados. Sendo uma tupla do conjunto contendo vários atributos valorados que podem definir um atributo classe em dados rotulados ou sem atributo classe quando os dados não são rotulados. Este atributo classe quando presente pode assumir valores discretos para classificação ou contínuos para casos de regressão.

O aprendizado indutivo pode ser dividido em três formas, conforme Figura 2.4: o supervisionado, não supervisionado e semi-supervisionado. O aprendizado supervisionado é quando os dados estão rotulados com o atributo classe. A forma não supervisionada é utilizada quando os dados não são rotulados. Agrupamentos são exemplos de aprendizado não supervisionado. O semi supervisionado utiliza dados não rotulados e rotulados para o aprendizado.

Alguns paradigmas mudaram com o passar do tempo com o aumento do poder computacional. O surgimento do aprendizado fim a fim foi um dos paradigmas que impactaram na área de visão computacional. A partir dele as etapas de pré-processamento, extração de características e reconhecimento de padrões, passaram a ser realizadas em um algoritmo de aprendizado de máquina dentro de um mesmo modelo (Bojarski et al., 2016). Normalmente este modelo é construído utilizando algoritmos de aprendizado profundo.

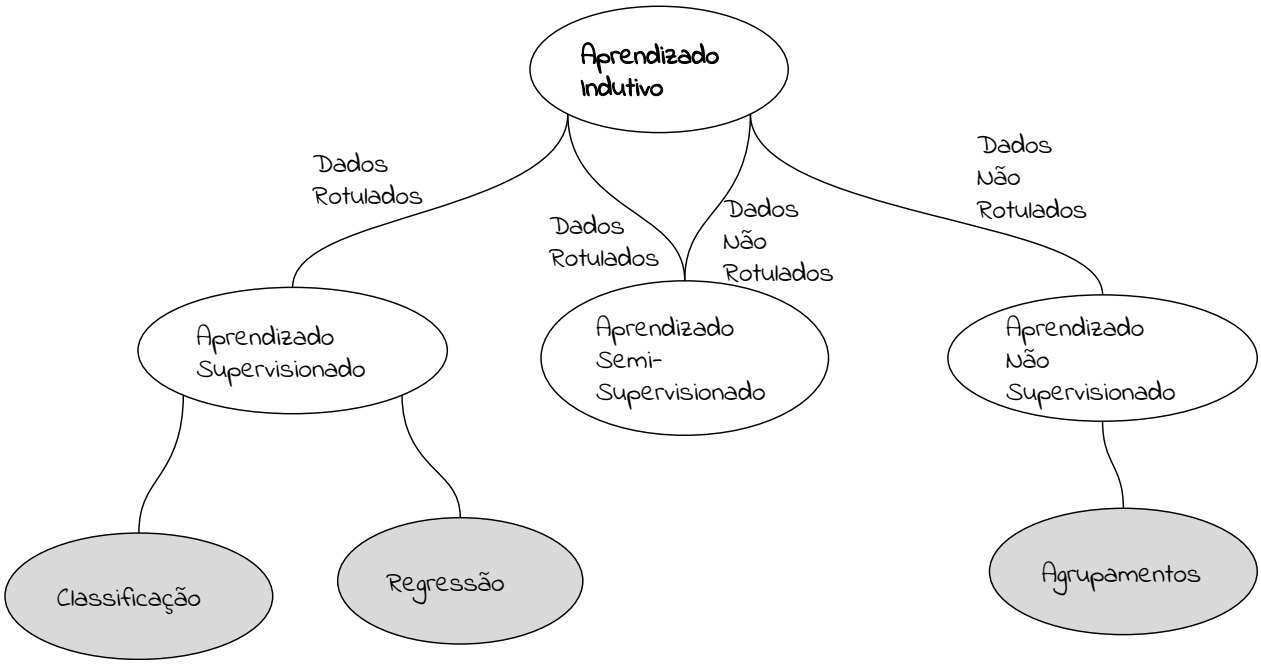


Figura 2.4: Diagrama das estratégias de aprendizado

2.2 Aprendizado Profundo

O aprendizado profundo é uma subárea do Aprendizado de Máquina, que permite o processamento de dados através de várias camadas de processamento (Goodfellow et al., 2016). Estes algoritmos nos últimos anos conseguiram avançar o estado da arte em diversos domínios como processamento de sinais, texto e imagens. (LeCun et al., 2015)

Algoritmos convencionais de aprendizado de máquina usualmente precisam de dados pré-processados. Máquinas de Vetores de Suporte (Vapnik, 1998) ou Árvores de Decisão (Quinlan, 1993) são exemplos clássicos de aprendizado de máquina que normalmente requerem dados pré-processados no formato de tabelas atributo-valor e usualmente não são capazes de receber dado bruto. Nos algoritmos de arquitetura profunda, o aprendizado ocorre através de múltiplas camadas de perceptrons, que em conjunto trabalham para encontrar o melhor caminho utilizando pesos e conexões distintas para chegar a uma hipótese. Abaixo serão detalhados alguns conceitos a fim de detalhar melhor a técnica de aprendizado profundo.

2.2.1 Perceptron

Em redes neurais profundas cada perceptron é uma representação de um neurônio. Os pesos, ou parâmetros, que conectam os neurônios simulam a sinapse do cérebro, assim estes pesos são denominados de pesos sinápticos (Rosenblatt, 1958). Ne tentativa de aprender, os pesos sinápticos mudam seus valores de acordo com a entrada do algoritmo. Essa multiplicação é apresentada na Equação 2.3. Observe que cada valor de entrada x é multiplicado pelo seu respectivo peso θ. A soma destes valores é dado por um produto escalar $\theta^T x$. Se os perceptrons forem utilizados em conjunto, possibilitam diferentes tipos de aprendizados.

$$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + .. + \theta_n x_n \tag{2.1}$$

2.2.2 Redes Neurais Artificiais

Uma rede neural é um processador distribuído, maciço e paralelo feito de processamento de unidades simples (perceptron) que têm uma propensão natural para armazenar conhecimento experiencial e torná-lo disponível para uso (Haykin, 2010). Para dirigir um veículo, o nosso cérebro utiliza parte dos seus 86 bilhões de neurônios (Azevedo et al., 2009). Dessas conexões partes delas são ativadas de acordo com a decisão a ser tomada como frear, acelerar e trocar de marcha do veículo. O estudo de LeCun et al. (1998) tentava imitar o funcionamento cérebro humano computacionalmente, através de redes neurais artificiais. Porém diferente do cérebro humano, as conexões das redes neurais artificiais clássicas ocorrem apenas em um sentido (*Forward*). A informação de cada neurônio passar por funções que definem a sua real saída para a próxima camada.

2.2.3 Funções de Ativação

Funções de ativação são utilizadas para transformar um sinal de entrada em outro sinal de saída para a proxima camada (Sharma and Sharma, 2017). O resultado da Equação 2.1, é o sinal que será submetido a uma função de ativação que será transformado de acordo com a função utilizada para possibilitar uma

relação não linear entre as entradas e saídas do neurônios. Essa não linearidade possibilita a rede neural representar hipóteses mais complexas. *ReLu*, *Logística*, *Tahn* são exemplos de funções de ativação não lineares. A *ReLU* na sua idealização buscou resolver o problema do *Vanish Gradient*, presente no uso da função *Logística* (Dahl et al., 2013). A formulação matemática da *ReLU* pode ser observada na Equação 2.2. Não há uma função exata para cada problema, cada uma possui suas particularidades.

$$ReLU(\theta^T x) = max(0, \theta^T x) \tag{2.2}$$

2.2.4 Normalização

O processo de transferência de informações pode utilizar processos de normalização como a Normalização de Batch (Ioffe and Szegedy, 2015). A última normaliza os valores com média 0 e variância 1 além de adicionar parâmetros β e α, que são aprendidos pelo modelo, e podem vir a substituir os valores padrões da média e da variância, respectivamente. O uso dessa norma se provou eficiente, pois melhora a velocidade de convergência do modelo em atributos que não são fortemente correlacionados (LeCun et al., 2012).

Outra opção de normalização que auxilia resolução de problemas de convergência, como a oscilação nos mínimos, são as inicializações dos pesos. Uma solução é a inicialização aleatória, porém outras soluções são mais complexas, como a inicialização de Kaiming (He et al. (2015)) e de Xavier (Glorot and Bengio (2010)). As Equações 2.4 e 2.3 representam matematicamente ambas as ideias de acordo com a camada l em que são aplicadas. Ambas, Xavier e Kaiming, foram utilizadas com as funções de ativação *tahn* e *ReLU* respectivamente em suas formulações.

$$W_{He}^{[l]} = \sqrt{\frac{2}{size^{[l-1]} + size^{[l]}}} \tag{2.3}$$

$$W_{Xavier}^{[l]} = \sqrt{\frac{1}{size^{[l-1]} + size^{[l]}}} \tag{2.4}$$

2.2.5 Funções de Perda

Todos os passos até o momento fazem parte processo de *Forward Pass*, que transforma uma entrada em uma saída (LeCun et al., 2015). Para melhorar os resultados, o aprendizado profundo é incremental a partir dos erros. Se um jogador faz uma determinada pontuação em um jogo qualquer e deseja melhorar a pontuação, estudará seus possíveis erros para pontuar mais. De maneira semelhante, o modelo utiliza a diferença entre o valor de saída e o valor esperado como forma de medir o desempenho do indutor. Se o valor de um imóvel era de $R\$10.000$ e o valor predito pelo algoritmo é de $R\$4.000$, o erro $J(\theta)$ seria de $R\$6.000$.

Funções de perda são utilizadas para avaliar o modelo e direcionar o aprendizado (Bottou, 2010). Para classificações é utilizada a Entropia Cruzada, que penaliza o modelo proporcional a divergência do resultado esperado da classificação. O cálculo é feito com o auxílio de *One Hot Vectors* que armazena o valor de 1 para a classificação esperada e 0 para as demais. Dessa forma, o erro é dado pela soma das probabilidade $p(i)$ para a classe i multiplicadas pelo logaritmo da probabilidade h resultante do classificador. A soma reflete o erro de todas as predições, como pode ser observado na Equação 2.5.

$$
\begin{aligned}
EntropiaCruzada &= -\sum_{i=1}^{c} p(i) \log h(i) \\
MSE &= \frac{1}{N} \sum_{e=1}^{N} (h(x^{(e)}) - y^{(e)}))^2 \\
MAE &= \frac{1}{N} \sum_{i=1}^{N} (h(x^{(i)}) - y^{(i)}))
\end{aligned}
\tag{2.5}
$$

2.2.6 Retropropagação

O erro é utilizado como entrada para o algoritmo de *Backpropagation* ou retropropagação para ajustar os parâmetros do modelo aos dados. Essa tarefa é realizada em computadores através do cálculo do gradiente descendente (LeCun et al., 1998) e retropropagação pela regra da cadeia. Assim procura minimizar a função de custo $J_{min}(\theta)$, de forma a convergir para regiões de menor erro, conforme ilustrado na Figura 2.5.

Um dos principais problemas relacionados ao gradiente descendente são os

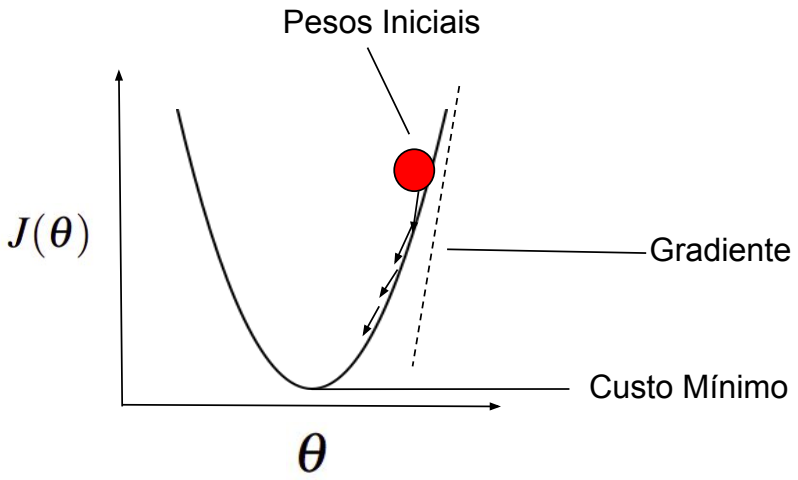


Figura 2.5: Caminho do gradiente descendente em busca do mínimo local de uma parábola

mínimos locais, exemplificado na Figura 2.6. Dessa forma processo de descida do gradiente pode ser auxiliado por taxas de aprendizado que controlam a magnitude do ajuste dos pesos, para que atualizações maiores permitam a saída dos mínimos locais.

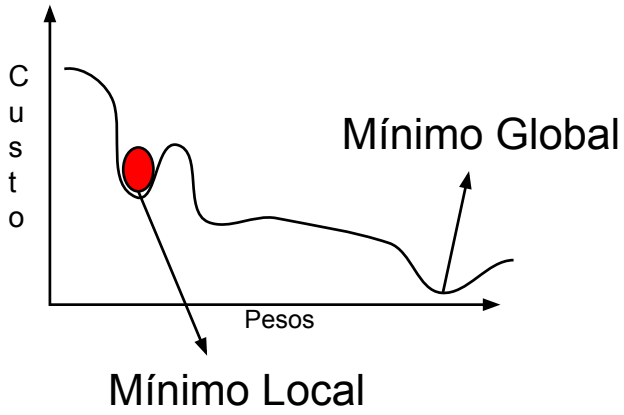


Figura 2.6: Mínimo local de uma função composta

A partir da derivada parcial do gradiente descendente e levando em consideração o resultado da função de perda, é obtida a regra de atualização representada na Equação 2.6. Esta regra, sem algoritmos de otimização, atualiza os parâmetros a cada leitura de exemplos do conjunto de dados, sendo necessário o

carregamento total do conjunto de dados na memória (Bottou, 2012).

$$\theta_j = \theta_j - \alpha(\nabla \mathrm{J}) \tag{2.6}$$

2.2.7 Otimizadores

Com quantidades enormes de dados e limitações de memória, as atualizações podem ocorrer com auxilio de algoritmos de otimização que atuam em cima dos hiperparâmetros, escolhidos antes do processo de treinamento, e da retropropagação. Variações do Gradiente Descendente como o Gradiente Descendente Estocástico (SGD), Gradiente Descendente em Lotes ou Gradiente Descendente em Mini Lotes são opções. Por outro lado, algoritmos como Adam, Momentum, RMSProp e Adagrad foram propostos com o intuito de facilitar o processo de descida do gradiente.

O Gradiente Descendente Estocástico adiciona aleatoriedade ao conjunto, para os dados serem lidos de maneira aleatória através de lotes menores de dados. A quantidade de dados presente varia de acordo com o tamanho do lote BS configurado como hiperparâmetro baseado usualmente na capacidade de memória da GPU. Dessa forma o algoritmo atualiza os pesos do modelo a cada interação completa de um mini lote de dados e não do lote completo que é quando se completa uma época de treinamento. É importante a escolha de um bom tamanho do lote, pois se bem configurado pode-se obter resultados semelhantes quanto a variar a Taxa de Aprendizado durante o treinamento. Com um lote grande e Taxa de Aprendizado alta a curva em direção ao mínimo sofre menos variações, os pesos sofrem menos atualizações e menos tempo de treinamento. Para o SGD com Adam, Momentum e Nesterov Momentum essas abordagens se mostraram válidas (Smith et al., 2017).

RMSProp, Adagrad e Adam são algoritmos muito próximos, pois alteram a taxa de aprendizado no decorrer do treinamento. O RMSProp altera por um fator dado pela média dos gradientes ao quadrado, o Adagrad utiliza o histórico dos gradientes passados para reduzir a taxa, o que leva a taxas de aprendizado muito pequenas. Por último, o Adam adiciona correção do viés com a exclusão nas primeiras iterações a fim de reduzir sua influência no decaimento da taxa de aprendizado (Kingma and Ba, 2014).

2.2.8 Arquiteturas Profundas

O surgimento de arquiteturas com múltiplos níveis de abstração deu forma ao aprendizado profundo e permite respostas cada vez mais complexas (LeCun et al., 2015). O aumento no nível de abstração foi possível através da adição de camadas ocultas entre a entrada e a saída tornando a arquitetura profunda. Essas adições fazem com que a rede neural aprenda uma função composta por outras várias funções, como na Equação 2.7.

$$y = h_3(h_2(h_1(x, \theta_1), \theta_2)\theta).. \tag{2.7}$$

A transição de arquiteturas rasas para o uso de arquiteturas profundas se deu principalmente pelo uso de GPUs para treinamento, que permitiu processamento de lotes maiores de dados, aumento na velocidade de processamento e redes cada vez mais profundas, como a da Figura 2.7. Surgiram também vários estudos e implementações complexas que acrescentam camadas extras que não necessariamente são totalmente conectadas. Um exemplo são Redes Neurais Convolucionais ou CNN que implementam camadas de extração automática de características.

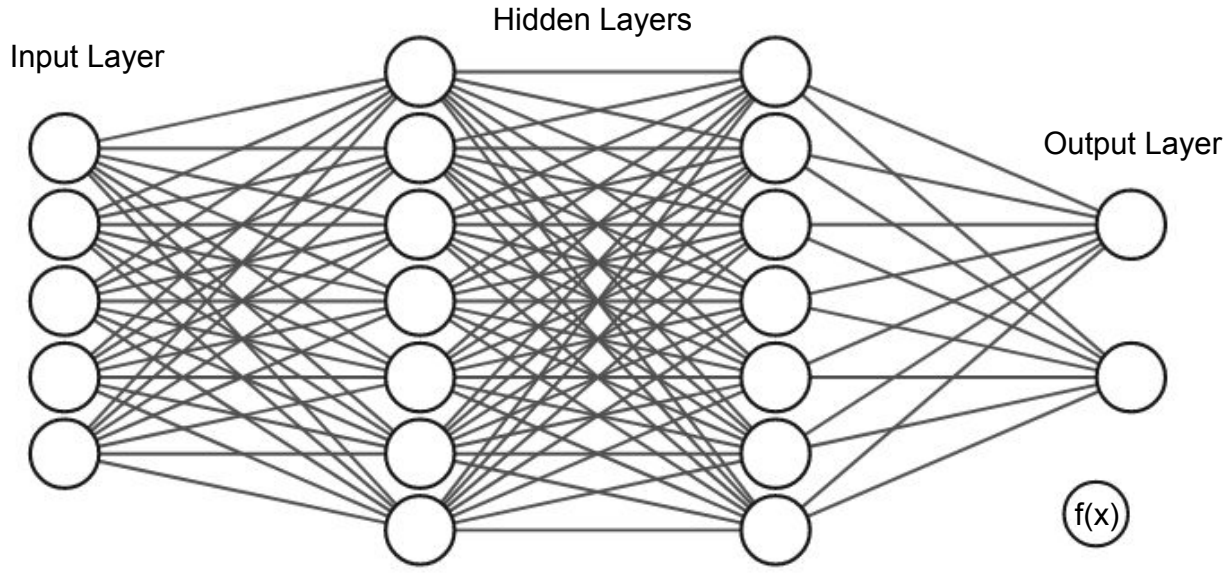


Figura 2.7: Rede Neural Feedforward exemplo criada por meio do software NN SVG. 4 camadas, sendo duas camadas ocultas ou *Hidden Layers*

2.2.9 Redes Neurais Convolucionais

A primeira estrutura de Rede Neural Convolucional(CNN) foi proposta por Fukushima em 1988 (Fukushima, 1988), porém dado ao baixo poder computacional da época não foi muito utilizada. Na década de 90 a LeNet (LeCun et al., 1998) conseguiu resultados com o uso de técnicas de gradiente, aplicadas ao problema de classificação de dígitos escritos a mão.

As arquiteturas mais populares de CNNs consistem em blocos básicos compostos por convolução, pooling e a camada densa. A LeNet (LeCun et al., 1998), AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2014) são exemplos dessas redes. Outras porém, são mais elaboradas, como a GoogleNet (Szegedy et al., 2015) com *Inception Modules* e as ResNets (He et al., 2016) com conexões residuais.

A arquitetura conhecida como LeNet-5, conforme Figura 2.8 é composta por 2 blocos de compostos por uma Convolução e uma camada de *Pooling*, 2 camadas totalmente conectadas e por último a saída obtida por uma Gaussiana.

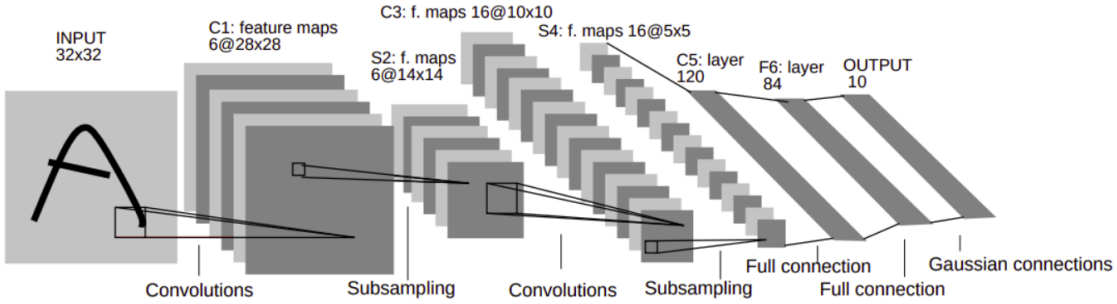


Figura 2.8: Arquitetura LeNet-5 LeCun et al. (1998). Uso autorizado pelo autor

A AlexNet proposta por Alex Krizhevesky conseguiu resultados relevantes frente aos algoritmos tradicionais no *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* (Russakovsky et al. (2015a)), considerado o desafio mais difícil da ImageNet. Estes resultados foram alcançados através do uso de uma rede profunda convolucional com paralelização do treinamento das imagens em GPUs. Essa abordagem de treinamento é utilizada até hoje em diversos modelos. A estrutura da AlexNet é composta pela entrada do modelo seguido por 5 camadas convolucionais, das quais as duas primeiras continham operações de *Max Pooling*, e 3 camadas densas na penúltima camada. A partir da AlexNet surgiram diversas redes, cada vez mais profundas, entretanto começaram a aparecer problemas. Por

muito tempo foram utilizadas funções de ativação como a logística e a tangente hiperbólica. Com isso o problema do *Vanish Gradient* também estava presente, fazendo com que os parâmetros perdessem sua representatividade mesmo com valores significativamente grandes. Posteriormente surgiram propostas que vieram para minimizar, ou sanar esse problema, como a ReLU já mencionada na Seção 2.2.3.

Algumas arquiteturas possuem variações, como a ResNet (He et al., 2016) com versões de 34, 50, 101, 152 camadas. A ResNet50 possui 49 camadas de convolução e uma única camada totalmente conectada. As conexões residuais presentes nessa arquitetura fazem com que diferentes caminhos possam ser aprendidos no treinamento, adiantando informações à camadas mais profundas através de *Skip Connections*. Para isso é necessário manter tamanhos semelhantes dos mapas de características da *Skip Connection* com o fluxo normal para que possam ser somados. Essa implementação permite com que blocos não tão significativos para o problema sejam ignorados, minimizando o problema do *Vanish Gradient* e permitindo redes ainda mais profundas. Redes como a ResNet são *Backbones* para as mais diversas aplicações que possuem imagens como entrada.

2.2.10 Convolução

Como citado anteriormente Redes Neurais Convolucionais implementam extração automática de características, sendo uma Rede Neural Feedforward que possui pelo menos uma camada Convolucional (Goodfellow et al., 2016). Caracteristicamente as estrutura dessas redes são implementadas por meio do uso de camadas de convolução, *pooling* e camadas densas. Este tipo de Rede Neural está presente no estado da arte em diversas áreas, como processamento de linguagem natural, sinais, modelagem 3D, dentre outras e principalmente processamento de imagens.

Na camada convolucional presente na CNN é aplicada a operação de convolução, como extrator de características e possibilita que representações 2D bem como 3D possam ser aprendidas (Goodfellow et al., 2016). Esta camada contém filtros também chamados de Kernels que percorrem a imagem. Ao percorrer a imagem, os pixeis da imagem são multiplicados pelos pesos do filtro e posteriormente somados. Essa operação gera uma nova matriz que contem o resultado do filtro na posição $x[i, j]$ na matriz resultante, chamado campo receptivo. Vá-

rias funções podem ser aprendidas pelos filtros de convolução, como detecção de borda, de identidade, suavização, ou adaptadas ao problema. Desta maneira uma convolução completa sob a imagem gera um mapa de características, ou *Feature Maps*. A saída é utilizada por funções de ativação antes de ser repassado às próximas camadas.

Sem a convolução, seria necessário aprender pesos para cada posição de uma matriz. Em uma imagem 1.000 x 1.000, sem a operação de convolução, aproximadamente 1 milhão de pesos seriam aprendidos. No entanto, com a operação de convolução os pesos são relativos somente ao filtro, reduzindo o número de pesos e a quantidade de memória.

Figura 2.9: Exemplo de operação de convolução

A Figura 2.9 contém um filtro 3x3 que percorre uma imagem 5x5 a fim de gerar um mapa de características 3x3 com os resultados das operações de convolução. No exemplo foi utilizado somente um filtro, o numero de filtros é um parâmetro a ser escolhido durante a formulação da arquitetura. Uma camada *Pooling* pode estar associada às convoluções em algumas arquiteturas. Essa camada utiliza filtros para modificar a dimensão espacial como altura e largura, sem modificar número de mapas de características representado pelas dimensões. A saída da camada de *Pooling* (Zhou and Chellappa, 1988) pode variar de acordo com a operação utilizada, ser o máximo ou a média entre os pixels, conforme Figura 2.10. O Pooling busca fazer com que representações se tornem menos suscetíveis a pequenas variações das imagens preservando a localização das bordas.

A convolução bem como o *Pooling* possuem hiperparâmetros como o *Stride* e *Padding* que são configurados durante a construção da arquitetura. O passo representa o número de pixeis a serem deslizados pelo filtro sob a imagem após

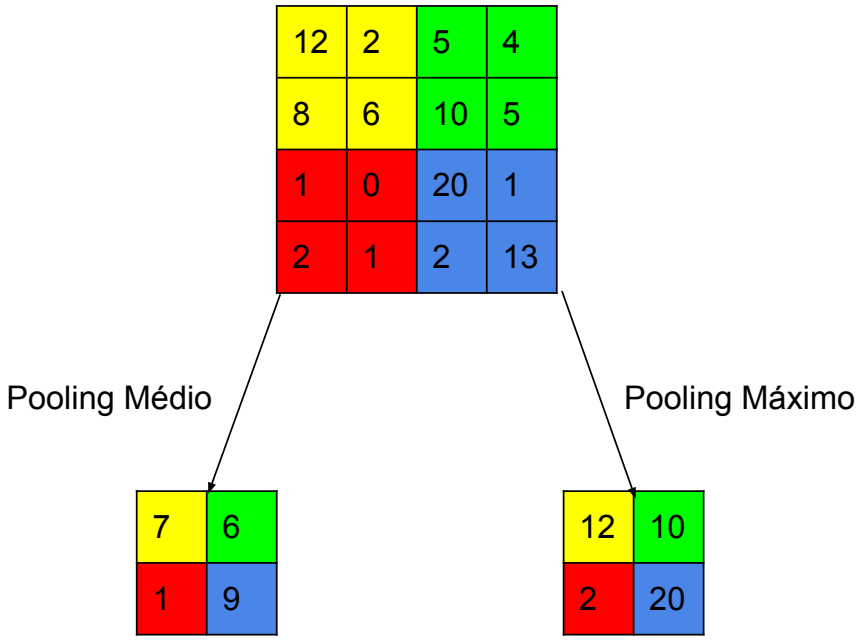


Figura 2.10: Operação de pooling.

uma operação completa, por outro lado o *Zero Padding* mantém o tamanho da imagem após a aplicação do filtro com a adição de zeros nas bordas da imagem. Ambas as operações estão visíveis na Figura 2.11, onde p representa quantas unidades de borda serão adicionadas e s quantas unidades serão percorridas pelo filtro.

Com esses conceitos um passo importante nas arquiteturas pode ser implementado, pois possibilita o uso dessas características para outras tarefas mais específicas, como por exemplo detectar e localizar objetos distintos em uma mesma imagem.

2.3 Deteção de Objetos

A detecção de objetos consiste em localizar um objeto na imagem e classifica-lo. Essa tarefa é base para muitas aplicações, pois informa o objeto e seu posicionamento na imagem, dessa forma segmentação, rastreamento de objetos, dentre outras tarefas podem ser realizadas.

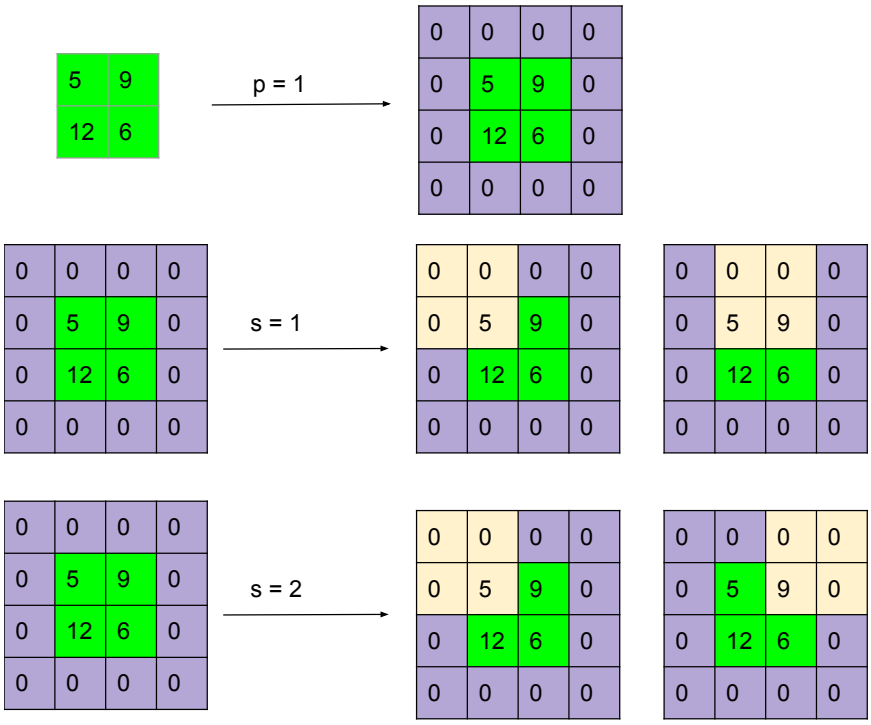


Figura 2.11: Operação de de *Padding* na primeira linha e *Stride* na segunda e terceira linha

2.3.1 Frameworks

Na literatura há abordagens para construção de *frameworks* genéricos para serem utilizados em diversas aplicações (Ren et al., 2015), ou trabalhos que modificam esses *frameworks* para adaptar o seu funcionamento a um contexto específico com uma saída específica como Deng et al. (2019b) na RetinaFace. Em 2012 o número de publicações era de 600, em 2018 o número de publicações que tem seu foco em detecção de objetos beirava 1200 em um único ano (Zou et al., 2019).

Os *frameworks* genéricos podem ser divididos em duas famílias, uma mais complexa que a outra. Ambas levam em conta o acerto quanto a localização e classificação para avaliarem o desempenho.

- *One-Stage Detectors*: *Single Shot Detector* (SSD) (Liu et al., 2016), YOLOV3 (Redmon and Farhadi, 2018), dentre outras.
- *Two-Stage Detectors*: Mask R-CNN He et al. (2017), Faster R-CNN Ren et al. (2015), dentre outras.

A primeira família de *One-Stage Detectors* é composta por somente um estágio, sendo as tarefas de localizar o objeto e classificar realizadas como tarefas sequenciais e não concorrentes. Por outro lado, os *Two-Stage Detectors* implementam a detecção como um processo de dois estágios, primeiro localiza os objetos na imagem e posteriormente classifica. A diferença nos processos é que na segunda família podem ser empregadas redes específicas para o processo de localização, por exemplo. Um exemplo de *Pipeline* complexo (Ren et al., 2015) com o uso de *Region Proposal Networks*. Essa abordagem implementa uma rede específica para propor regiões que serão utilizadas pelo classificador.

2.3.2 Anchor Boxes

Buscando melhorar o tempo de processamento e as métricas em seus algoritmos, algumas abordagens como o uso de *Anchor Boxes* são comuns em detectores de objetos (Lin et al., 2017), (Redmon and Farhadi, 2018). Normalmente um processo de detecção percorre a imagem em busca de localizações. A proposta do uso de *Anchor Boxes* parte do princípio de utilizar caixas pré estabelecidas. A predição utilizando caixas pré estabelecidas ocorre separadamente para cada uma, resultando em informações da presença ou não de objetos P_c, as coordenadas B_x, B_y, B_h, B_w da localização e a classe C_1, C_2 predita, conforme Figura 2.12 que ilustra o vetor de uma única *Anchor Box*. Dessa maneira calcula-se o IoU com o objeto, caso seja mais de 50% deve retornar a classificar como resultado o objeto de maior interseção com a caixa. Se a maior interseção é menor que 40% não devem ser retornadas classificações.

Por fim é aplicado o algoritmo de *Non Maximum Supression* (Neubeck and Van Gool, 2006), funcionando de forma repetitiva enquanto houverem caixas preditas a serem analisadas, iniciando pela caixa de maior probabilidade de conter um objeto e compara a sua interseção com as demais. Se for maior que um determinado limiar estas devem se unir somando as intersecções. Posteriormente seleciona-se novamente a caixa com maior probabilidade, até que não sobre nenhuma outra caixa.

2.3.3 SSD

Um exemplo de arquitetura *One Stage Detector* é a SSD (Single Shot MultiBox Detector). Essa abordagem leva esse nome pois em um único passo de *Forward*

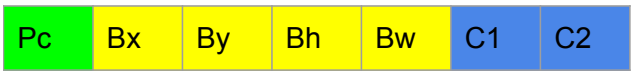


Figura 2.12: Estrutura de uma *Anchor Box*

detecta objetos utilizando várias *Bouding Boxes*. A ideia é que na arquitetura há um conjunto de *Boxes* previamente definidas e a detecção é baseada na busca das *boxes* que contenham objetos ou parte de objetos que tenham representatividade.

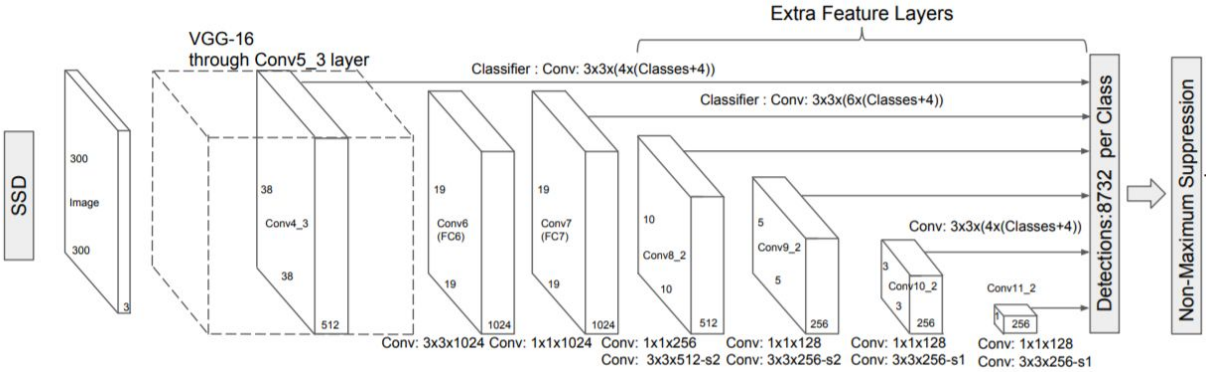


Figura 2.13: Arquitetura SSD (Liu et al., 2016). Uso autorizado pelos autores.

Sua arquitetura é composta pela base da VGG-16, adicionadas camadas extras de convolução com diversos tamanhos a fim de extrair mapas de características de tamanhos distintos ao invés de uma CNN específica para essa tarefa, conforme Figura 2.13. Por fim uma convolução é aplicada em cada mapa de característica tendo como resultado 4 + n canais, sendo 4 deles para altura, largura, centro da coordenada x e y de uma *Bouding Box* e o restante referente as probabilidades de n classes como pode ser observado na Figura 2.14.

A maior probabilidade é escolhida para representar a classe presente na *Bouding Box*. Vale ressaltar que para uma única área podem haver várias *Bouding Boxes* verdadeiras com pelo menos 50% de sua área dentro do *Ground Truth* para aquele objeto. Dessa forma é aplicado o algoritmo de *Non-Max Supression* para reduzir esse número para somente uma. Na SSD300 que utiliza imagens de 300x300 como entrada, há 8732 *Bouding Boxes* pré definidas através de resoluções pré selecionadas das convoluções. A perda do modelo é calculada separadamente para a localização dos objetos pelas *Bouding Boxes* e pela perda na

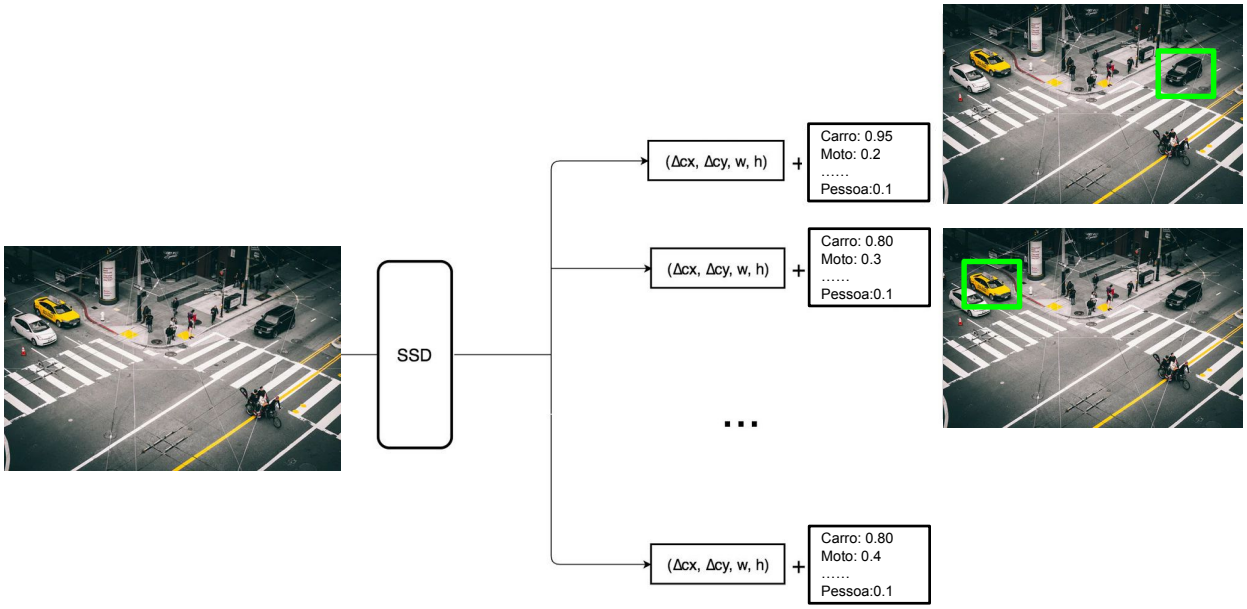


Figura 2.14: Regressão e classificação das *Bouding Boxes* na SSD. Figura própria.

classificação dos mesmos, sendo assim uma perda composta.

2.3.4 Faster R-CNN

No outro grupo de *Two-Stage Detectors* a Faster R-CNN (Ren et al., 2015) é um exemplo clássico, pois sua ideia teve um mudanças significativas no estado da arte. A ideia que serve como base para essa arquitetura é a de utilizar uma rede específica *Region Propostal Network* (RPN) que é treinada especificamente para geração de regiões que podem conter ou não objetos. Utiliza como base a Fast-R CNN (Girshick, 2015), que como SSD tem a VGG-16 como base do modelo.

De forma mais detalhada, a imagem é utilizada por uma série de camadas de convolução que por sua fez resultará em um mapa de características. O resultado será utilizado pela RPN, onde cada local no mapa de características irá gerar k *Bouding Boxes* de escalas e proporções distintas. Cada *Bouding Box* será classificada em "Possui objeto ou não"e terá também como saída a regressão de suas coordenadas centralizadas em x e y com a altura e largura. O número de âncoras totais é dado por $L * A * k$, onde L é a largura da entrada, A é a altura da imagem e k um hyperparâmetro. Ambas as redes são treinadas separadamente, a perda da RPN é calculada com base na perda da classificação L_{cls} e regressão

das coordenadas L_{reg}. A informação das regiões propostas pela RPN é repassada pela rede para a parte da arquitetura que irá fazer a detecção. As regiões propostas pela RPN, como dito anteriormente, podem ter diversas escalas e proporções com isso há a necessidade de aplicar uma operação de *RoI Pooling* para que as dimensões estejam de acordo com as camadas totalmente conectadas.

Sendo assim, as regiões propostas serão classificadas e a saída será composta pelo objeto que está naquela região em conjunto com suas respectivas coordenadas. Uma grande diferença para a SSD é que na Faster-R CNN as regiões são geradas e não pré estabelecidas, podendo ser mais precisa porém mais lenta dado o número de âncoras geradas. A arquitetura da Faster-R CNN pode ser vista na Figura 2.15.

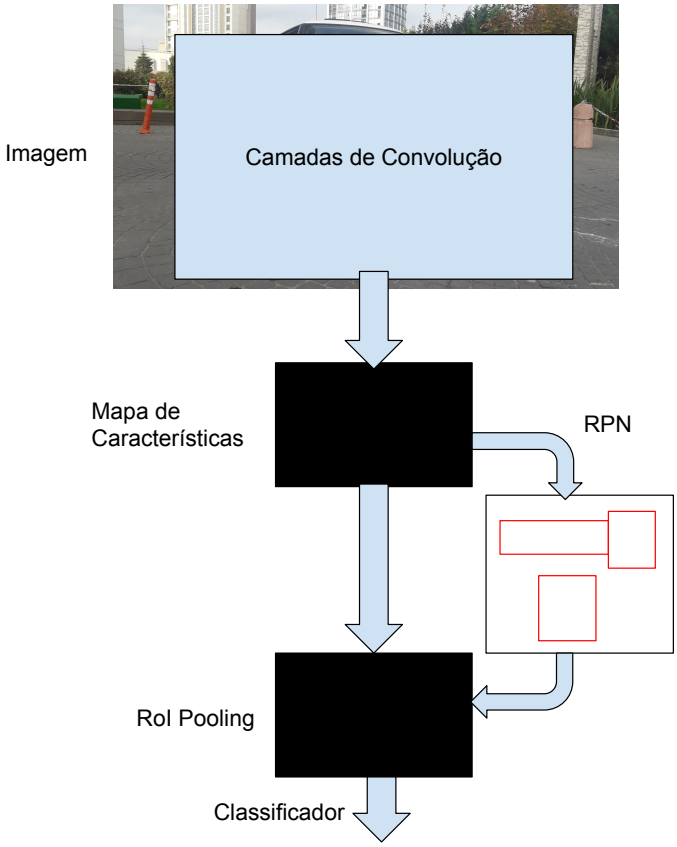


Figura 2.15: Estrutura da Faster-R CNN

O uso do aprendizado profundo tem feito com que diversas áreas possam ser estudadas. Contudo a necessidade de grandes quantidades de dados pode ser um problema, por mais complexos que sejam os algoritmos. O que leva a pensar em certos casos que reescalar a arquitetura em profundidade nem sempre é sinal

de maior precisão. A precisão pode ser refletida também na largura das camadas, maior resolução das imagens ou a junção desses atributos que se forem manipulados da maneira correta é possível obter resultados precisos para o problema e administrar o custo computacional (Tan and Le, 2019).

2.3.5 Métricas de Avaliação

A avaliação dos classificadores pode utilizar de métricas como *Accuracy*, *Precision*, *Recall* e *F1 Score*. Em detectores de objetos é utilizado também o *Mean Average Precision* (mAP) (Lin et al., 2014). Nessa última medida, para cada predição é calculada a *Precision* e o *Recall*. Para isso exemplos são considerados Verdadeiros Positivos quando $IoU > 0.5$ se não são Falsos Positivos (Everingham et al., 2010). Então esses exemplos são ordenados em sentido decrescente e plotados em um gráfico *Precision*/*Recall*. O gráfico plotado é serrilhado e decrescente, dada a ordenação do *Recall* e a variação da *precision* conforme é feita a leitura dos exemplos. O *mean Average Precision* mAP é calculado pela área abaixo da curva com o auxílio da interpolação de pontos, substituindo a *precision* em cada *Recall* pela maior a sua direita. Dessa forma ajuda a suavizar a curva e diminuir os serrilhados. No dataset PASCAL VOC (Everingham et al., 2010) em 2012 eram utilizados 11 pontos de $0, 0.1 ... 0.9, 1$ para dividir o *Recall* e então calcular a máxima *precision* em cada ponto para no final obter a média do somatório das precisões. No conjunto de dados COCO para avaliação são utilizados 101 pontos ao invés de 11 e o limiar de IoU a ser considerado pode ser variado. Por exemplo para um $AP@[.5 : .95]$ é uma média das AP começando com limiares de IoU em 0.5 até 0.95 variando de 0.05, para as 80 classes presentes no dataset.

2.4 Rastreamento de Objetos

Junto ao uso de detectores de objetos, técnicas de rastreamento de objetos são estudadas dependendo do problema a ser abordado. Essa área de rastreamento de objetos possui diversas aplicabilidades que buscam rastrear pedestres, carros ou até partes do corpo com intuito de auxiliar em diversas tarefas. Esse rastreamento consiste em estimar as posições do objeto dado um ponto inicial de detecção, nos próximos quadros de uma sequência de imagens (Yilmaz et al., 2006). Um exemplo prático é um pedestre que anda sob uma calçada, dado o momento inicial em que aparece no vídeo a ideia é estimar suas posições quadro após quadro até que não seja mais possível rastrear. Um carro em um tráfego de veículos também pode ser rastreado como na Figura 2.4.

Figura 2.16: Rastreamento de um veículo na rodovia

Vários algoritmos implementam essa técnica, cuja avaliação se da pelo somatório do número de quadros em que o objeto esteve presente e quantos desse total foram detectados pelo algoritmo, levando em conta perda de identificação, a

re identificação, falsos positivos e falsos negativos (Milan et al., 2016). Nos últimos anos vários algoritmos se mostraram precisos (Wojke et al., 2017; Bochinski et al., 2017a; Henriques et al., 2014), porém quanto mais precisos maior é seu custo devido ao uso de mais informações consequentemente o aumento do custo computacional.

Uma maneira simples de rastrear um objeto, é atribuir uma identificação a um objeto e no próximo quadro verificar qual das detecções se assemelha ao objeto, a que estiver mais próxima da identificação anterior possivelmente é sua próxima posição. Para isso existem dois tipos de rastreadores, os que rastreiam apenas um objeto de interesse são conhecidos por Rastreadores de Objetos Únicos. Quando o rastreio ocorre para vários objetos, são conhecidos como Rastreadores de Objetos Múltiplos do inglês *Multi Object Tracking* (MOT) (Milan et al., 2016).

Exemplos de algoritmos são os baseados em filtros de correlação (Bolme et al., 2010). Inicialmente o objeto é centralizado em uma *Bouding Box*, então os filtros trabalham em conjunto com rastreadores para através da operação de correlação detectar o objeto de interesse no próximo quadro. Usualmente estes filtros utilizam uma Gaussiana como função para gerar filtros 2D, onde a altura do filtro tem seu ponto mais alto no centro do objeto e diminui assim que se distancia do centro.

Para otimização, a correlação é computada no domínio da Transformada Rápida de Fourier, que pelo teorema da convolução quando levadas ao domínio da frequência a correlação do filtro sob a imagem se torna uma multiplicação de elementos (Guan et al., 2018). Após a operação os resultados são trazidos de volta ao domínio espacial pela Transformada Inversa de Fourier permitindo a detecção e rastreio do objeto. O rastreamento ocorre pela distância do ponto que corresponde de maior altura da correlação com o ponto de maior altura do quadro anterior. Com essas configurações o algoritmo atingiu 669 quadros por segundo em uma CPU Core 2 Duo de 2.4Ghz, podendo ser utilizada em aplicações de tempo real.

Existem outras abordagens que utilizam de Redes Neurais Convolucionais para solucionar o problema. Como é o exemplo do TRACKTOR (Bergmann et al., 2019), este algoritmo utiliza apenas detecção de objetos para rastrear múltiplos objetos, o algoritmo é treinado para regredir coordenadas de *Bouding Box* para as mais próximas possíveis no próximo quadro. A implementação do algoritmo

consiste na detecção de um objeto e o rastreio se dá pela regressão das coordenadas do quadro $t-1$ para uma nova detecção no quadro t e a re atribuição do identificador. Quando a oclusão ocorre, o algoritmo de *Non Maximum Supression* é utilizado tendo como entrada as caixas de detecção que estavam na região da oclusão. O TRACKTOR pertence ao grupo de rastreadores que processam informações *Online*, ou seja, conforme a detecção ocorre os objetos são rastreados. O algoritmo original atingiu 1.7 quadros por segundo (Milan et al., 2016), excluindo a detecção, de acordo com os autores. Não foi específicado o hardware de teste.

Um exemplo de rastreador do tipo *offline* é o IoU-Tracker (Bochinski et al., 2017b), que não utiliza informações da imagem para o rastreio. A entrada do algoritmo é um histórico de detecções, obtido por um algoritmo de detecção de objetos qualquer. Esse histórico contém em cada linha dados do quadro em que foi detectado, as coordenadas da caixa de detecção e a confiança da classificação. O algoritmo utiliza o cálculo de IoU como base para o rastreio, este é configurado com um limiar σ_{iou} para a atribuição de novos identificadores ou re atribuição dos já existentes. Se a interseção for maior que o limiar estabelecido o identificador é reatribuído, caso menor, um novo identificador é atribuído. O problema dessa abordagem é a alta mudança de identificadores. Durante o rastreamento são utilizadas somente detecções que tenham confiança maior que o limite σ_l configurado no rastreador.

O rastreamento somente por IoU é rápido porém não é um dos mais precisos pois por ser um rastreamento por detecção, ou seja, primeiro são detectados os objetos e posteriormente o rastreamento. Isso se dá pela presença de falsos negativos nas detecções dos algoritmos que por sua vez não possuem sequência nos quadros seguintes e acabam por atrapalhar no rastreio, e como não há informação visual não é possível estimar a posição do objeto. A implementação inicial do algoritmo é um exemplo que não deu certo, porém foi corrigida em um nova versão estendida do algoritmo. Na versão estendida do algoritmo chamada *Extended Iou-Tracker* (Bochinski et al., 2018) os autores se preocuparam em utilizar uma abordagem de detecção de um único objeto para casos em que há falsos negativos e quadros onde a detecção é perdida. Sendo assim há uma aplicação do conceito de rastreamento de um único objeto em detecção de múltiplos objetos.

O TRACKTOR bem como o IoU-Tracker com sua versão estendida estão, na segunda e quarta posição respectivamente, no desafio de rastreio do CVPR 2019

(Dendorfer et al., 2019). A primeira posição do desafio dos rastreadores não teve seu artigo publicado até a escrita dessa defesa. Entretanto há outras opções viáveis e robustas na literatura que oferecem um bom balanço entre rastreamento e performance, como o DeepSort (Wojke et al., 2017).

No DeepSort após uma primeira detecção por parte da CNN as detecções seguintes são obtidas a partir dos Filtros de Kalman (Kalman, 1960) que constituem o processo do algoritmo de rastreamento. O rastreamento ocorre com base em 3 fontes de dados: Detecções, Filtros de Kalman e Algoritmos de Correspondência. O algoritmo DeepSort resolve a associação entre novas detecções e as predições dos filtros de Kalman por meio do algoritmo Hungaro como um problema de atribuição. Além disso, são integradas informações de movimento que correspondem a distância de Mahalanobis e de aparência extraídas por CNN que serão posteriormente relacionadas por meio da distância de cosseno. Ambas as distâncias combinadas formam um único peso.

No algoritmo, quando um objeto é detectado este passa para o estado inicial de Tentativa ou Não Confirmado, na publicação original quando esse objeto tem 3 sucessos de relacionamento seguidos com uma detecção esse passará ao estado de Confirmado, caso contrário terá seu estado modificado para Deletado.

Os resultados da associação produzem os 3 seguintes conjuntos: Novos rastreadores, rastreadores atualizados e rastreadores não atualizados. Então é aplicada uma última operação de intersecção entre as áreas dos objetos de cada um desses conjuntos. Consequentemente é calculado para cada rastreador k a quantidade de frames desde sua associação mais recente com algum rastreador. Se o tempo for maior que A_{max} o rastreador é descartado do conjunto. O fluxo é reiniciado com os rastreadores com a adição de novas detecções. O fluxograma do algoritmo DeepSort pode ser analisado na Figura 2.17.

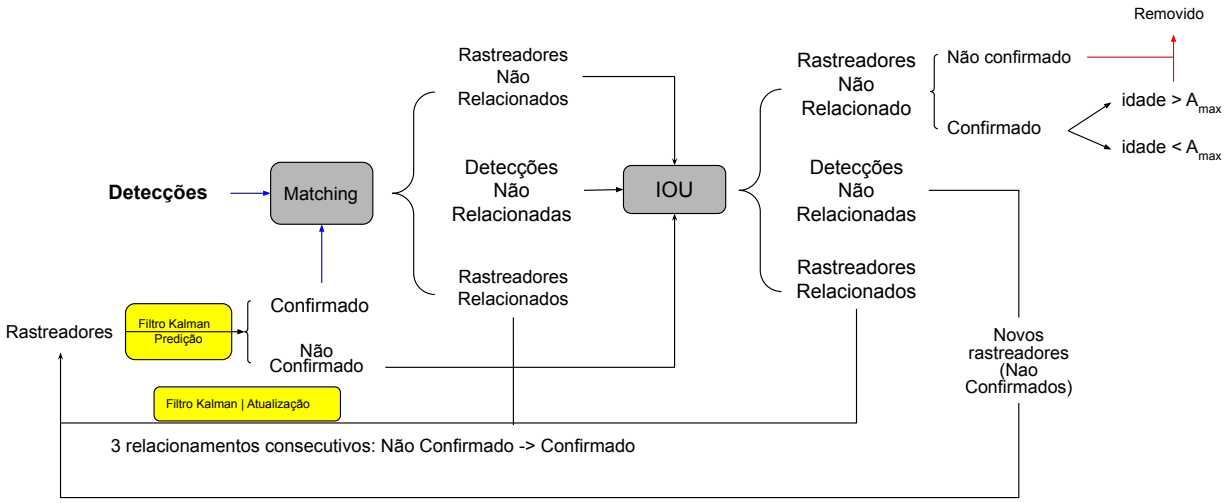


Figura 2.17: Fluxograma de relacionamento de detecções do algoritmo DeepSort

2.5 Otimização de Inferência

Quando combinados os rastreadores de objetos às detecções há um aumento no uso do poder computacional considerando o número de operações matemáticas para que o rastreamento se torne possível. Algumas bibliotecas podem auxiliar na otimização da inferência dos algoritmos de detecção. Ao considerar que o poder computacional das GPUs possibilitou a inferência em tempos cada vez menores (Wang et al., 2019) quando comparado às CPUs, surgiram também bibliotecas que otimizam ainda mais esse processamento como no caso da biblioteca TensorRT exclusiva em placas Nvidia. Com o auxílio dessa biblioteca é possível otimizar a arquitetura e também os pesos das conexões de rede totalmente conectadas, dentre outras otimizações de baixo nível.

Em uma arquitetura totalmente conectada várias camadas são criadas, com diferentes funções, operações e distintos caminhos. Para o treinamento um caminho bem descrito e bem formado pode acarretar em uma boa precisão, porém uma vez treinada é possível otimizar esses caminhos, já os melhores pesos para tais foram aprendidos durante o treinamento e na inferência não são atualizados. Dessa forma podemos fundir camadas em uma só, isso faz com que seja criado apenas um *Kernel* de operação pela placa de vídeo e assim impactando positivamente no *Troughput* da inferência. Um exemplo é um bloco sequencial formado por uma Convolução(C) 3x3, *Bias*(B) e uma Ativação ReLu(R), essas operações

podem formar um único bloco CBR 3x3 com o auxílio da biblioteca TensorRT para inferência. Nesse caso houve uma fusão vertical das instruções. Da mesma forma, várias camadas de CBR podem ser fundidas horizontalmente ou verticalmente, mantém seus pesos originais mas durante a execução da camada é utilizado somente uma camada CBR pois convoluções em *frameworks* padrões lançam kernels para cada convolução, mesmo que iguais. Esses dois exemplos são uma das possíveis otimizações de um modelo para inferência e podem ser melhor observados na Figura 2.18

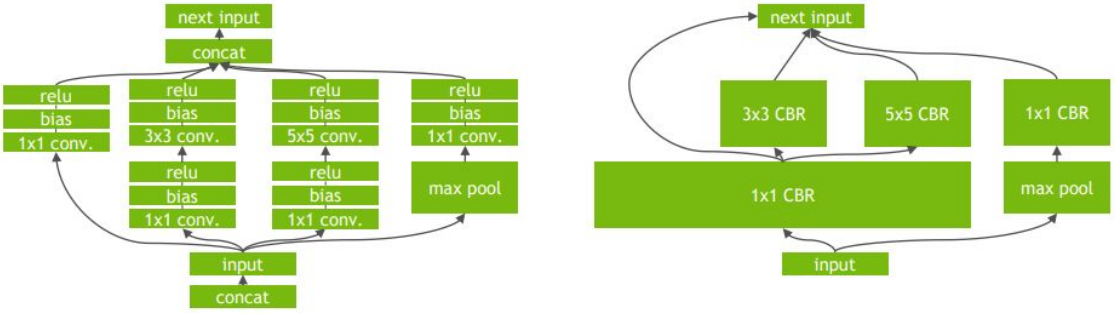


Figura 2.18: Otimização de Camadas (Vanholder, 2016). Uso autorizado pelos autores.

Além das otimizações na arquitetura é possível otimizar os pesos aprendidos pelo modelo durante o treinamento. Dado a quantidade de poder computacional para as inferências, é possível diminuir as representações numéricas de 32 para até 8 bits impactando no tempo para o ciclo de busca na memória. É desconhecido o impacto matemático dessas otimizações. Da mesma forma é preciso mapear os pesos através de uma quantização linear seguindo a Equação 2.8, onde um o Tensor resultante T é dado por um fator de escala sf por um conjunto de 8 bits t.

$$T = sf * t \tag{2.8}$$

O fator de escala é treinado com um conjunto de dados a fim de refletir na melhor precisão possível. Dessa forma, os pesos do menor ao maior aprendidos em 32 bits são mapeados para -127 a +127. Porém levando em consideração do menor ao maior peso acarretou em perdas significativas de precisão. Então é fixado um limiar (T) para que os demais pesos sejam arredondados para -127 ou

+128 de acordo com seu valor. A Figura 2.19 ilustra esse mapeamento.

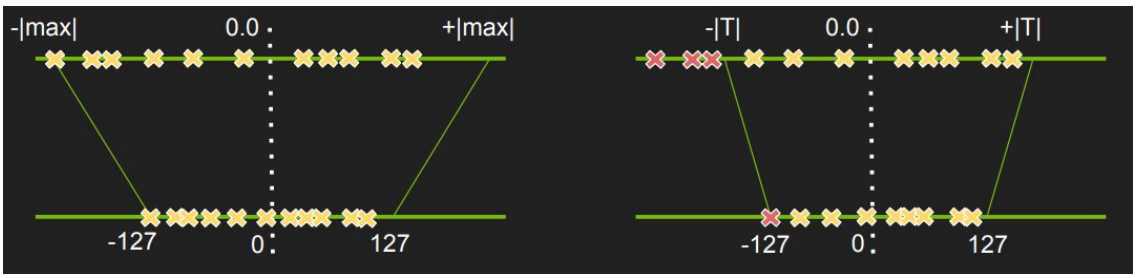


Figura 2.19: Mapeamento de pesos (Vanholder, 2016). Uso autorizado pelos autores.

Com essas otimizações é possível aumentar o *Troughput* e fazer com que esses modelos possam ser utilizados em dispositivos de ponta como um *Jetson Nano* e suas variações. Assim ambas têm um impacto significativo no custo de aplicações que utilizam de técnicas de aprendizagem profunda para inferência.

2.6 Considerações Finais

O entendimento da teoria se mostrou fundamental para o desenvolvimento do projeto. O aprendizado profundo por sua vez, estado da arte em diversas áreas, necessita de um entendimento para ser manipulado e utilizado. A fundamentação se mostra necessária ao realizar testes de validação de arquiteturas, pois a compreensão de conceitos base como o funcionamento de uma convolução permite uma interpretação cada vez melhor dos resultados obtidos. Dessa forma, é possível treinar algoritmos do início com a manipulação de dados ou utilizar abordagens pré-treinadas reconhecendo as metodologias ali utilizadas.

O conhecimento de *Frameworks* de detecção de objetos auxilia na escolha das melhoras abordagens, para isso sabe diferenciar *Two-Stage Detectors* de *One-Stage Detectors* auxilia no balanceamento entre precisão e desempenho. Balanceamento esse que está embasado na compreensão das métricas de avaliação dos mais diversos modelos propostos.

Por fim, em conjunto com o conhecimento dos mais diversos rastreadores de objetos da literatura, bem como as competições em torno do problema enriquece o conjunto de possibilidades. Sendo assim, a utilização de Redes Neurais Convolucionais, da família *One-Stage Detectors* e Rastreadores Online de Múltiplos Objetos foram fundamentos base para o desenvolvimento dessa proposta.

CAPÍTULO

3

Trabalhos Relacionados

Neste capítulos são descritos trabalhos que possuem relação com o tema proposto e serviram como fonte de pesquisa. Os resultados propostos bem como abordagens desses trabalhos serviram de fundamentos para o desenvolvimento do projeto e escrita da proposta de contagem do fluxo de pessoas.

3.1 Estimativa e Contagem de Pessoas

A contagem de pessoas pode ter abordagens distintas, duas delas são por região de interesse (ROI) (Velipasalar et al., 2006; Cai et al., 2014), e linha de interesse (LOI), (Ma and Chan, 2013; Sun et al., 2019). A primeira abordagem citada conta a quantidade de pessoas em uma região específica da imagem de entrada ou video. Com isso é possível por exemplo obter estimativa em uma rua de grande fluxo ou contabilizar multidões. Pode ser baseado na densidade ou no número aproximado de pessoas. A segunda abordagem tem uma dependência mais forte das detecções e de algoritmos de rastreamento, pois o número de pessoas é estimado com base no seu movimento através da linha imaginária, por isso o nome de Linha de Interesse. Assim a primeira abordagem pode ser aplicada em imagens únicas, porém a segunda baseada apenas na Linha de Interesse requer um video pois depende do movimento dos objetos.

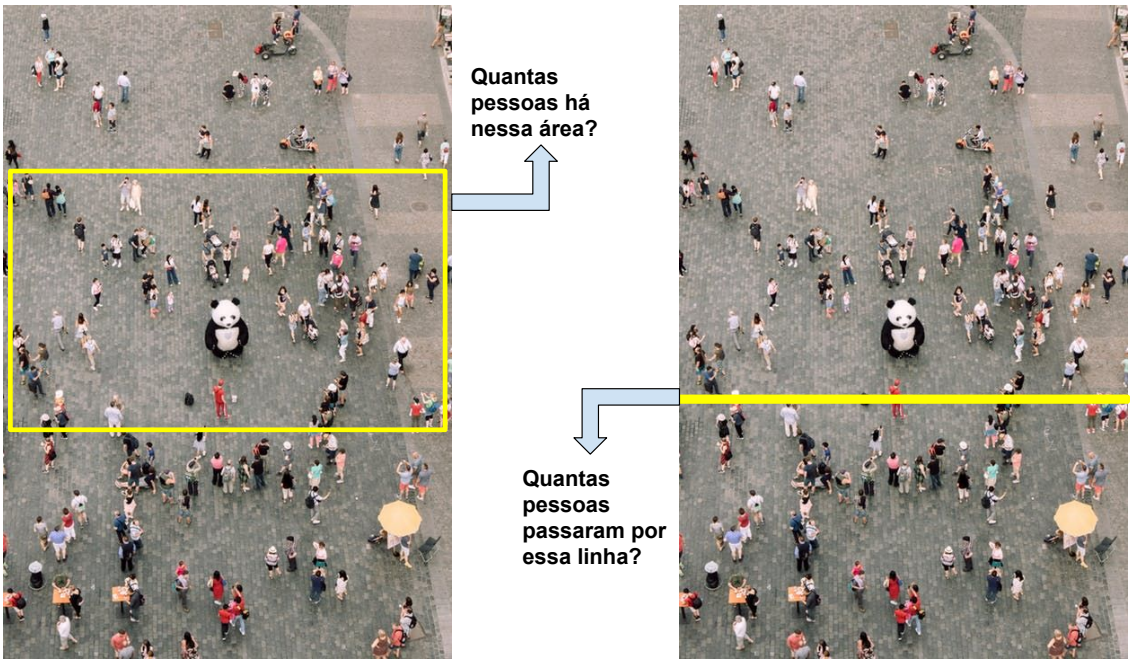


Figura 3.1: Diferença entre ROI e LOI. Imagem sem direitos autorais da internet.

Para contar o número de pessoas em estabelecimentos ou estimar o fluxo, muitas abordagens são baseadas na Engenharia com o uso de sensores ou implementações específicas de hardware. Um exemplo é a radiação infravermelha passiva (PIR) implementada via hardware para detectar a presença e contar pessoas (Sruthi, 2019). As pessoas analisadas na pesquisa, ao passarem pela área de incidência do PIR implicam no início da contagem e na gravação do momento em que passam por esses sensores. Um dos problemas mencionado na pesquisa é a oclusão de pessoas sendo que a emissão de PIR não atravessa o corpo para contabilizar a pessoa oclusa. Quando duas ou mais passam ao mesmo tempo sob a área de incidência, são consideradas oclusas pessoas que não são atingidas pelo PIR. Outro problema está relacionado a pessoas com vestimentas ou carregando objetos que podem refletir a emissão da radiação.

A proposta RetailNet (Nogueira et al., 2019) utiliza técnicas de visão computacional para segmentar pessoas em imagens RGB, adquiridas por câmeras de vigilância. O objetivo da segmentação é contar pessoas para elaborar um mapa de zonas quentes. A viabilidade do trabalho deu-se através da adição de um canal denominado P aos canais canais RGB, canal este que é adicionado atra-

vés de um pré processamento de imagens com subtração de fundo antes de ser fornecido a CNN para o treinamento. Este canal extra tem o intuito de indicar a presença de pessoas por meio de uma imagem agora chamada de RGBP. Por mais que os resultados tenham sido satisfatórios, foram identificados problemas, dentre eles identificação de funcionários e vendedores dos estabelecimentos, que foram contabilizados nos resultados da contagem.

Adicionalmente há várias abordagens que estudam a estimativa de multidões, bem como entender como elas se comportam. O número de pesquisas que têm o objetivo de estudo relacionado a pessoas em multidões tem crescido, trazendo atenção de pesquisadores, dada as diversas aplicabilidades práticas comportadas por essa área (Sindagi and Patel, 2018). Podem ocorrer modificações aplicadas diretamente nas arquiteturas de redes neurais profundas utilizadas (Zhao et al., 2016), a fim de automatizar o processo de contagem. Estas modificações além da contagem, permitem estimar densidade e também a velocidade do fluxo de pessoas de uma só vez como resultado da classificação. O processo pode ser realizado através da seleção de pares de imagens, que são analisadas através da seleção de pixeis, para obtenção dos resultados.

Em outro trabalho (Yam et al., 2011) são implementados detectores de objetos para trabalhar em cooperação com algoritmos de rastreamento, o primeiro fornece as classificações a serem utilizadas por algoritmos de rastreio nos próximos quadros. A área de contagem pode variar de acordo com a proposta do trabalho, por exemplo, uma área disposta a 45 graus com contagem bidirecional, tanto fluxo de saída como de entrada.

3.2 Contagem de Veículos

Tanto pessoas como veículos compõe o tráfego em grandes cidades. O entendimento do fluxo de veículos é essencial para o monitoramento do tráfego a partir de câmera de segurança. A partir dessa hipótese surgem abordagens que buscam aplicar técnicas antes aplicadas à pessoas também aplicadas ao tráfego de veículos. Essas técnicas aplicam modificações em CNNs, a fim de adaptar a maneira com que detectam os objetos ao implementar detecção de vários veículos ao invés de somente um. Essa detecção auxilia o estudo da densidade nas vias como forma de estimativa de fluxo (Zhang et al., 2017b).

Ainda no âmbito de câmeras de segurança, há limitações nessa fonte de dados relacionadas aos seus parâmetros, como a baixa taxa de quadros, grande ângulo de visão, oclusão e baixa qualidade de imagem. Pensando nisso, pesquisas buscam contornar esses problemas ao analisar conjuntos de pixeis ao invés da imagem como um todo. Um exemplo de proposta é o uso de *Long Short Term Memory Networks* (LSTM) para predição temporal no controle de fluxo e movimento, bem como, *Fully Conected Networks* (FCN) para a predição e detecção de objetos usando pixels como referência para uma melhor precisão (Zhang et al., 2017a). Para isso foram utilizados pilhas de redes totalmente conectadas para predição, com LSTMs para sanar problemas com baixa taxa de quadros.

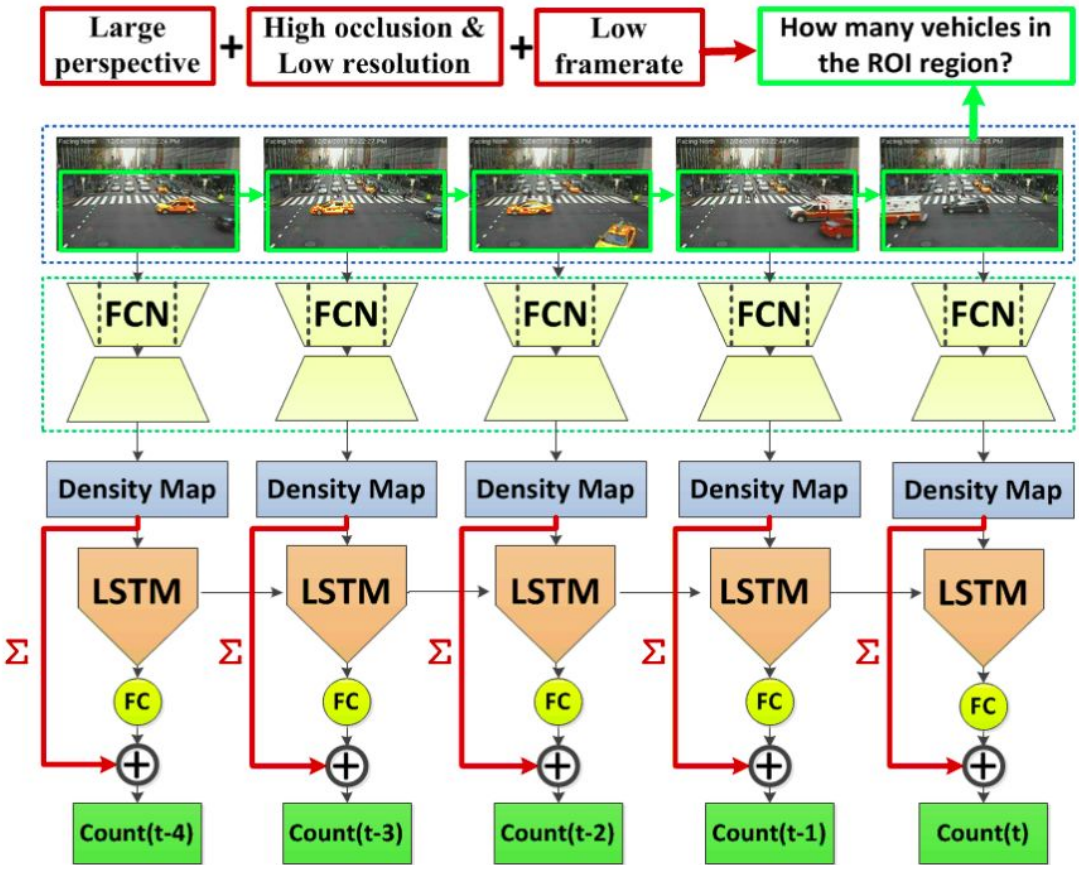


Figura 3.2: Arquitetura da proposta FCN-rLSTM (Zhang et al., 2017a). Uso autorizado pelos autores.

Outras abordagens utilizam de imagens aéreas para contagem de veículos, empregando algoritmos de processamento em tempo real. Por exemplo Polidoro et al. (2019), que utiliza versões menores da YOLOv3 (Redmon and Farhadi, 2018) como a Tiny-Yolov3 para a contagem de veículos em áreas com grande trafego de

veículos. A abordagem se mostrou promissora atingindo 87.24% de acerto na contagem.

3.3 Reconhecimento e Detecção Facial

Na área de detecção facial, estudos recentes buscam incrementar a precisão na detecção de faces em imagens paralelamente a diminuição da necessidade de processamento. Muitas implementações de detecção facial são leves a ponto permitirem o processamento em dispositivos móveis em tempo viável (Deng et al., 2019b). Algumas implementações de reconhecimento facial têm sido muito precisas, ao trabalhar adaptação da função de perda da rede. Uma dessas implementações utiliza Perda de margem angular aditiva (ArcFace) como forma de diferenciação de faces (Deng et al., 2019a), apresentando resultados como 99,53% de precisão em conjuntos como LFW (Huang et al., 2007) e 95,56% no conjunto CFP-FP (Sengupta et al., 2016).

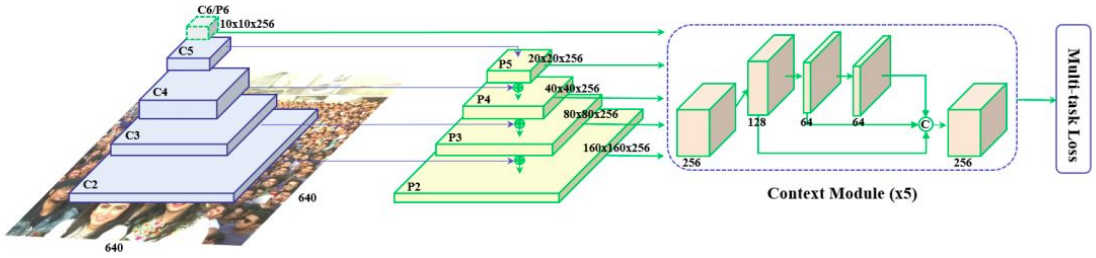


Figura 3.3: RetinaFace com FPN e *Multi-Task Loss* (Deng et al., 2019a). Uso autorizado pelos autores

3.4 Algoritmos de aprendizado profundo com baixo custo computacional

Com o intuito de tornar aplicações rentáveis algumas abordagens buscam diminuir o custo computacional. A instância EC2 mais barata para computação com GPUS, nos serviços da amazon sob demanda, custa 0.526 dólares a hora. Portanto a utilização de algoritmos leves são de grande importância para redução do custo computacional e consequentemente o custo de produtos que dependam destes algoritmos.

A PeleeNet proposta por Wang et al. (2018), tem uma variação da DenseNetA camada densa de dois sentidos, em um sentido utiliza uma convolução 3x3 e no outro duas convoluções 3x3 empilhadas para melhorar a detecção de objetos maiores. A adição do *Stem Block* antes da primeira camada densa tem o foco em aumentar a precisão do modelo cuidando para que o número de canais não cresça de forma descontrolada através de operações de *Max Pooling* de um lado e convoluções do outro. Esse bloco tem como inspiração os módulos Inception-v4 (Szegedy et al., 2017). Outro ponto interessante é o número dinâmico de canais que varia de acordo com o tamanho de entrada do modelo, dessa forma, diferente da DenseNet a proposta busca controlar o crescimento do número de canais e com isso diminuir o custo computacional em até 28.5%. Por último a fim de evitar perda de informações durante o processo, nos blocos de transição (Convolução + Pooling), o número de canais é igual entrada e saída. Com isso a PeleeNet conseguiu atingir 240 quadros por segundo em imagens 224x224 em uma NVIDIA TX2.

Stage		Layer	Output Shape
Input			224 x 224 x 3
Stage 0	Stem Block		56 x 56 x 32
Stage 1	Dense Block	DenseLayer **x 3**	28 x 28 x 128
	Transition Layer	1 x 1 conv, stride 1	
		2 x 2 average pool, stride 2	
Stage 2	Dense Block	DenseLayer **x 4**	14 x 14 x 256
	Transition Layer	1 x 1 conv, stride 1	
		2 x 2 average pool, stride 2	
Stage 3	Dense Block	DenseLayer **x 8**	7 x 7 x 512
	Transition Layer	1 x 1 conv, stride 1	
		2 x 2 average pool, stride 2	
Stage 4	Dense Block	DenseLayer **x 6**	7 x 7 x 704
	Transition Layer	1 x 1 conv, stride 1	
Classification Layer		7 x 7 global average pool	1 x 1 x 704
		1000D fully-connecte,softmax	

Figura 3.4: Arquitetura PeleeNet. Fonte: (Wang et al., 2018). Uso autorizado pelos autores.

Por outro lado a proposta da ThunderNet (Qin et al., 2019) tem como objetivo

otimizar *Two-Stage Detectors* para que estes tenham um custo computacional próximo a *One-Stage Detectors*. Dessa maneira modifica o backbone, que extrai características, com uma proposta chamada SNet e para a parte de deteção é empregada a mesma técnica da *Light-Head RCNN* com algumas modificações. Como o objetivo é a otimização de *Two-Stage Detectors*, há também modificações aplicadas à RPN. Dessa forma, após as características serem extraídas pela SNet os mapas de características passam por um módulo de aprimoramento de contexto (CEM) para que sejam extraídos mapas de características de várias escalas. Como a arquitetura busca ser leve e portanto lidar com imagens pequenas, isso acaba por dificultar as propostas de regiões da RPN sendo difícil diferenciar fundo do que realmente é interessante para o problema. Pensando nisso o é introduzida outra técnica chamada Modelo de atenção espacial, que realiza a multiplicação do mapa de característica resultante do CEM com as regiões propostas pela RPN. Para que ambos os mapas tenham as mesmas dimensões é aplicada uma convolução 1x1 no mapa de características da RPN que torna a operação menos custosa e então esse resultado é utilizado por uma sigmoid antes de ser multiplicado. Por fim é realizado um *RoI Align* e então uma camada totalmente conectada utiliza as informações para predizer a classificação e localização dos objetos. Dessa forma a SNet49, atingiu 24.1 quadros por segundo em um processador ARM Snapdragon 845 em dispositivos móveis e 267 quadros por segundo em uma NVidia 1080Ti.

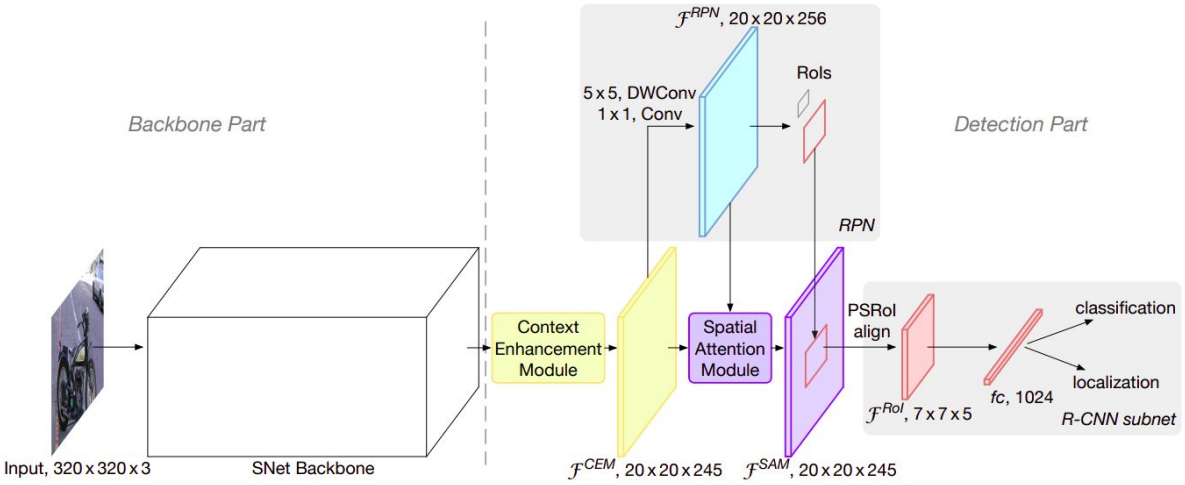


Figura 3.5: Arquitetura ThunderNet (Qin et al., 2019). Uso autorizado pelos autores.

Algumas redes buscam arquiteturas com baixo número de parametros e que tendem a ser rápidas em dispositivos de ponta (He et al., 2019; Yoo et al., 2019). A EXTD de Yoo et al. (2019) tem como principal uma arquitetura simples que faz o uso de *Skip Connections*, blocos residuais e *Upsample Blocks*. Nessa arquitetura é proposta a ideia de recorrência ao *Backbone* para reaproveitar os mapas de características gerados em diferentes resoluções. Dessa forma cada mapa de características é somado a saída da primeira extração de características via *Skip Connections*, essa informação é processada através de blocos residuais e novamente utilizada para gerar mapas de características em várias resoluções. Paralelamente a soma com a *Skip Connection*, é feito *Upsample* para aumentar o número de informações e que no próximo passo são somadas com o mapa de característica anterior ao pipeline. Assim a extração de características em várias escalas funciona de forma semelhante a SSD e a parte de *Upsample* com a soma dos mapas de características vem de características presentes em *Feature Piramid Networks* (FPN). Os autores reportam 0.063 Milhões de parâmetros com os mapas de características com 32 canais.

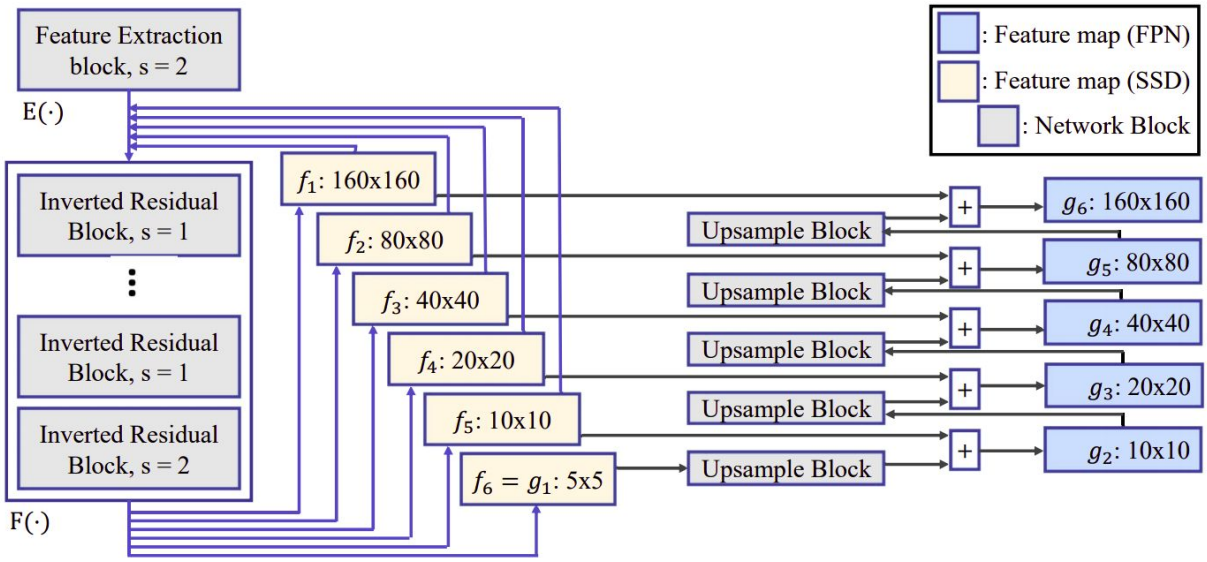


Figura 3.6: Arquitetura EXTD (Yoo et al., 2019). Uso autorizado pelos autores.

Outros algoritmos que valem a pena ser mencionados são a EfficientDet e a Yolov5. A Yolo vem sendo utilizada desde sua primeira versão para *deploys*, pois possui um balanço entre precisão e velocidade devido as várias versões que disponibiliza da Yolov5s até Yolov5x. A sua arquitetura é composta de 3 partes, sendo a primeira o Backbone com uma CSPDarknet, a segunda parte da *Neck*

por uma PANet e por ultimo a *Head* pela camada Yolo, que tem como saída uma classe, confiança, localização com coordenadas e tamanho da *Bouding Box*. O funcionamento consiste em uma entrada que é alimentada para a CSPDarknet afim de extrair características, posteriormente as características são concatenadas pela PANet e por ultimo ocorre a predição pela camada Yolo. Sua arquitetura está ilustrada na Figura 3.7.

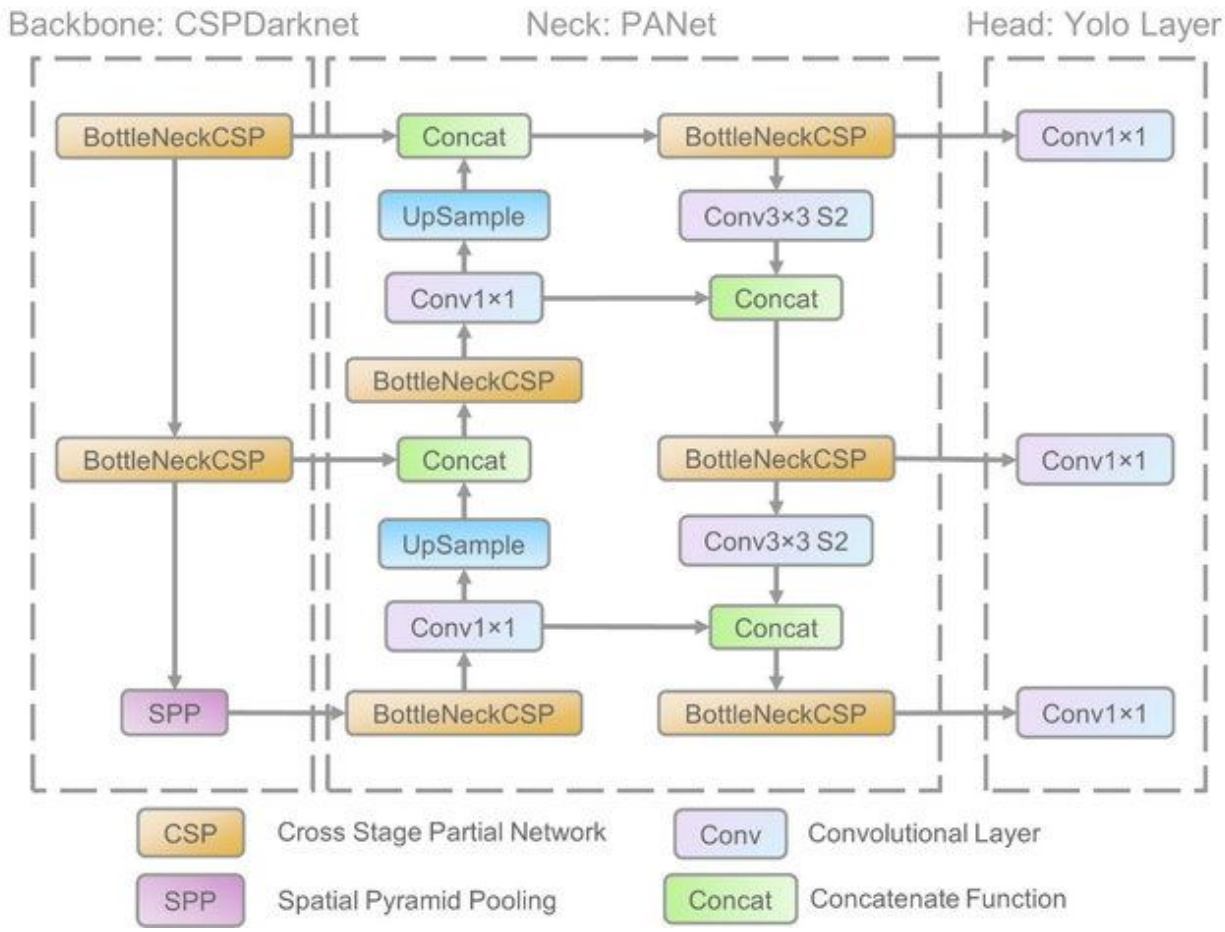


Figura 3.7: Arquitetura Yolov5 (Jocher et al., 2021). Licença de Imagem CC BY 4.0

Do outro lado a EfficientDet também é composta por três partes, um *Backbone* chamado de EfficientNet, *Neck* com Bi-FPN e por ultimo a *Head* composta por convoluções como exemplificado na Figura 3.8. Similar a como acontece na Yolov5, dada uma entrada, primeiro é extraído as características pela EfficientNet e posteriormente concatenadas pelas Bi-FPNs. Por ultimo a *Head* tem como saída as detecções com a classe, confiança, localização e tamanho da *Bouding*

Box.

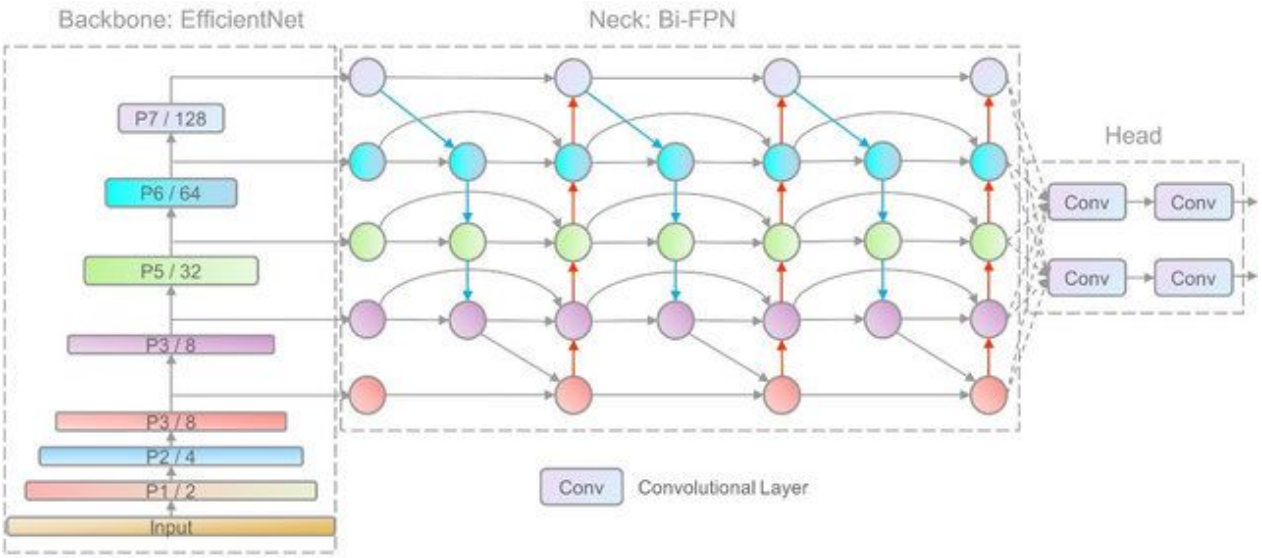


Figura 3.8: Arquitetura EfficientDet (Tan et al., 2020). Licença de Imagem CC BY 4.0

3.5 Considerações Finais

A partir do estudos da teoria e suas aplicações, foram observadas as abordagens e para atingir os objetivos deste trabalho, a abordagem utilizando uma linha de interesse (LOI) é a mais adequada ao trabalho. As abordagens de contagem de pessoas serviram de base para descoberta de métodos, pontos fortes e fracos. A estimativa de multidões tem por objetivo enriquecer e listar abordagens para estudo de possíveis adaptações. A contagem de veículos tem a contagem como tópico de interesse e seus estudos trazem ideias que podem ser adotadas também para outras classes. Por fim a busca por algoritmos de reconhecimento facial de alto desempenho teve resultados, porém há a necessidade de avaliar esses algoritmos no mundo real.

CAPÍTULO

4

Proposta

Neste capítulo é apresentada a proposta para avaliação dos objetivos específicos a partir de experimentos baseados na literatura e de algoritmos que possibilitaram o desenvolvimento da pesquisa. São propostos experimentos que vão desde a coleta das imagens, contagem, reconhecimento até a otimização dos algoritmos.

4.1 Visão geral da proposta

A proposta consiste em dois problemas: (1) contagem de pessoas e (2) identificação dos funcionários. Na contagem de pessoas é feito o processamento dos vídeos para detecção das pessoas e rastreio até entrarem na fachada da loja. Na identificação são separados os funcionários dos clientes, ao entrar na loja é verificado através do *upsample* da face se esse potencial cliente é de fato um cliente ou funcionário da loja.

Para a resolução dos dois problemas em questão foi elaborado o fluxograma da Figura 4.1 para ilustrar o fluxo de dados da proposta, bem como referência para compreensão dos experimentos apresentados no Capítulo 5. Assim, a Seção 4.2 está relacionada ao item (a) do Fluxograma, a Seção 4.3 ao item (b) e (c) e por fim a Seção 4.3.1 descreve os passos (d) e (e).

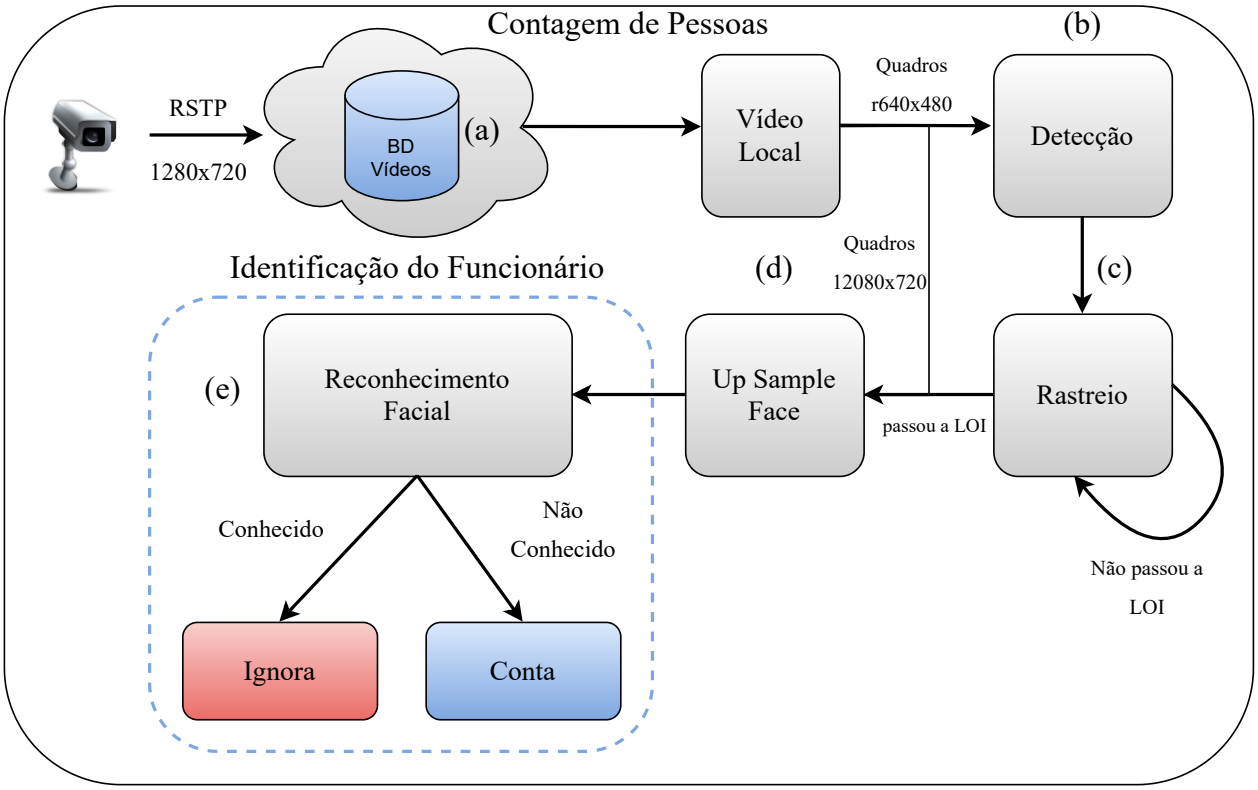


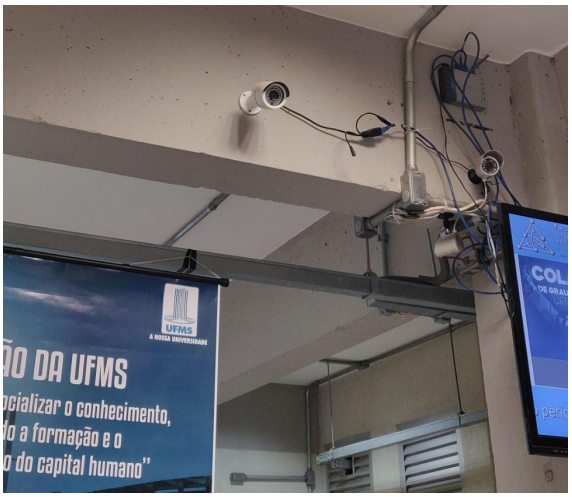
Figura 4.1: Fluxograma da proposta.

Para a contagem inicialmente a ideia era utilizar somente a face para o desenvolvimento de todo o sistema. Entretanto ao utilizar somente a face foram identificados o seguintes problemas: rotação das faces, angulação da face e oclusão da face. Devido aos resultados insatisfatórios com a utilização somente da face, no décimo terceiro mês do andamento deste trabalho a proposta foi alterada. Foi reestruturada para contemplar detecção do corpo inteiro das pessoas para detecção, contagem e posteriormente o *upsample* da face das pessoas detectadas, a fim de identificar os funcionários/colaboradores da loja.

4.2 Construção dos bancos de vídeos

A primeira coleta de dados aconteceu na Faculdade de Computação, FACOM, na UFMS em Campo Grande no Mato Grosso do Sul. Para isso foi instalada uma câmera marca genérica, HD, com gravação a 25 quadros por segundo e sensor de movimento. A mesma foi posicionada em direção a porta principal do prédio, distante de 6,30 metros e angulação de 22 graus conforme Figura 4.2a. Para

gravação foi utilizado um notebook HP Intel Core i3, 4 Gigabytes de memória RAM, 500 Gigabytes de HD e o software open source de gravação ZoneMinder para obtenção das imagens via protocolo RTSP (Schulzrinne et al., 1998). Os vídeos capturados através do acionamento do sensor de movimento possuem em média 9 quadros por segundo e quando gravados de maneira contínua foram armazenados vídeos com até 25 quadros por segundo, podendo ocorrer oscilações dependendo da rede variando a taxa de gravação. Um exemplo de quadro que demonstra a visão obtida por essa configuração e posicionamento da câmera, está ilustrado na Figura 4.2b



(a) Posicionamento da câmera na FACOM.

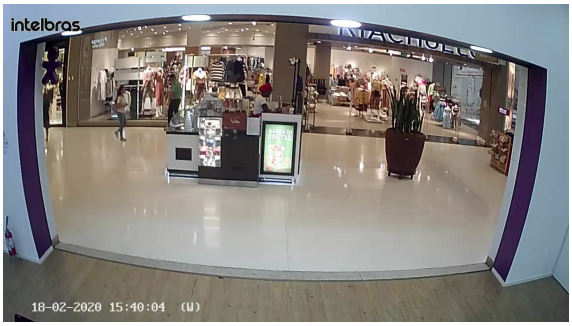
(b) Exemplo de imagem obtida pela gravação na FACOM. A iluminação interna tem interferência na imagem.

Figura 4.2: Configuração de ambiente na FACOM.

Na segunda coleta foram obtidos vídeos de uma loja localizada no Shopping Norte Sul na cidade de Campo Grande no Mato Grosso do Sul. A coleta nesse estabelecimento tinha por objetivo refletir um ambiente real de funcionamento da aplicação proposta. Para a gravação foi utilizada câmera modelo Intelbras

IC3, com gravações em HD (1280 x 720), de maneira contínua com 30 quadros por segundo podendo variar até 15 quadros dependendo da rede. A máquina que realiza a captura e armazenamento dos vídeos é um notebook HP Intel Core i3, 4 Gigabytes de memória RAM, 500 Gigabytes de HD e o software open source de gravação ZoneMinder. Foram gravadas um total de 207 horas e 1 minuto, resultando em 117,98 Gigabytes de dados de vídeos na FACOM. Já na loja do Shopping Norte Sul foram 58 horas e 44 minutos de gravações, totalizando 43,66 Gigabytes de dados.

(a) Posicionamento da Câmera.



(b) Visão da Câmera.

Figura 4.3: Configuração do ambiente de gravação em cenário real.

Durante a coleta de vídeos da loja em ambiente real, houve a tentativa de usar o próprio circuito interno de televisão (CCTV) como fonte de armazenamento dos vídeos, porém as câmeras possuíam a qualidade original de 240p. Devido a baixa qualidade, há menos informações nas imagens o que impossibilita o reconhecimento facial.

Um dos objetivos principais dessa proposta é viabilidade econômica da implantação do algoritmo de contagem. Uma das otimizações feitas é recortar da imagem somente a área relativa a pessoa que passou na LOI. Dessa forma a de-

tecção facial ocorre em uma imagem menor e não na imagem toda. Durante o processamento para detecção de pessoas, as imagens são redimensionadas para qualidades menores para que o processamento seja eficiente. O redimensionamento do quadro para detecção afeta o reconhecimento facial, pois a face quando detectada possui menor qualidade assim pode não ser reconhecida, não detectada ou reconhecida erroneamente.

4.3 Contagem de pessoas

Após a construção do Banco de Vídeos (a) e seguindo o fluxograma, a detecção ocorre no passo (b) da Figura 4.1 através da Yolov5. Uma vez que o objeto é detectado, acima de 80% de confiança, o objeto é submetido ao algoritmo de rastreio em (c). Os objetos testados foram a Face e Pessoa. O rastreamento que inicialmente seria feito pelo somente centróide do objeto e a distância euclidiana entre os centróides, porém passou a ser substituído pelo algoritmo DeepSort (Wojke et al., 2017) explicado na Seção 2.4, com os parâmetros padrões.

Ainda com relação ao rastreamento, a fim de evitar o uso do classificador em todos os quadros, e tornar a aplicação menos custosa computacionalmente foi implementada uma redução na quantidade de quadros processados pelo algoritmo de detecção. Essa redução foi implementada tendo como parâmetro inicial a taxa de quadros do vídeo FPS, posteriormente a detecção é aplicada a cada n quadros com $n = FPS/2$, nos quadros entre uma detecção e outra a predição fica por conta do algoritmo de rastreamento.

Cada rastreador possui um histórico de posições do objeto nos quadros anteriores considerando um conjunto de quadros. Com base na diferença entre a média desse histórico de posições e a posição atual, é possível obter uma noção de direção do movimento do objeto. Quando positiva o objeto se move em direção ao estabelecimento, caso contrário está em direção à área externa.

4.3.1 Upsample

Em cada quadro foi analisada a posição atual do objeto. Ao passar pela linha de interesse (LOI), é verificado a direção do movimento e se já houve a contabilização desse objeto. Por conseguinte a face em maior resolução é extraída do quadro original no item (e), armazenado anteriormente no item (a), por meio da

correlação das medidas das *Bouding Boxes* para seu tamanho original. O *upsample* da face possibilitou um reconhecimento facial mais em (e). Este último é utilizado para distinguir funcionários de possíveis clientes.

Para contornar o problema do redimensionamento do quadro para detecção e o efeito desse redimensionamento no reconhecimento facial é armazenado o quadro original. A face quando detectada, após a detecção da pessoa, é reescalada(*upsampled*) para o seu tamanho original e consequentemente a qualidade original. As faces eram reconhecidas através da extração de *embeddings* faciais com o algoritmo pré treinado do ArcFace (Deng et al., 2019a). Dessa forma não houve a necessidade de treinamento para reconhecimento específico de uma face uma vez que são comparados os *embeddings* preditos com o banco de faces conhecidas.

Como passo final para uma contagem bem sucedida é necessário que condições sejam verdadeiras. A primeira é que direção do movimento seja em direção ao estabelecimento, o objeto não deve ter sido contabilizado anteriormente verificado através de um status atrelado ao rastreador, então se o objeto passar a LOI é verificado se este está no conjunto de funcionários. O contador será incrementado se todos os passos descritos forem verdadeiros. Para que as verificações ocorressem da melhor maneira possível, foi necessário avaliar o ambiente de contagem e definir limite de funcionamento do algoritmo.

CAPÍTULO

5

Resultados e Discussão

As seções desse capítulo estão descritas de acordo com o Fluxograma elaborado no capítulo 4. Elas descrevem os testes realizados bem como os resultados obtidos em cada passo até a elaboração final do algoritmo. Cada seção busca responder uma pergunta relacionada ao trabalho.

5.1 Configuração de ambiente

Esta seção tem como intuito responder a pergunta: quais condições de ambiente e de configuração de câmera tem impactos positivos na detecção de objetos e reconhecimento facial?

Para este teste foram utilizadas duas câmeras distintas:

Intelbras IC3 Câmera HD 1280x720 com sensor 1/4"HD Progressive Scan CMOS de 3.9mm por 2.9 e uma lente de 2.8mm;

Câmera IP Genérica Câmera HD 1280x720 genérica IP. Não foram encontradas informações da lente e do sensor.

A fim de extrair os melhores resultados para o algoritmo de detecção facial com o fluxograma anterior, foi realizado um experimento relacionado ao posicionamento da câmera e as detecções com base nesses posicionamentos. Para essa

Figura 5.1: Câmera Intelbras IC3.

Figura 5.2: Câmera IP Genérica.

avaliação foram elencados parâmetros como: distância da câmera do ponto de interesse, angulação e resolução da câmera. Esse primeiro experimento utilizou os tamanhos das detecções para a avaliação. Como a proposta do trabalho é usar câmeras de vigilância para reconhecimento e contagem de pessoas, informações dos limites de detecção no mundo real se tornaram fundamentais para o desenvolvimento do trabalho. A maneira encontrada para avaliar os limites, foi tendo como base uma pessoa parada em um local específico com o olhar em direção ao horizonte. A câmera para realização dos testes foi posicionada a 3,05 metros de altura no laboratório de inteligência artificial (LIA). Essa altura corresponde aproximadamente a altura dos locais dos quais houve gravação de vídeos. Sendo assim, a angulação da câmera foi testada até um limiar em que foi presenciado perda de 100% nas detecções da face da pessoa em questão. No primeiro momento houve variação do ângulo até o limite angular, então a câmera foi reposicionada mais próxima a pessoa teste. Sendo assim as distâncias levadas em

consideração no teste foram de 2, 4 e 6 metros tendo como referência a face da pessoa. Não houve variação além de 6 metros pois houve perda de detecção da face em média de 80% dos quadros em 6 metros. As medidas podem ser observadas na Tabela 5.1. Para extração das medidas foi utilizada a RetinaFace pré treinada tendo MobileNetV1 como *Backbone*. As imagens de entrada eram de resolução HD 1280x720, o tamanho da face recortada é aproximado devido a oscilações nas detecções e alterações dependendo do modelo, porém com o tamanho fixo de imagem.

Distância	**Ângulo**	**Total**	**Detectados**	**Perdidos**	**%**	***Face Crop***
2 metros	15	277	277	0	0%	∼45x60
2 metros	30	254	254	0	0%	∼45x60
2 metros	45	250	250	0	0%	∼45x60
2 metros	57	221	221	0	0%	∼45x60
4 metros	0	350	313	37	10.5%	∼28x32
4 metros	15	345	345	0	0%	∼28x32
4 metros	30	350	313	37	10.5%	∼28x32
6 metros	0	251	55	196	78%	∼10x12
6 metros	15	259	67	192	75%	∼10x12
6 metros	30	249	8	241	96.7%	∼10x12

Tabela 5.1: Tabela de medições e perda

A partir dos resultados da Tabela 5.1, onde as colunas Totais, Detectados e Perdidos se referem a quantidade de quadros. Observou-se que quanto maior a distância da câmera em relação a face, menor será o tamanho da face detectada. A perda de detecções é menor em distâncias como 2 e 4 metros.

O tamanho da box de 56x56 é utilizado neste trabalho, pois não há um consenso quanto a melhor resolução porém há um consenso de que o reconhecimento tem perda abaixo de 32x32 fazendo com que o algoritmo tenha taxas de acerto de no mínimo de 17,5% a mais com resoluções maiores (Lui et al., 2009). Quando a resolução é reduzida para menos de 32x32 as taxas de erro no reconhecimento tendem a aumentar (Boom et al., 2006) drasticamente, enquanto acima desta resolução as taxas de erro se mantêm constantes.

5.2 Resolução das imagens

No intuito de aproveitar o máximo do desempenho sem perder precisão da detecção, nessa seção será respondida a seguinte pergunta: Quanto a resolução das imagens do *setup* do ambiente afeta na detecção dos objetos?

O algoritmo escolhido para detecção dos objetos foi a Yolov5, por ter um balanceamento em questão de velocidade de processamento e mAP. Para uma investigação mais detalhada durante o mestrado foram feitos experimentos adicionais para avaliação do Yolov5s. Os experimentos realizados e descritos a seguir:

Yolov5 Variações Avaliação das variações da Yolov5 e comparativo com algoritmos estado da arte para definição da melhor arquitetura em termos de performance e assertividade;

Yolov5 Resoluções Estudo do impacto da resolução das imagens na assertividade dos algoritmos, bem como no desempenho.

Antes do comparativo de pessoa, face e o efeito desses objetos na contagem houve a necessidade de encontrar um algoritmo na literatura que viesse a oferecer uma troca considerável entre desempenho e precisão para a validação da resolução das imagens. Portanto, para a tarefa foram levados em consideração o processamento do algoritmo em milissegundos por imagens (ms/img) e o mAP dos algoritmos. Dessa forma, a literatura mostra vários algoritmos muito precisos como Dai et al. (2021), (Zhou et al., 2021), Ghiasi et al. (2021) e Wang et al. (2021). Por mais precisos que sejam esses algoritmos precisam sacrificar o custo computacional em busca de precisões mais altas. Dessa forma algoritmos como YoloV4(Bochkovskiy et al., 2020), YoloV5 (Jocher et al., 2021) e EfficientDet (Tan et al., 2020) possuem variações das quais possibilitam modificar algumas camadas para que o processamento seja mais rápido em versões menores.

Assim observando os resultados reportados pelos autores na literatura (Wang et al., 2021) e na Tabela 5.2, a YoloV5 em sua versão *small(s)* foi utilizada. Na tabela é possível observar os dados reportados no dataset COCO com 80 classes, usando a métrica padrão de mAP@0.5 no conjunto de validação. O conjunto de testes do COCO não é disponibilizado para o público. Todos os algoritmos foram testados em uma Tesla V100 da Nvidia com *batch=1*.

A Yolov5s obteve bom desempenho ao balancear precisão e velocidade de processamento. A arquitetura da EfficientDet possui o preceito de ser escalável, então quanto maior a versão maior a resolução de imagem que é utilizada para inferência, como a versão mais básica D0 já possui um tempo de processamento maior que a Yolov5s não foram reportados resultados para as arquiteturas D3 até a D7, e o D1 na tabela serve como comparativo dessa afirmação.

Modelo	**Tamanho**	**mAP^{val}**	**Velocidade$^{(ms/img)}$**	**FLOPS**
Yolov5s	640	36.7	**2.0**	17.0B
Yolov5m	640	44.5	2.7	51.3B
Yolov5l	640	48.2	3.8	115.4B
Yolov5x	640	**50.4**	6.1	218.8B
Yolov4	608	45.5	16.1	128.5B
Yolov4-tiny	608	22.6	3.22	**6.9B**
EfficientDet-D0	512	34.3	10.2	2.5B
EfficientDet-D1	640	40.2	13.5	6.1B

Tabela 5.2: Comparativo entre Yolov4/v5 e EfficientDet

O primeiro passo foi treinar o algoritmo em datasets específicos para pessoas (Taiana et al., 2013), (Loy et al., 2019). No treinamento foram utilizadas 300 épocas, com *data agumentation* tendo como resolução de entrada 640x640. Após o treinamento as imagens foram redimensionadas para tamanhos variando de 256 até 1536 com *batch size* de 32, a fim de estudar o efeito da precisão de performance em diferentes resoluções para auxiliar na escolha da melhor resolução. Os resultados estão ilustrados na Figura 5.3. Na figura é possível observar que o maior mAP em ambos os casos está na em torno de 640, que se assemelha a resolução VGA de 640x480. Já relacionado ao tempo de pré processamento foi extraído métricas tanto antes do processamento quanto após o processamento pelo algoritmo.

Assim é possível analisar o tempo para ser aplicado o redimensionamento da imagem (*t_preprocess*), tempo de inferência de fato (*t_inference*) e tempo de filtragem das detecções com algoritmo de Non-maximum Suppression (*t_NMS*). Ao analisar o tempo de inferência é notável que é diretamente proporcional à resolução da imagem e o tempo de *NMS* varia de acordo com a quantidade de detecções na saída da rede. Em imagens maiores a quantidade de Propostas de Regiões pode ser maior que em imagens menores, pois é possível visualizar melhor os objetos e como o dataset do WiderPerson é bem mais populoso que o

dataset do INRIA Person, o primeiro tem bem mais detecções em seu conjunto de dados e consequentemente mais tempo de *NMS*. Após a análise do conjunto de informações nota-se que o tempo é diretamente proporcional à resolução e faz com que a escolha da melhor resolução seja baseada na Precisão(P), Recall(R) e na precisão média(mAP) dos algoritmos. Dessa forma foi definido que o *resize* do quadro para processamento seria para a resolução VGA.

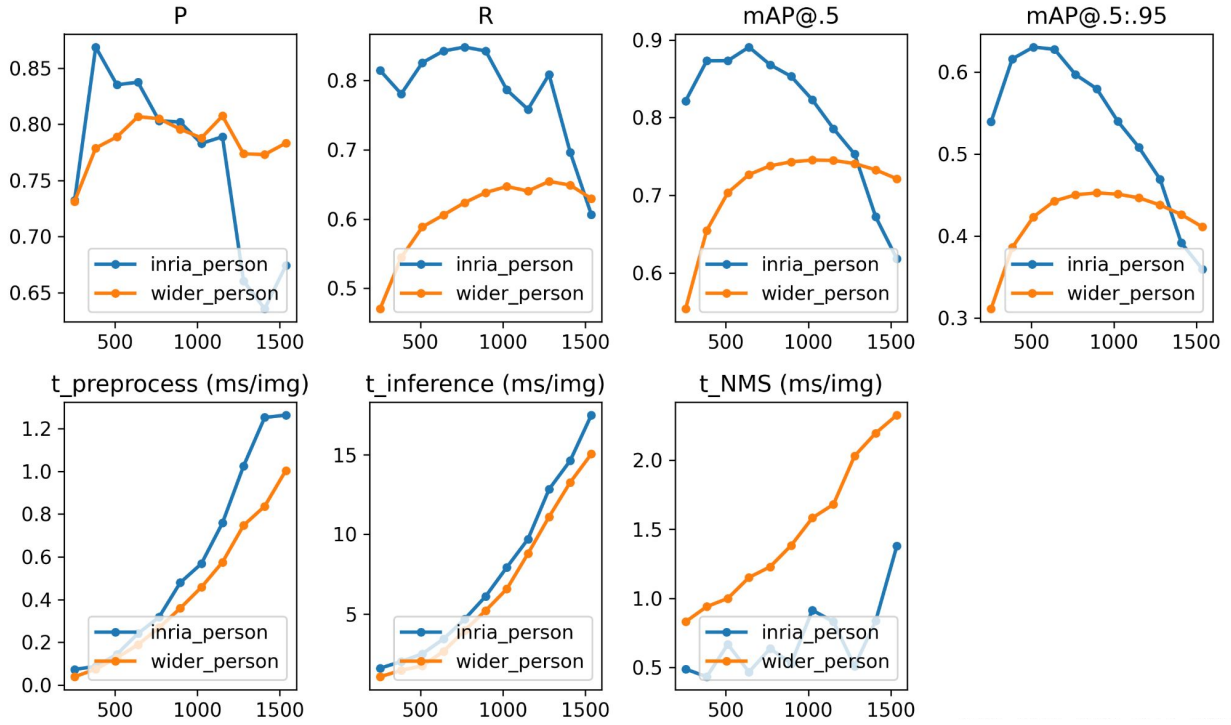


Figura 5.3: Conjunto de métricas extraídas nos datasets WIDER Person e INRIA Person de acordo com Resolução de Imagem de entrada.

5.3 Avaliação do erro de contagem

Nessa seção será respondido o que é melhor, contar utilizando a detecção somente da face ou contar utilizando a detecção do corpo inteiro? Caso a detecção do corpo seja melhor, existe diferença significativa quando comparada com somente a face?

Para responder a estas perguntas foi utilizado algoritmos treinados em datastes específicos para pessoas e pré-treinados em faces. Então a partir de exemplos coletados de ambiente real, com fluxo de pessoas moderado foram obtidos resul-

tados preliminares somente com algoritmo de detecção facial e um menor intervalo de análise de vídeos. No primeiro teste em ambiente real, no dia 03/02/2020 das 19:28 até às 20:28, foi anotado um fluxo real de 20 pessoas. Foram contabilizadas pessoas mesmo que estas tenham entrado na loja duas ou mais vezes, pois em todas as visitas o cliente é um comprador em potencial. Crianças foram ignoradas, por mais que o algoritmo não tenha contabilizado nenhuma. O resultado da contagem real bem como a do algoritmo tendo como objeto a Face está expressado na Figura 5.4.

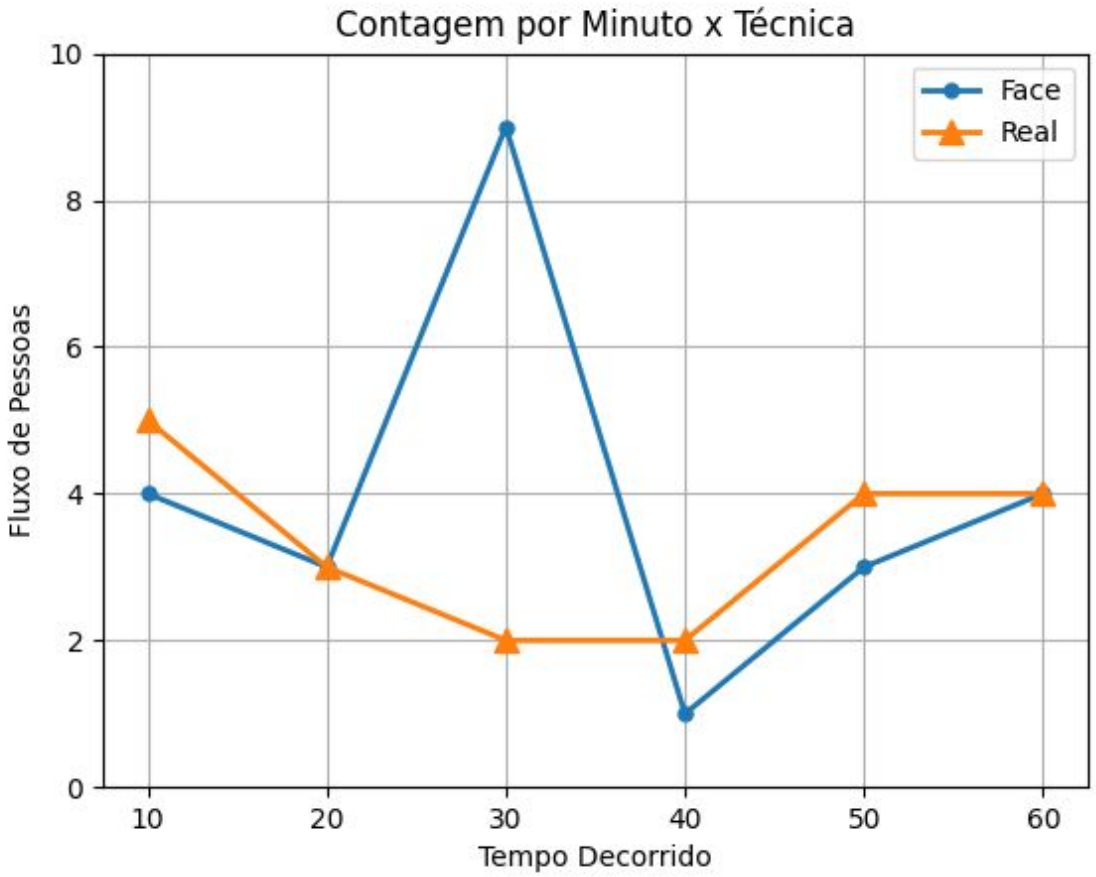


Figura 5.4: Contagem estimada do Algoritmo de Face vs Real

Em um primeiro momento os resultados com a face se mostraram promissores, porém o tempo de análise de 1 hora não foi conclusivo, necessitando uma análise temporal maior. O que chamou atenção foi no minuto 30, em que há uma grande diferença entre a contagem real e o contabilizado pelo algoritmo. Isso pode ocorrer por dois fatores, o primeiro está relacionado a rotatividade da

face pois se a pessoa faz a ação para algum dos lados e depois é detectada novamente, será contabilizada outra vez caso passe a LOI. O segundo fator é o reconhecimento facial, pois o banco de imagens dos funcionários foi obtido por imagens extraídas do próprio vídeo, isso pode ter influenciado no reconhecimento do algoritmo. Por isso a diferença notada entre o real e o algoritmo é devido a funcionários que foram contabilizados várias vezes. Como é uma loja aberta no shopping os funcionários em alguns momentos do dia ficam parados na fachada da loja a fim de cativar clientes a entrarem na loja, assim sua face está em constante rotação. Mas não somente isso, influenciou também pessoas que foram contabilizadas várias vezes por conta da rotatividade da face. Pois como não há detecção por se tratar de faces oclusas será reatribuído novos identificadores a estas faces. Portanto nessa avaliação foi observado que as faces para contagem tinham seu desempenho afetado por alguns aspectos como angulação, tamanho e rotacionamento constante por parte dos clientes e funcionários, isso dificultava o rastreamento e consequentemente a contagem dos clientes.

Após o treinamento de algoritmos para detecção de Pessoas foi realizada uma análise mais extensa com um número maior de vídeos com o objeto Pessoa, com os dados obtidos da avaliação do ambiente a fim de comparar os resultados. Foram processadas 22 horas 56 minutos e 17 segundos, utilizando apenas a face foi obtido um acerto de 62,57%, considerando a pessoa o resultado foi de 105,38% em relação a anotação manual da contagem de pessoas que entraram na loja, significando que foram contabilizadas pelo algoritmo pessoas além do anotado. O percentual acima de 100% pode ser explicado por erros na anotação manual devido ao cansaço do anotador ou a precisão do algoritmo de rastreamento. Os resultados estão dispostos na Tabela 5.3.

Técnica	**Visitas**	**Taxa de Acerto**
Real	179	100%
Pessoa	188	105,38%
Face	112	62,57%

Tabela 5.3: Comparativo das técnicas de contagem

Com os resultados comparativos das visitas foi possível aplicar testes estatísticos nas Técnicas de contagem real, por face e por pessoa. Para isso foi criado um conjunto de dados contendo a técnica de contagem (Pessoa, Face ou Real) e a contagem reportada por aquela técnica por vídeo. Sendo assim estão listadas

120 observações no total, com 2 variáveis de Contagem e Técnica.

Após a montagem do conjunto de dados aplicado um teste Anova de 2 fatores com α de 0.5 e métrica PCC nos conjuntos de contagem, para verificar se houve uma diferença significativa entre as contagens e quais eram mais próximas. Considerando a Hipótese Nula (H_0) em que todos os métodos são iguais, essa pode ser descartada pois P-Valor < α considerando o resultado de P-Valor=0.0003. Os resultados da Anova estão dispostos na Tabela 5.4

	DF	SumSq	MeanSq	F-Value	Pr(>F)
Técnica	2	86.2	43.11	6.056	0.0003

Tabela 5.4: Resultado Anova

Pelo boxplot da Figura 5.5 é possível ver que a técnica de Face possui *Outliers* e sua mediana está bem abaixo das demais técnicas. Dessa forma o desempenho por contagem de Face é imprevisível quando comparado aos demais.

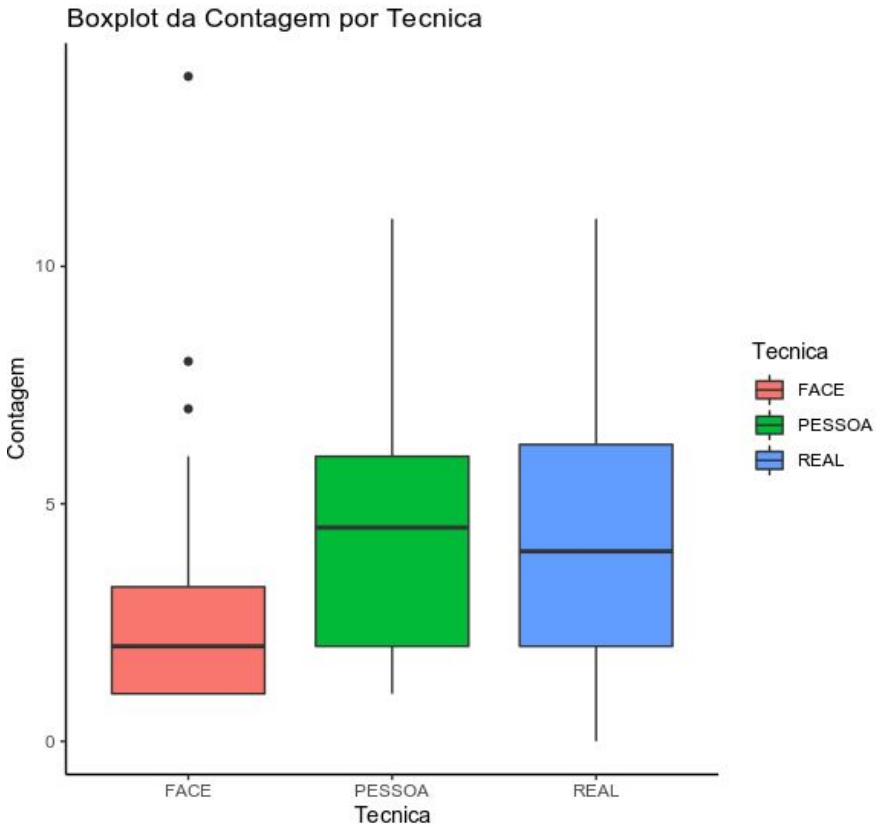


Figura 5.5: Boxplot da PCC por técnica utilizada.

Após o teste foi aplicado um pós teste de Tukey para observar as diferenças entre as técnicas. Com o teste foi notado que a Contagem por Pessoa e Contagem

Real possuem um Desempenho Médio semelhante usando a métrica de *Pearson correlation coefficient* (PCC). Na Figura 5.6 é possível observar que as técnicas possuem semelhança estatística pela barra horizontal. No caso do comparativo *Pessoa-Face* e *Real-Face* ambas possuem suas barras horizontais totalmente à esquerda, mostrando diferença entre o desempenho médio das técnicas.

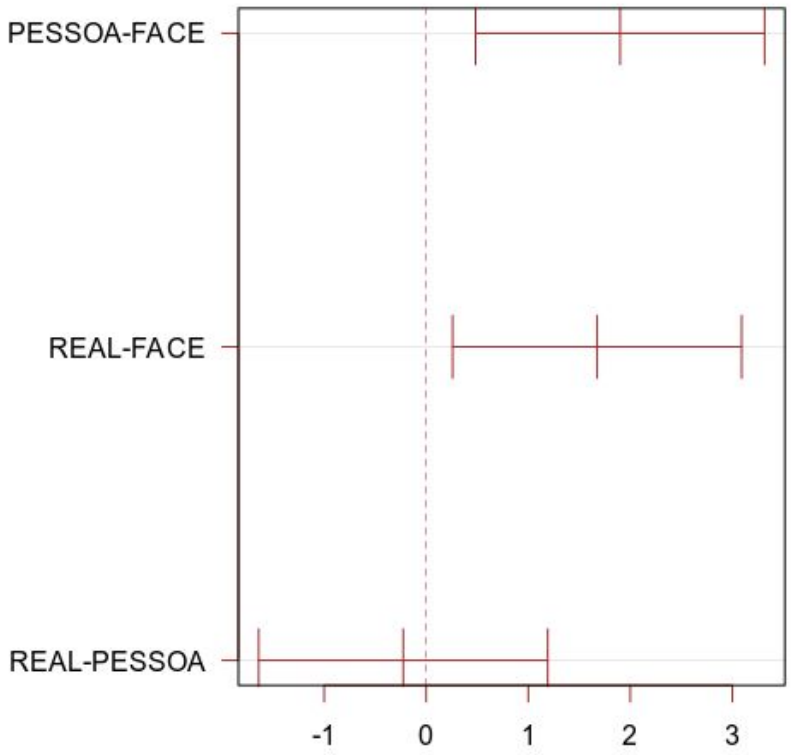


Figura 5.6: Pós teste de Tukey com 95% de nível de confiança

Vale ressaltar que para o problema em questão é interessante identificar técnicas que possuem a menor diferença média possível, ou seja, se assemelham a contagem Real. Assim é possível validar que *Pessoa-Face* e *Real-Face* são significativamente diferentes entre si, dessa forma devem ser estudadas mais a fundo para uma possível melhora em seus comparativos. No comparativo 2 a 2 entre Pessoa e Real foi possível validar que possuem desempenho médio semelhante, ou seja, suas contagens reportaram resultados médios parecidos.

5.4 Upsample da Face

O upsample mostrou-se efetivo melhorando o reconhecimento e também observou-se melhora em duas situações: a primeiro é que quando a face é reconhecida em ambas as resolução sendo que face obtida por meio do *upsample* apresenta confiança maior, a segunda situação é quando a face não seria reconhecida e com a aplicação da técnica essa face passa a ser reconhecida. Ainda, não distante, ocorrem casos em que pessoas são reconhecidas erroneamente e quando há erro em ambas, a com *upsample* possui uma confiança menor em seu resultado.

Figura 5.7: Exemplo de não reconhecimento

Na Figura 5.7 é possível observar que na resolução menor, na parte da direita, não há reconhecimento da cliente fictícia Rose. No mesmo local, na Figura 5.8 quando houve um reconhecimento de Jairo, na resolução maior da esquerda há uma confiança maior.

Em outro local, na FACOM-UFMS, foi possível presenciar erros no reconhecimento. Nesse local a câmera em questão era de modelo genérico sem dados publicamente disponíveis e dessa forma foi interessante observar o comportamento do reconhecimento com *upsample* em situações de saturação maior de luz devido aos sensores da câmera. Assim na Figura 5.9 dá para notar que Carlos foi reconhecido erroneamente na imagem de menor resolução, a direita. No mesmo local, na Figura 5.10 quando o erro ocorre em ambas as resoluções, a menor resolução reporta um erro com uma confiança relevante.

Figura 5.8: Exemplo de diferença na confiança do reconhecimento

Figura 5.9: Exemplo de erro no reconhecimento

5.5 Redução do modelo por quantização

Para a melhora do desempenho, não somente o redimensionamento dos quadros foi aplicado, mas também a otimização a nível de modelo da CNN. Assim, essa seção procura responder: como otimizar o modelo? E qual o impacto dessa otimização no tempo de processamento e precisão desse modelo otimizado?

Foram executadas medições quanto ao desempenho dos algoritmos levando em consideração o FPS, quantização e tamanho do quadro. Para as medições,

Figura 5.10: Exemplo de erro no reconhecimento com confiança alta

os pesos do modelo foram otimizados para INT8 através da biblioteca TensorRT para aceleração de inferência em GPUs Nvidia. Na extração dos dados utilizou-se o mesmo algoritmo treinado para as contagens de pessoas, Yolov5 em sua versão *Small*.

Dessa forma houve uma perda mínima da precisão menor que 1% em razão do ganho relativamente alto em otimização do custo computacional. Os resultados das medições de desempenho da Yolov5s em testes locais com GPU GTX 1060 podem ser observados na Tabela 5.5, bem como o número de câmeras que podem ser processadas em paralelo considerando tempo real a inferência de até 20 FPS. Para os testes com FP16 utilizou-se da plataforma *Jetson Nano*, uma vez que o hardware GTX 1060 e T4 não possuem suporte para operações com essa precisão.

Precisão	**Biblioteca**	**Resolução**	**Inferência**	**FPS**	**Câmeras**	**GPU**
FP32	PyTorch	640x352	14ms	~71	3	T4
FP32	TensorRT	640x352	12ms	~83	4	T4
INT8	TensorRT	640x352	6ms	~166	8	T4
FP16	TensorRT	640x352	70ms	~14	0.7	JetsonNano

Tabela 5.5: Comparativo de Desempenho Yolov5s otimizado

Os resultados com a quantização acarretam em perda considerada quase inexistente quando levado em consideração o ganho em tempo de processamento (Migacz, 2017). O comportamento foi observado no algoritmo Yolov5s utilizado

neste trabalho.

Na AWS, serviço de máquinas na nuvem da Amazon, é possível estimar um custo mensal para implantação do algoritmo otimizado. Na máquina *g4dn.xlarge* com uma NVIDIA T4 de 16GB o custo é de 0.54$ a hora. Um lojista abre sua loja em média 12 horas por dia, até mesmo um shopping center tem seu funcionamento baseado nesse horário. Com os dados da Tabela 5.5, 30 dias de funcionamento e 8 câmeras em paralelo, o custo médio por câmera para o lojista é 24.4USD por mês. Esse custo pode ser até 70% mais baixo se as instâncias forem reservadas, pois o preço calculado anteriormente em cima do uso sob demanda. Esse valor é viável, se levar em consideração que é uma máquina em nuvem que não demanda manutenção, um algoritmo de contagem que possui em sua implementação algoritmos de estado da arte implementados, reconhecimento facial e com possibilidade de utilização de câmeras já instaladas. Vale ainda mencionar que todo o processo ocorre em uma única câmera.

CAPÍTULO

6

Conclusões

Nas seções seguintes são apresentados os pontos fracos, as dificuldades bem como as contribuições resultantes desse trabalho e possibilidades de trabalhos futuros relacionados ao tema proposto de Contagem do fluxo de pessoas utilizando aprendizado profundo.

6.1 Pontos fracos e dificuldades

Algumas dificuldades e pontos fracos podem ser mencionados:

- Avaliação experimental em qualidade em HD pode não permitir generalizar os resultados para outras resoluções. Acreditamos que um estudo em um intervalo maior de resoluções possa permitir melhor generalizar este trabalho em outras resoluções;
- O laboratório não tem disponível GPUs de arquiteturas mais recentes para avaliar de maneira mais criteriosa a quantização. O laboratório possui apenas uma Jetson Nano que possibilitou um experimento com FP16;
- A falta de fotos reais dos funcionários nos vídeos. Alguns dos funcionários dos vídeos não tinham mais vínculo com a empresa, ou se recusaram em

fornecer dados de sua face. Assim as faces eram extraídas direto do vídeo coletado, podendo enviesar o algoritmo de reconhecimento facial.

6.2 Contribuições

Mesmo com as dificuldades houveram contribuições interessantes para a comunidade científica e para o público em geral. Assim, é esperado que o desenvolvimento dos algoritmos propostos viesse a trazer o menor impacto possível para quem fosse utilizar e também de maneira clara, explicar seu funcionamento e embasamento para o leitor. Dessa forma esse trabalho contribui com:

- Avaliação criteriosa da contagem de pessoas através de técnicas distintas;
- Avaliação em ambientes de funcionamento distintos que, possibilitam a contagem de clientes com distinção de funcionários por reconhecimento facial;
- Reconhecimento facial em imagens redimensionadas através do da técnica de *upsample* da face.

6.3 Trabalhos Futuros

Com o desenvolvimento da proposta alguns pontos de melhoria e ideias foram identificados, estes podem ser estudados em trabalhos futuros. Alguns deles são:

- Validar as contagens em intervalos maiores de tempo;
- Otimizações com processamento apenas de áreas de interesse do quadro, para reduzir a quantidade de informação processada;
- Algoritmo único de detecção de pessoas e reconhecimento facial em um único passo de *forward* nas CNNs;
- Explorar qualidades de imagens superiores para a técnica de *upsample*;

Por último, incrementar o algoritmo com análise facial do cliente como: Gênero, Idade Aproximada e Humor. Há também a possibilidade de análise de atributos da pessoa como um todo: Acessórios no corpo, tipo de vestimenta, dentre outros atributos. Estes trabalhos futuros podem enriquecer ainda mais a contagem, a análise do fluxo e fomentar a busca por novos campos de pesquisa.

Referências Bibliográficas

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541. 11

Bergmann, P., Meinhardt, T., and Leal-Taixe, L. (2019). Tracking without bells and whistles. 28

Bochinski, E., Eiselein, V., and Sikora, T. (2017a). High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE. 28

Bochinski, E., Eiselein, V., and Sikora, T. (2017b). High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE. 29

Bochinski, E., Senst, T., and Sikora, T. (2018). Extending iou based multi-object tracking by visual information. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE. 29

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. 53

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*. 9

Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE. 28

Boom, B., Beumer, G., Spreeuwers, L. J., and Veldhuis, R. N. (2006). The effect of image resolution on the performance of a face recognition system. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE. 52

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer. 13

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer. 15

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.". 6

Cai, Z., Yu, Z. L., Liu, H., and Zhang, K. (2014). Counting people in crowded scenes by video analyzing. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 1841–1845. IEEE. 34

Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8609–8613. IEEE. 12

Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., and Zhang, L. (2021). Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382. 53

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., and Leal-Taixé, L. (2019). CVPR19 tracking and detection

challenge: How crowded can it get? *arXiv:1906.04567 [cs]*. arXiv: 1906.04567. 30

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019a). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699. vi, 38, 49

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., and Zafeiriou, S. (2019b). Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*. 21, 38

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338. 26

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130. 17

Gantz, J. and Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16. 2

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. (2021). Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928. 53

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. 24

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. 12

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. 10, 18

Guan, M., Li, Z., He, R., and Wen, C. (2018). High speed tracking with a fourier domain kernelized correlation filter. *arXiv preprint arXiv:1811.03236*. 28

Guyon, I. and Elisseeff, A. (2006). An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer. 7

Haykin, S. (2010). *Neural networks and learning machines, 3/E*. Pearson Education India. 11

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. 21

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. 12

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 17, 18

He, Y., Xu, D., Wu, L., Jian, M., Xiang, S., and Pan, C. (2019). Lffd: A light and fast face detector for edge devices. *arXiv preprint arXiv:1904.10633*. 41

Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596. 28

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst. 5, 38

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*. 12

Jain, A. K. and Farrokhnia, F. (1990). Unsupervised texture segmentation using gabor filters. In *1990 IEEE international conference on systems, man, and cybernetics conference proceedings*, pages 14–19. IEEE. 7

Jocher, G., Stoken, A., Borovec, J., NanoCode012, Chaurasia, A., TaoXie, Changyu, L., V, A., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Hajek, J., Diaconu, L., Marc, Kwon, Y., oleg, wanghaoyang0106, Defretin, Y., Lohia, A., ml5ah, Milanko, B., Fineran, B., Khromov, D., Yiwei, D., Doug, Durgesh, and Ingham, F. (2021). ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations. vi, 42, 53

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. 30

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 15

Krig, S. (2016). Image pre-processing. In *Computer vision metrics*, pages 35–74. Springer. 6

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. 5

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 17

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. 10, 13, 16

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. v, 11, 13, 17

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer. 12

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 22

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer. 26

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer. v, 21, 23

Loy, C. C., Lin, D., Ouyang, W., Xiong, Y., Yang, S., Huang, Q., Zhou, D., Xia, W., Li, Q., Luo, P., et al. (2019). Wider face and pedestrian challenge 2018: Methods and results. *arXiv preprint arXiv:1902.06854*. 54

Lui, Y. M., Bolme, D., Draper, B. A., Beveridge, J. R., Givens, G., and Phillips, P. J. (2009). A meta-analysis of face recognition covariates. In *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–8. IEEE. 52

Ma, Z. and Chan, A. B. (2013). Crossing the line: Crowd counting by integer programming with local features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 34

Michalski, R. S. (1983). A theory and methodology of inductive learning. In *Machine learning*, pages 83–134. Springer. 9

Migacz, S. (2017). 8-bit inference with tensorrt. In *GPU technology conference*, volume 2, page 5. 62

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*. 28, 29

Mitchell, T. M. et al. (1997). Machine learning. 8

Neubeck, A. and Van Gool, L. (2006). Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE. 22

Nogueira, V., Oliveira, H., Silva, J. A., Vieira, T., and Oliveira, K. (2019). Retailnet: A deep learning approach for people counting and hot spots detection in retail stores. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 155–162. IEEE. 35

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., and Walsh, J. (2019). Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer. 5

Polidoro, C. H., de Castro, W. V., Marcato, J., Salgado Filho, G., and Matsubara, E. T. (2019). Counting cars from aerial videos using deep learning. In *EPIA Conference on Artificial Intelligence*, pages 637–649. Springer. 37

Prince, S. J. (2012). *Computer vision: models, learning, and inference*. Cambridge University Press. 4, 6, 7

Qin, Z., Li, Z., Zhang, Z., Bao, Y., Yu, G., Peng, Y., and Sun, J. (2019). Thundernet: Towards real-time generic object detection on mobile devices. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6718–6727. vi, 39, 40

Quinlan, J. R. (1993). *C4. 5: Programs for Machine Learning*. Morgan Kaufmann. 10

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. 21, 22, 37

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99. 21, 22, 24

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. 11

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015a). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252. 6, 17

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015b). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252. 6

Schulzrinne, H., Rao, A., and Lanphier, R. (1998). Real time streaming protocol (rtsp). 46

Sengupta, S., Cheng, J., Castillo, C., Patel, V., Chellappa, R., and Jacobs, D. (2016). Frontal to profile face verification in the wild. In *IEEE Conference on Applications of Computer Vision*. 38

Sharma, S. and Sharma, S. (2017). Activation functions in neural networks. *Towards Data Science*, 6(12):310–316. 11

Shumsky, R. A., Debo, L., Lebeaux, R. M., Nguyen, Q. P., and Hoen, A. G. (2021). Retail store customer flow and covid-19 transmission. *Proceedings of the National Academy of Sciences*, 118(11). 1

SI, C. H. and Stephens, M. (1988). A combined corner and edge detection. In *Proc. 4th Alvey Vision Conference*, pages 147–151. 7

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 17

Sindagi, V. A. and Patel, V. M. (2018). A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16. 36

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*. 15

Sruthi, M. (2019). Iot based real time people counting system for smart buildings. *International Journal of Emerging Technology and Innovative Engineering*, 5(2). 35

Sun, S., Akhtar, N., Song, H., Zhang, C., Li, J., and Mian, A. (2019). Benchmark data and method for real-time people counting in cluttered scenes using depth sensors. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3599–3612. 34

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*. 39

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. 17

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media. 5

Taiana, M., Nascimento, J. C., and Bernardino, A. (2013). An improved labelling for the inria person data set for pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 286–295. Springer. 54

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*. 26

Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790. vi, 43, 53

Tretyak, O. A. and Sloev, I. (2013). Customer flow: evaluating the long-term impact of marketing on value creation. *Journal of Business & Industrial Marketing*. 1

Vanholder, H. (2016). Efficient inference with tensorrt. v, vi, 32, 33

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley. 6, 10

Velipasalar, S., Tian, Y.-L., and Hampapur, A. (2006). Automatic counting of interacting people by using a single uncalibrated camera. In *2006 IEEE International Conference on Multimedia and Expo*, pages 1265–1268. IEEE. 34

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13029–13038. 53

Wang, R. J., Li, X., and Ling, C. X. (2018). Pelee: A real-time object detection system on mobile devices. In *Advances in Neural Information Processing Systems*, pages 1963–1972. vi, 39

Wang, Y. E., Wei, G.-Y., and Brooks, D. (2019). Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*. 31

Witten, I. H. and Frank, E. (2002). Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77. 8

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE. 28, 30, 48

Yam, K.-Y., Siu, W.-C., Law, N.-F., and Chan, C.-K. (2011). Effective bi-directional people flow counting for real time surveillance system. In *2011 IEEE International Conference on Consumer Electronics (ICCE)*, pages 863–864. IEEE. 36

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es. 27

Yoo, Y., Han, D., and Yun, S. (2019). Extd: Extremely tiny face detector via iterative filter reuse. *arXiv preprint arXiv:1906.06579*. vi, 41

Zhang, S., Wu, G., Costeira, J. P., and Moura, J. M. (2017a). Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 3667–3676. vi, 37

Zhang, S., Wu, G., Costeira, J. P., and Moura, J. M. (2017b). Understanding traffic density from large-scale web camera data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5898–5907. 36

Zhao, Z., Li, H., Zhao, R., and Wang, X. (2016). Crossing-line crowd counting with two-phase deep neural networks. In *European Conference on Computer Vision*, pages 712–726. Springer. 36

Zhou, X., Koltun, V., and Krähenbühl, P. (2021). Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*. 53

Zhou, Y.-T. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78. 19

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*. 21