UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ALINE MOURA ARAÚJO

# DETECÇÃO E DESTAQUE EM VÍDEO DE OBJETOS UTILIZANDO YOLO

JOÃO PESSOA - PB
2022

ALINE MOURA ARAÚJO

# DETECÇÃO E DESTAQUE EM VÍDEO DE OBJETOS UTILIZANDO YOLO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba, como requisito parcial à obtenção do título de Mestre em Informática.

Área: Sistemas De Computação.

Linha de pesquisa: Sinais, Sistemas Digitais E Gráficos, Programa De Pós-graduação em Informática.

Orientadora: Profª. Drª. Thaís Gaudencio do Rêgo
Co-Orientador: Prof. Dr. Lincoln David Nery e Silva

JOÃO PESSOA - PB

2022

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

**Catalogação na publicação**
**Seção de Catalogação e Classificação**

A663d Araújo, Aline Moura.
Detecção e destaque em vídeo de objetos utilizando YOLO / Aline Moura Araújo. - João Pessoa, 2022.
67 f. : il.

Orientação: Thaís Gaudencio do Rêgo.
Coorientação: Lincoln David Nery e Silva.
Dissertação (Mestrado) - UFPB/CI.

1. Sistemas de computação. 2. YOLO. 3. Instrumentos musicais. 4. Detecção de objetos. 5. Detecção em vídeo. I. Rêgo, Thaís Gaudencio do. II. Silva, Lincoln David Nery e. III. Título.

UFPB/BC CDU 004.42(043)

Elaborado por WALQUELINE DA SILVA ARAUJO - CRB-15/514

**UNIVERSIDADE FEDERAL DA PARAÍBA**
**CENTRO DE INFORMÁTICA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Aline Moura Araújo, candidata ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 30 de outubro de 2022.

Aos trinta dias do mês de outubro, do ano de dois mil e vinte e dois, às quinze horas e trinta minutos, no Centro de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final da Srª. Aline Moura Araújo, vinculada a esta Universidade sob a matrícula nº 20201006310, candidata ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Sinais, Sistemas Digitais e Gráficos", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Thaís Gaudencio do Rêgo (PPGI-UFPB), Orientadora e Presidente da Banca, Leonardo Vidal Batista (PPGI-UFPB), Examinador Interno, Lincoln David Nery e Silva (UFPB), Examinador Externo ao Programa e Co-orientador, Telmo de Menezes e Silva Filho (UFPB), Examinador Externo ao Programa, Yuri de Almeida Malheiros Barbosa (UFPB), Examinador Externo ao Programa. Dando início aos trabalhos, a Presidente da Banca cumprimentou os presentes, comunicou a finalidade da reunião e passou a palavra à candidata para que ela fizesse a exposição oral do trabalho de dissertação intitulado "Detecção e Destaque em Vídeo de Objetos Utilizando Yolo". Concluída a exposição, a candidata foi arguida pela Banca Examinadora que emitiu o seguinte parecer: "**aprovada**". Do ocorrido, eu, Fernando Menezes Matos, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 30 de outubro de 2022.

Prof. Fernando Menezes Matos

Prof. Thaís Gaudencio do Rêgo
Orientadora (PPGI-UFPB)

Prof. Leonardo Vidal Batista
Examinador interno (PPGI-UFPB)

Prof. Lincoln David Nery e Silva
Examinador Externo ao Programa (UFPB)

Prof. Telmo de Menezes e Silva Filho
Examinador Externo ao Programa (UFPB)

Prof. Yuri de Almeida Malheiros Barbosa
Examinador Externo ao Programa (UFPB)

# RESUMO

Em dispositivos celulares e câmeras, está crescentemente presente a funcionalidade de ampliação de imagem (do inglês, *zoom*), no entanto, ainda é uma inovação utilizar detecção de objetos para automatizar esta tarefa. A detecção é um problema clássico relacionado à visão computacional que trata da localização de instâncias de objetos semânticos em uma classe específica. Nesse sentido, o objetivo deste trabalho foi implementar um sistema capaz de realizar a detecção de objetos de forma automática em vídeo, avaliando seu funcionamento com guitarra, violão e microfone. Após essa detecção, um novo vídeo foi gerado dando ênfase ao instrumento detectado, a fim de facilitar a observação de sua execução. Para tanto, foi desenvolvido um sistema que, utilizando um modelo YOLOv4, é capaz de identificar objetos e fazer um procedimento semelhante a um *zoom in* no vídeo. Foi implementado um *pipeline*, onde é feita primeiramente uma extração dos quadros dos vídeos, e, em seguida, a detecção de um objeto parametrizado em um intervalo de 12 quadros. Após a detecção, é feito o recorte seguindo uma metodologia de estabilização para tratar a fluidez do vídeo, e, por fim, um novo vídeo é gerado a partir desses recortes. Foram feitos testes com diferentes parâmetros de extração dos quadros, utilizando vídeos recuperados do Youtube, avaliando 4 cenários para a extração das imagens. Nesses testes, foram avaliados a performance do modelo de detecção, o tempo levado para a extração e o percentual de informação excluída no vídeo de saída em cada cenário. Para a validação desse *pipeline*, foi adotada uma metodologia, assumindo que detecção funcionaria de forma eficiente, para validar a heurística implementada na estabilização, a confiança do modelo para os 4 cenários de extração e o comportamento do sistema ao lidar com problemas de oclusão no vídeo. Na validação, foi utilizado um modelo pré-treinado da rede neural de código aberto YOLOv4 com 80 classes, realizando a detecção de objetos arbitrários, sendo gato e cachorro as classes escolhidas. Além disso, também foi treinado um modelo personalizado YOLOv4 para que seja capaz de fazer a detecção específica de instrumentos musicais, utilizando a base de dados da Imagenet. Em relação aos resultados da rede, apesar de não ser o foco deste trabalho, a precisão média alcançada pelo modelo nas classes guitarra, violão e microfone foi 61,90%, 87,94% e 62,27%, respectivamente. No sistema de *zoom in*, foi possível perceber que, quanto melhor o parâmetro de extração, maior é a quantidade de objetos detectados pelo modelo, como, também, a detecção tem maior precisão e qualidade. Houve uma pequena perda de qualidade em relação à resolução do vídeo original, e não houve perda significativa de conteúdo do vídeo devido ao intervalo dos quadros na detecção. Concluindo as análises dos resultados obtidos, é possível afirmar que a proposta do trabalho obteve êxito, pois, todos os objetivos apresentados foram alcançados. Para trabalhos futuros, almeja-se testar novos modelos de detecção, implementação de novos critérios de avaliação do vídeo de saída e paralelização das etapas do pipeline.

**Palavras-chave:** Instrumentos Musicais, Detecção de Objetos, Detecção em Vídeo, YOLO

# ABSTRACT

In mobile devices and cameras, the image magnification (zoom) functionality is increasingly present, however, it is still an innovation to use object detection to automate this task. Detection is a classic problem related to computer vision that deals with the location of instances of semantic objects in a specific class. In this sense, the objective of this work was to implement a system capable of performing the detection of objects automatically in video, evaluating its operation with acoustic guitar, electric guitar and microphone. After this detection, a new video was generated emphasizing the detected instrument, in order to facilitate the observation of its execution. For that, a system was developed that, using a YOLOv4 model, is able to identify objects and perform a procedure similar to a zoom in on the video. A pipeline was implemented, where the frames are first extracted, and then the detection of a parameterized object in an interval of 12 frames. After detection, the clipping is made following an interpolation methodology to deal with the fluidity of the video, and, finally, a new video is generated from these clippings. Tests were made with different parameters for extracting the frames, using videos retrieved from Youtube, evaluating 4 scenarios for extracting the images. In these tests, the performance of the detection model, the time taken for extraction and the percentage of information excluded in the output video in each scenario were evaluated. For the validation of this pipeline, a methodology was adopted, assuming that detection would work efficiently, to validate the heuristic implemented in the interpolation, the confidence of the model for the 4 extraction scenarios and the behavior of the system when dealing with occlusion problems in the video.. In the validation, a pre-trained model of the YOLOv4 open source neural network with 80 classes was used, performing the detection of arbitrary objects, with cat and dog being the chosen classes. In addition, a customized YOLOv4 model was also trained to be able to perform specific detection of musical instruments with the Imagenet database. Regarding the network results, although not the focus of this work, the average accuracy achieved by the model in the guitar, acoustic guitar and microphone classes was 61.90%, 87.94% and 62.27%, respectively. In the zoom in system, it was possible to notice that the better the extraction parameter, the greater the number of objects detected by the model, as well as the greater precision and quality of detection. There was a small loss of quality from the resolution of the original video, and there was no significant loss of video content due to frame gap in detection. Concluding the analysis of the results obtained, it is possible to affirm that the proposal of the work was successful, therefore, all the presented objectives were reached. For future work, we aim to test new detection models, implement new output video evaluation criteria and parallelize the pipeline steps.



**Keywords:** Musical Instruments, Object Detection, Video Detection, YOLO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS

AM – Aprendizagem de máquina

API – *Application Programming Interface* (Interface de Programação de Aplicação)

CNN – *Convolutional Neural Network* (Rede Neural Convolucional)

CUDA – *Compute Unified Device Architecture* (Arquitetura de Dispositivo de Computação Unificada)

DL – *Deep Learning* (Aprendizagem Profunda)

FPS – *Frames per Second* (Frames por segundo)

GAN – *Generative Adversarial Network* (Rede Adversária Generativa)

GPU – *Graphics Processing Unit* (Unidade de Processamento Gráfico)

IA – Inteligência Artificial

RAM – *Random Access Memory* (Memória de Acesso Randômico)

SPP - *Spatial Pyramid Pooling* (Pooling de Pirâmide Espacial)

YOLO – *You Only Look Once*

## Sumário

# 1 INTRODUÇÃO

## 1.1 Contextualização do Problema

A funcionalidade de ampliação de imagem (do inglês, *zoom*) está presente cada vez mais em telefones celulares e câmeras hoje, melhorando exponencialmente as suas resoluções para sanar a necessidade do usuário de conseguir visualizar algo que esteja distante, com um certo nível de detalhamento. As pessoas fazem *zoom* em objetos em uma imagem ou vídeo, como animais ou jogadores em esportes, para visualizar o conteúdo com mais detalhes. A proposta do trabalho é automatizar esse *zoom* em um vídeo, focando em um objeto específico, para criar um novo vídeo executando o método de detecção de objetos utilizando inteligência artificial.

A detecção de objetos é um problema clássico relacionado à visão computacional e ao processamento de imagens, que trata da localização de instâncias de objetos semânticos de uma classe específica (como instrumentos musicais, humanos, edifícios, mesas ou carros) em imagens digitais e vídeos, que consistem em uma sequência de quadros [1].

Na última década, a aplicação de redes neurais profundas impulsionou significativamente o estado da arte em visão computacional, levando a avanços em tarefas como classificação de imagens ou detecção de objetos [2] [3]. Mesmo diante de todo o progresso em modelos de classificação e detecção, ainda é um desafio para sistemas de visão computacional realizar tais tarefas, por exemplo, em situações em que objetos estejam parcialmente obstruídos.

Neste trabalho, é proposta uma metodologia de edição de vídeo inteligente que seja capaz de gerar um recorte do vídeo original, com ênfase em um objeto específico, que deverá ser parametrizado para o usuário, auxiliando-o a acompanhar o objeto de forma mais clara. O sistema proposto funciona para a detecção de objeto, porém, a aplicação avaliada neste trabalho foi em instrumentos musicais. Podendo ser aplicado ainda, para auxiliar o aprendizado do instrumento, uma vez que melhora a visualização de sua execução.

Para uma aplicação de detecção e reconhecimento de objetos, por exemplo, temos instrumentos musicais, em específico, que, para os humanos, exige-se uma combinação de modalidades de percepção múltipla. Para distinguir um violão de um violino, podemos

simplesmente observar as diferenças em seu tamanho, presença e movimento do arco, posição do instrumento em relação ao corpo do instrumentista, entre outras características. Apesar da tarefa ser relativamente fácil de ser executada por humanos, combinar informações multimodais não é trivial para algoritmos de aprendizagem de máquina [4].

Para o seu desenvolvimento, foi escolhida a rede neural de código aberto YOLO [5], treinada para que seja capaz de fazer a detecção de instrumentos musicais, a aplicação deste presente trabalho, em vídeo, de forma eficiente e prezando majoritariamente pela sua performance e qualidade.

Além da dificuldade em desenvolver um modelo eficiente e rápido, há ainda o obstáculo de conseguir detectar, corretamente, o objeto desejado, destacar (*zoom in*) a detecção em cada quadro para a criação de um novo vídeo que não perca qualidade e transformá-lo em um produto que seja útil para o usuário. Dessa forma, foram considerados aspectos importantes no desenvolvimento do *pipeline* do trabalho, como resolução, fluidez e o mínimo de distorção possível do vídeo de saída, balanceados com o menor tempo de execução e precisão ao longo da detecção.

### 1.1.1 Objetivo geral

Desenvolver, implementar e avaliar uma abordagem capaz de realizar a detecção de objetos de forma automática em vídeo e gerar um novo vídeo para dar ênfase ao objeto desejado, escolhido pelo usuário.

### 1.1.2 Objetivos específicos

- Criação de um conjunto de dados aplicado ao contexto de instrumentos musicais, correspondente ao cenário em que o modelo será usado;

- Definição de um processo para realizar a detecção e aproximação de objetos em vídeo de forma automatizada;

- Implementação de um sistema para realizar a detecção e aproximação de objetos em vídeo de forma automatizada;

- Avaliação de combinações de parâmetros que impactam na resolução de extração de quadros para avaliar a capacidade de detecção do modelo;

- Aplicação e avaliação do sistema no contexto de instrumentos musicais.

## 1.2 Estrutura da dissertação

No Capítulo 2, serão abordados temas importantes para o entendimento do atual trabalho, tais como, uma breve introdução sobre Aprendizagem de Máquina, Detecção de objetos e Detecção de objetos em vídeo, e uma apresentação sobre o funcionamento da Rede Neural de código aberto utilizada neste trabalho, a YOLO v4 [6]. Em seguida, no Capítulo 3, serão apresentados trabalhos relacionados a este e suas abordagens, apresentando suas principais semelhanças e diferenças.

No Capítulo 4, será descrita toda a metodologia utilizada no desenvolvimento deste trabalho, iniciando pelas dependências e *hardware* utilizados no seu desenvolvimento, e, em seguida, a implementação do *pipeline* proposto e o método de avaliação utilizado. Por fim, será discutido o treinamento da rede, onde serão abordadas informações sobre o material utilizado, os ajustes realizados na rede, como foi feito o tratamento das imagens da base de dados e as métricas utilizadas na etapa de treinamento.

No Capítulo 5, serão exibidos os resultados obtidos nas etapas de treinamento da rede e do sistema responsável por gerar o novo vídeo. Será feita uma discussão dos resultados obtidos em ambas as etapas, bem como, as limitações do sistema e as próximas etapas do trabalho. Por fim, no Capítulo 6, serão apresentadas as considerações finais e trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar conceitos relacionados à metodologia utilizada neste trabalho. Inicialmente, é feita uma breve introdução sobre o conceito de Aprendizagem de Máquina (AM). Em seguida, são apresentados os fundamentos teóricos de Detecção de Objetos em imagens e em vídeos e o método de avaliação de desempenho da detecção, conhecido por Interseção pela União (do inglês, *Intersection over Union* - IoU). Por fim, é feita uma apresentação de como a YOLO [5] funciona e, em mais detalhes, a versão utilizada neste trabalho, bem como suas melhorias.

## 2.1 Aprendizagem de Máquina

Aprendizagem de máquina pode ser definida como um subconjunto da inteligência artificial (IA), que trata do estudo científico de algoritmos e modelos estatísticos, que os sistemas computacionais usam para executar uma tarefa específica, sem usar instruções explícitas, confiando em padrões e inferências [7]. Dessa forma, pode-se afirmar que AM é um método de análise de dados que automatiza a construção de modelos analíticos. Os algoritmos de AM constroem um modelo matemático baseado em dados de amostra, conhecidos como dados de treinamento, para fazer predições ou decisões [7].

Esse ramo da IA se baseia na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana. Nesse contexto, algoritmos de AM possuem grande espectro de aplicações, como processamento de linguagem natural [8-10], mineração de texto [11], reconhecimento de voz [12-14] e visão computacional [15-18]. Neste trabalho, a AM foi utilizada para a detecção de objetos em vídeos, uma técnica utilizada em recursos visuais.

## 2.2 Detecção de Objetos

Na área de visão computacional, a detecção de objetos é um termo geral para descrever uma coleção de tarefas que envolvem a identificação de objetos em imagens digitais [19]. Esta técnica identifica e localiza objetos em uma imagem ou vídeo e geralmente consiste em diversas aplicações como detecção de pedestres na rua [20], de

faces [21], de veículos em vídeo ou imagem [22, 23] e de sistema de contagem automática [24, 25].

Como um dos problemas fundamentais da visão computacional, a detecção de objetos é capaz de fornecer informações para a compreensão semântica de imagens e vídeos e está relacionada a muitas aplicações, incluindo classificação de imagens [26, 27], análise do comportamento humano [28] e reconhecimento facial [29]. Com este tipo de identificação e localização, a detecção de objetos pode ser usada para contar objetos em uma cena, determinar e rastrear suas localizações e rotulá-los com precisão.

Devido a grandes variações nos pontos de vista, poses, oclusões e condições de iluminação, é difícil realizar perfeitamente a detecção de objetos com uma tarefa adicional de localização de objetos. Desta forma, muita atenção tem sido atraída para este campo nos últimos anos [5] [30-32].

A definição do problema da detecção de objetos é determinar onde eles estão localizados em uma determinada imagem (localização do objeto) e a qual categoria cada um pertence (classificação do objeto). Portanto, o *pipeline* dos modelos tradicionais de detecção de objetos pode ser dividido principalmente em três estágios: seleção de região informativa, extração de características e classificação.

Vale ressaltar que, na primeira fase, a de seleção de região informativa, os objetos podem aparecer em qualquer posição da imagem e ter proporções ou tamanhos diferentes. Para tanto, é feito, comumente, a análise da imagem inteira com uma janela deslizante em várias escalas.

Essa estratégia, apesar de descobrir todas as posições possíveis dos objetos, é exaustiva. Devido ao grande número de janelas candidatas, é computacionalmente cara e produz muitas janelas redundantes. No entanto, se apenas um número fixo de modelos de janela deslizante for aplicado, regiões insatisfatórias podem ser produzidas [33].

Enquanto isso, o estágio de extração de características tem como objetivo reduzir uma imagem de tamanho variável a um conjunto fixo de atributos visuais, que são identificados através de fotos ou figuras, para contextualizar ou fornecer o significado da palavra alvo, além da utilização dos aspectos gráficos. Modelos de classificação de imagens são normalmente construídos usando métodos de extração de atributos visuais, que sejam representativos para a tarefa e use esses atributos para determinar a classe da imagem com maior eficiência

Quer sejam baseados em abordagens tradicionais de visão computacional [34] ou métodos de Aprendizagem Profunda (do inglês, *Deep Learning* - DL), todos esses modelos de classificação têm o mesmo objetivo, que é extrair atributos da imagem de entrada. Em estruturas de detecção de objetos, normalmente se usa modelos de classificação de imagem pré-treinados para extrair atributos visuais, já que eles tendem a generalizar bem.

A etapa de classificação, por fim, consiste em prever a classe de um objeto em uma imagem, nesse caso, uma porção da imagem principal. No processo final, a detecção é responsável por determinar a localização do objeto em uma imagem, especificar uma caixa delimitadora contendo o objeto e classificá-lo de acordo com as classes aprendidas no treinamento. Desta forma, a detecção de objetos em vídeo pode ser útil para detecção instantânea na visão de computadores, como veremos a seguir.

## 2.3 Detecção de Objetos em Vídeo

Na primeira metade da década de 2010, os trabalhos pioneiros no campo da visão computacional abordaram principalmente o processamento de imagens, como classificação [26, 27], detecção [35-38] e segmentação [39-42], enquanto o campo do processamento de vídeo foi menos explorado.

Um motivo claro para essa diferença é porque um vídeo é essencialmente uma sequência de imagens, chamados de quadros (do inglês, *frame*). No entanto, essa definição não pode encapsular todo o cenário do que é o processamento de vídeo, uma vez que este adiciona uma nova dimensão temporal ao problema.

Os vídeos não são apenas sequências de imagens, e, sim, sequência de imagens relacionadas. Embora pareça uma diferença mínima, é possível explorar essa dimensão de diversas maneiras que não se aplicam a imagens únicas. Além disso, devido à complexidade de dados em vídeo (tamanho, anotações relacionadas) e ao alto custo de treinamento e inferência, tem sido mais difícil avançar neste campo.

Recentemente, no entanto, com o lançamento do ImageNet VID [43], e outros conjuntos de dados de vídeo durante a segunda metade da década de 2010, mais trabalhos de pesquisa relacionados ao assunto surgiram, tais como, métodos de pós-processamento [44-46], multi-*frame* [47,48] e fluxo ótico [49,50]. Um dos métodos mais utilizados é o modelo YOLO, proposto por Redmon et al. (2016) [5], que, em sua simplicidade, prevê

caixas delimitadoras e probabilidades de classe com uma única rede em uma única avaliação, permitindo previsões em tempo real.

## 2.4 YOLO

YOLO (do inglês, *You Only Look Once*) utiliza Redes Neurais Convolucionais (do inglês, *Convolutional Neural Networks*, CNN) para detecção de objetos. Essa rede propõe uma abordagem ao problema de detecção e lida com essa tarefa como um único problema de regressão. Assim, com uma CNN, podemos prever caixas delimitadoras, e a confiança da classe, olhando apenas uma vez para a imagem.

O diferencial desse método é que, como o próprio nome sugere, a rede olha uma única vez para a imagem, fazendo da YOLO um dos algoritmos mais rápidos de detecção já implementados. Porém, para que isso se torne possível, parte da precisão do algoritmo é comprometida. A YOLO é capaz de detectar objetos em tempo real de até 30 FPS (do inglês, *frames per second*) [5].

Para realizar a detecção, a imagem é dividida em uma grade de $S$ x $S$ células. O tamanho dessas células pode divergir de acordo com a versão da YOLO, podendo ter tamanhos como 3x3, 5x5 e 19x19. Em cada uma das células são previstas $N$ (letra ilustrativa) possíveis caixas delimitadoras e a confiança de cada uma delas. Isso significa que as caixas são calculadas por $S$ x $S$ x $N$. Para fazer uma filtragem de todas as detecções encontradas na imagem, o algoritmo define um limiar de confiança e as caixas delimitadoras com valores abaixo desse limiar são desconsideradas.

Há ainda a probabilidade de classe condicional para cada célula contendo mais de um objeto, onde apenas uma classe pode ser detectada, independente do número de caixas delimitadoras, ou seja, cada célula só pode ter uma classe detectada. Se o centro de um objeto estiver sobre uma célula da grade, aquela célula é responsável pela detecção daquele objeto.

Para lidar com o problema de várias detecções em uma mesma célula, é feito um mapa de probabilidade de classes, onde apenas uma classe pode ser detectada por célula, independente do número de caixas delimitadoras.

As caixas restantes, que se sobrepõem, passam por um processo de supressão, que eliminará possíveis objetos duplicados e, assim, deixará apenas os mais precisos. Por fim,

as caixas delimitadoras são definidas a partir das extremidades definidas nas células do mapa de probabilidade [5], exemplificado na Figura 1.

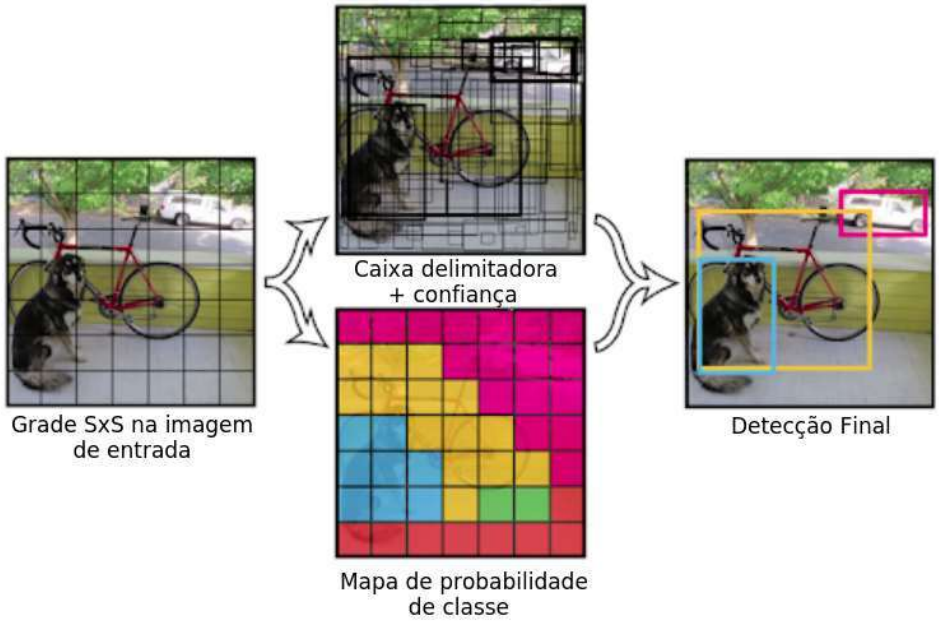



**Figura 1 -** Exemplo de detecção usando YOLO. No mapa de probabilidade de classe, cada cor representa a possível presença de uma determinada classe naquela grade. Em detecções finais, podemos ver que as cores, rosa, azul e laranja representam a classe a qual o objeto pertence.

**Fonte**: Adaptada de [4].

Por causa desse princípio de identificar apenas uma classe por célula da grade, a primeira versão da YOLO torna-se limitada quando há objetos muito próximos na imagem, não conseguindo detectar todos [5].

As correções de limitações na YOLO [5] foram sendo realizadas conforme o surgimento de novas versões, porém, outras limitações surgiram. Com novas previsões em várias escalas, a YOLOv3 [51], por exemplo, tem um desempenho relativamente alto, porém um desempenho pior em objetos de tamanho médio e grande, em comparação ao tamanho da imagem.

Cada caixa delimitadora apresenta alguns resultados atribuídos a elas, tais como, a interseção pela união, a precisão dessa caixa, assim como outros 4 elementos: ($x$, $y$, $w$, $h$). A tupla ($x$, $y$) representa o ponto central da caixa delimitadora, enquanto que a tupla ($w$, $h$) representa sua largura e altura, respectivamente, como é possível visualizar na Figura 2. Desta forma, esses 4 elementos representam a posição e tamanho da caixa delimitadora detectada pela rede.

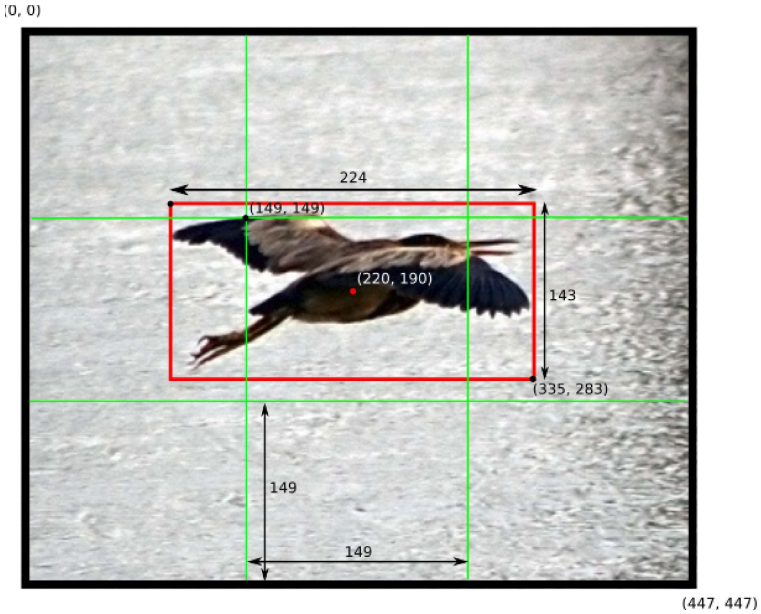



**Figura 2 -** Exemplo de uma imagem 448 x 448 dividida em células de grade de S = 3, representadas pelas linhas verdes, onde os componentes da caixa delimitadora do pássaro, representada pelas linhas vermelhas,. As duas tuplas (220, 190) e (224, 143) representam o ponto central da imagem detectada e a largura e altura da caixa delimitadora, respectivamente.

**Fonte**: [51].

A arquitetura da YOLO é representada por 3 estágios, são eles: *Backbone*, *Neck* e *Head*, como pode ser representado na Figura 3. O *Backbone* é uma rede neural profunda composta principalmente por camadas de convolução. Seu objetivo principal é extrair os atributos essenciais, portanto, a seleção dessa rede neural é um passo fundamental para melhorar o desempenho de detecção de objetos. Frequentemente, redes neurais pré-treinadas são usadas para treinar a rede do *Backbone*.

O *Neck* tem como papel essencial coletar mapas de características de diferentes estágios. E, por fim, temos a cabeça, ou *Head*, que tem como objetivo realizar uma previsão densa, que é composta por um vetor contendo as coordenadas da caixa delimitadora detectada (centro, altura, largura), a pontuação de confiança da previsão e o rótulo.

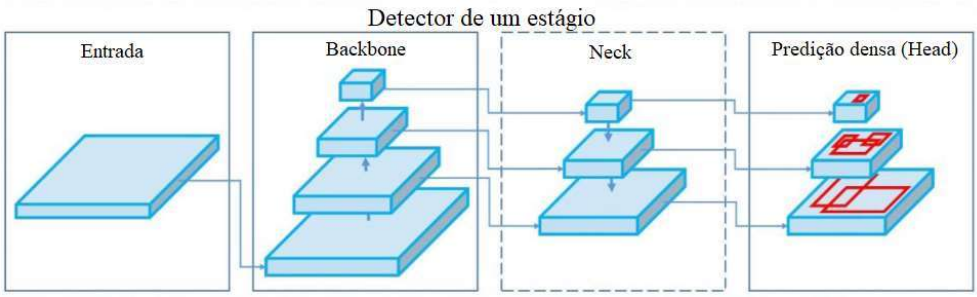



**Figura 3 -** Arquitetura geral de detector de objetos de um estágio, utilizada para a implementação da YOLOv4.

**Fonte**: Adaptada de [6].

A partir disso, podemos entender que o conceito de convolução se insere nas redes neurais, pois é um operador linear que, a partir de duas funções, consegue gerar uma terceira, com a finalidade de medir a soma do produto, a partir da região subentendida em função do deslocamento existente entre as funções.

Os algoritmos YOLO continuaram a evoluir em diferentes versões desde o lançamento inicial em 2016. As três arquiteturas YOLO, YOLOv2 e YOLOv3 foram desenvolvidas por Joseph Redmon, onde a YOLOv2 focou principalmente em melhorar a revocação e localização das detecções, enquanto mantém a precisão [52]. Para isso, foram feitas algumas alterações na sua arquitetura, como a normalização de *batch* em todas as camadas convolucionais para melhorar a performance e um classificador com maior resolução foi utilizado no treinamento, aumentando de 224 para 448. Ademais, a YOLOv2 obteve um aumento de acurácia e performance no reconhecimento de vários objetos na mesma imagem, utilizando o método *Anchor Box* [53].

Na YOLOv3 é prevista uma pontuação de confiança para cada caixa delimitadora usando regressão logística. Dessa forma, utilizando classificadores logísticos independentes, um objeto detectado pode pertencer a mais de uma classe. Outra modificação foi utilizar grades *S x S* menores nas imagens de entrada, assim, se um objeto encontrar-se dentro de uma célula, a grade deve detectá-lo. Cada célula estima as informações de posição das caixas delimitadoras e calcula as pontuações de objetividade delas [54].

Já no YOLOv4, a arquitetura escolhida para este trabalho, foram introduzidas modificações na arquitetura do modelo, como *Bag of Freebies* (com aprimoramentos), função de ativação *Mish* entre outros, como serão discutidos em mais detalhes na seção 2.4.1.

Foram implementadas versões posteriores à YOLOv4, baseadas em sua arquitetura, como a YOLOv5, YOLOv6, YOLOv7, PP-YOLO e PP-YOLOv2, com mudanças arquiteturais e implementações com o *framework Pytorch* que resultaram em melhorias no desempenho em relação à velocidade de processamento e à acurácia [55-58].

A YOLOv4 foi escolhida por ser a arquitetura que representava o estado da arte para detecção de objetos em imagens no início da elaboração deste trabalho. Nenhuma nova versão foi testada, pois, para realizar essa mudança de modelo, implicaria em algumas dificuldades, como estudar os modelos, configurar os novos parâmetros e comparar métricas. Por falta de tempo hábil, a versão utilizada foi a YOLOv4.

### 2.4.1 YOLOv4

A arquitetura escolhida para ser utilizada no presente trabalho foi a YOLOv4 [6], implementação baseada na arquitetura da CSPDarknet53. Sua arquitetura representava o estado da arte para detecção de objetos em imagens no início da elaboração deste trabalho.

A YOLOv4 é uma melhoria da YOLOv3 [51]. A implementação de uma nova arquitetura *Backbone* e as modificações *Neck* melhoraram o mAP (do inglês, *mean Average Precision*) em 10% e o número de FPS em 12%. Além disso, ficou mais fácil treinar essa rede neural em uma única GPU. O *Head* é o mesmo da YOLOv3 [51].

A arquitetura utilizada para o *Backbone* da YOLOv4 é composta por três partes: *Bag of Freebies*, *Bag of Specials* e a arquitetura CSPDarknet53.

- *Bag of Freebies*: é um conjunto de métodos que aumentam o custo do treinamento ou mudam a estratégia de treinamento, enquanto deixam o custo da inferência baixo. Foram utilizadas algumas técnicas de regularização, como *Data Augmentation*, *DropBlock* [59] e *Label Smoothing* [60]. A regularização é um método utilizado para permitir que as redes neurais tenham um maior poder de generalização. Esse método é utilizado para melhorar a precisão na fase de teste.

  *Data Augmentation* é uma técnica que consiste em aumentar a diversidade do conjunto de treinamento aplicando transformações aleatórias, para isso, foram utilizadas as estratégias *CutMix* [61] e *Mosaic* [62].

  O método *DropBlock* foi introduzido para combater a principal desvantagem do *Dropout* [63] de descartar atributos aleatoriamente, o que prova ser uma estratégia

eficaz para redes totalmente conectadas, mas, menos lucrativa, quando se trata de camadas convolucionais em que os atributos são espacialmente correlacionados. A técnica *DropBlock* descarta atributos em uma área correlacionada contígua chamada de bloco. Ao fazê-lo, consegue cumprir o propósito de gerar um modelo mais simples e colocar, no conceito de aprendizagem, uma fração dos pesos da rede em cada iteração de treino para penalizar a matriz de peso, o que, por sua vez, reduz o *overfitting*, ou sobreajuste.

A suavização de rótulos, ou *Label Smoothing*, é uma forma de tornar o modelo mais robusto para que seja bem generalizado, tratando os problemas de superajuste e sobreajuste. Um modelo de classificação é calibrado se suas probabilidades de resultados previstos refletirem sua precisão.

Um modelo super confiante não é calibrado e suas probabilidades previstas são consistentemente maiores do que a precisão. A suavização de rótulos lida com esse problema utilizando o método *soft label*, onde a confiança dos rótulos é alterada. Em essência, o *Label Smoothing* ajuda o modelo a treinar em torno de dados rotulados incorretamente, e, consequentemente, melhora sua robustez e desempenho [43,60].

- *Bag of Specials*: é um conjunto de métodos que aumenta o custo de inferência em um pequeno valor, mas pode melhorar significativamente a precisão da detecção de objetos. Foram utilizadas algumas técnicas, como a Função de Ativação *Mish* [64] e a arquitetura *Cross Stage Partial* (CSP).

  A função *Mish* é uma função de ativação suave e não monotônica, que pode ser definida a partir da Equação (1), onde ς(x) é a função de ativação *Mish*, descrita na Equação (2).

$$f(x) = x * tanh(\varsigma(x)) \quad (1)$$

$$\varsigma(x) = ln(1 + e^{x}) \quad (2)$$

  A razão pela qual a função de ativação *Mish* é usada na YOLOv4 [6] é seu baixo custo e suas várias propriedades, como sua natureza suave, não monotônica, e ser ilimitada acima e limitada abaixo, que melhoram seu desempenho quando comparada com outras funções popularmente usadas, como a Unidade Linear Retificada (do inglês, *Rectified Linear Unit* - ReLU) [65] e a Ativação da Unidade Linear Ponderada Sigmóide (do inglês, *Sigmoid Weighted Linear Unit Activation* -

Swish) [66, 67]. A Figura 4 mostra o seu desempenho, comparado com as funções ReLU e Swish, em que a função *Mish* apresenta melhores resultados.

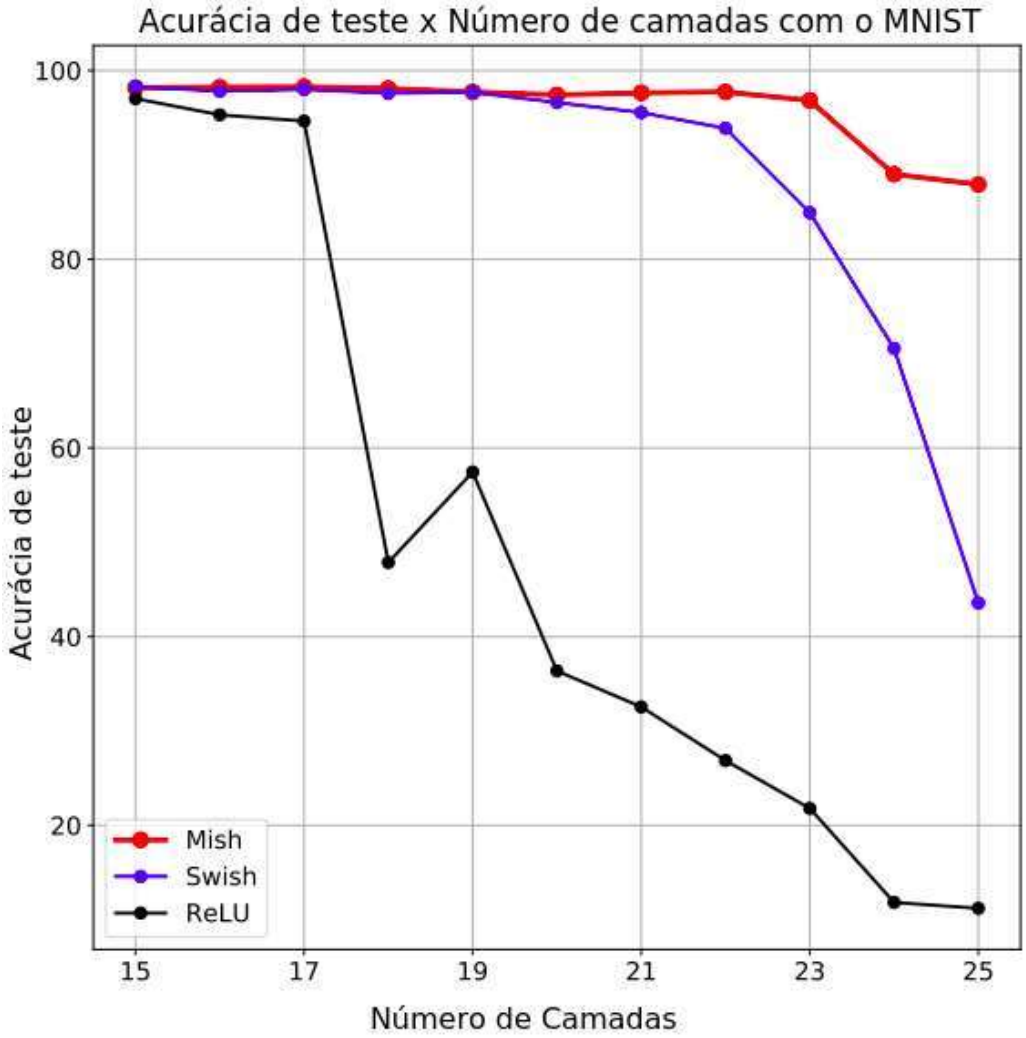



**Figura 4 -** Gráfico comparativo de desempenho das funções de ativação ReLU, Mash e Swish, avaliando a acurácia de teste e o número de camadas.

**Fonte**: Adaptada de [64].

Por fim, foi utilizada a arquitetura *Cross Stage Partial* (CSPDarknet53). A arquitetura é derivada da arquitetura DenseNet [68], que usa a entrada anterior e a concatena com a entrada atual, antes de passar para a camada densa. Essa arquitetura contém 29 camadas convolucionais, com filtro de tamanho 3x3.

O Neck da YOLOv4 utiliza a SPP [69] e uma versão adaptada da PaNet [70]. A camada de *pooling* de Pirâmide Espacial (do inglês, *Spatial Pyramid Pooling* - SPP) foi utilizada para sanar problemas causados pela CNN e Redes Totalmente Conectadas (do

inglês, *Fully Connected Network* - FCN), pois requerem uma imagem de tamanho fixo, porém, ao detectar objetos, as imagens não tem necessariamente essa característica. Ademais, outro problema causado pela CNN é o tamanho da janela deslizante ser constante.

Dessa forma, na saída das CNNs, são gerados os mapas de características, geradas por diferentes filtros. A SPP permite gerar características de tamanho fixo, independente do tamanho dos mapas. Para isto, são usadas camadas de *pooling*, como Max Pooling, que gera diferentes representações dos mapas de características [71].

A PaNet introduziu uma arquitetura que permite uma melhor propagação das informações da camada de baixo para cima ou de cima para baixo. Na implementação original da PaNet, a camada atual e as informações de uma camada anterior são adicionadas para formar um novo vetor.

Na implementação do YOLOv4, uma versão modificada é usada, onde o novo vetor é criado concatenando a entrada e o vetor de uma camada anterior, dessa forma, é possível evitar a perda de informação das camadas iniciais, que extraem informações localizadas de textura e padrão para construir as informações semânticas necessárias nas camadas posteriores [70].

Para avaliação de um modelo de detecção de objetos, geralmente são utilizadas métricas para quantificar a porcentagem de sobreposição entre a máscara de destino e o resultado de detecção do modelo. A métrica de avaliação mais popular usada nos modelos de detecção de objetos é a Interseção pela União, explicada a seguir.

## 2.5 Métricas

Existem métricas de avaliação de classificação binária amplamente utilizadas na literatura. Precisão e interseção pela união são métricas comumente utilizadas para avaliar modelos de detecção de objetos, bem como os modelos da YOLO.

Tendo em vista que uma das etapas do trabalho é realizar a detecção de objetos, temos duas possibilidades de resposta da rede, sendo positiva, para quando o modelo detecta a caixa delimitadora do objeto presente na imagem, e negativa, para quando o modelo não detecta nenhum objeto.

Por fim, as métricas utilizadas neste trabalho para a etapa de treinamento do modelo foram: precisão, revocação, precisão média e interseção pela união. Enquanto que,

para a etapa de validação do *pipeline* proposto, foram avaliados aspectos como tempo de execução do *pipeline*, a confiança média do modelo para o objeto de detecção e o percentual de informação excedente.

As métricas utilizadas neste trabalho foram: precisão, revocação, precisão média e IoU.

### 2.5.1 Precisão e Revocação

Essas métricas não foram utilizadas individualmente. Para o cálculo dessas métricas, são utilizadas as frequências de classificação para cada classe do modelo, sendo elas:

- Verdadeiro positivo (do inglês, *true positive* - TP): ocorre quando exemplos são classificados como positivo de maneira correta;
- Verdadeiro negativo (do inglês, *true negative* - TN): ocorre quando exemplos são classificados como negativos de maneira correta;
- Falso positivo (do inglês, *false positive* - FP): ocorre quando exemplos são classificados como positivo de maneira incorreta;
- Falso negativo (do inglês, *false negative* - FN): ocorre quando exemplos são classificados como negativos de maneira incorreta.

A precisão é a relação entre os verdadeiros positivos, e o total de previsões classificadas como positivas, sendo os verdadeiros positivos e falsos positivos. Ou seja, a precisão indica a porcentagem dos resultados que são relevantes e pode ser descrita na Equação (3):

$$Precisão = \frac{TP}{TP + FP} \tag{3}$$

A revocação, ou *recall*, mede a capacidade do modelo de detectar amostras positivas. A revocação pode ser considerada como a capacidade de um modelo de encontrar todos os pontos de dados de interesse em um conjunto de dados, ou seja, refere-se à porcentagem do total de resultados relevantes classificados corretamente por seu algoritmo e é apresentada na Equação (4) :

$$Revocação = \frac{TP}{TP + FN} \quad (4)$$

As métricas de precisão e revocação auxiliam na verificação do funcionamento correto do modelo que será utilizado, modelos que geralmente são supervisionados em aprendizagem de máquina. Os testes de métricas elucidam o conceito de qualidade do que está sendo desenvolvido, e para isso as métricas são utilizadas.

### 2.5.2 Precisão média

A métrica de precisão refere-se ao modelo em um determinado limiar de decisão. Especialmente se as classes não estão balanceadas, ou, se deseja favorecer a precisão em relação à revocação ou vice-versa, pode-se variar este limiar [72]. Vale ressaltar que estamos nos referindo a uma aproximação da área embaixo da curva de precisão/revocação.

A precisão média é uma forma de resumir a curva de precisão/revocação em um único valor que representa a média de todas as precisões. Essa métrica é calculada de acordo com a Equação (5), onde é representada pela soma ponderada das precisões em cada limiar em que o peso é o aumento na revocação pode ser calculada como AP (Equação 5) .

$$AP = \sum_{k=0}^{k=n-1} [Revocação(k) - Revocação(k+1)] * Precisão(k) \quad (5)$$

### 2.5.3 Interseção pela União

A IoU é uma métrica de avaliação, comumente utilizada para medir a qualidade de um modelo de detecção de objeto, em um conjunto de dados específico. Frequentemente, vemos essa métrica de avaliação usada em desafios de detecção de objetos, como o PASCAL VOC [73].

As caixas delimitadoras rotuladas do conjunto de teste especificam onde o objeto está na imagem, portanto, a IoU avalia o desempenho do modelo de detecção verificando a interseção entre a caixa delimitadora da detecção e do objeto previamente rotulado, pela área total de ambas as caixas delimitadoras, como mostra a Figura 5.

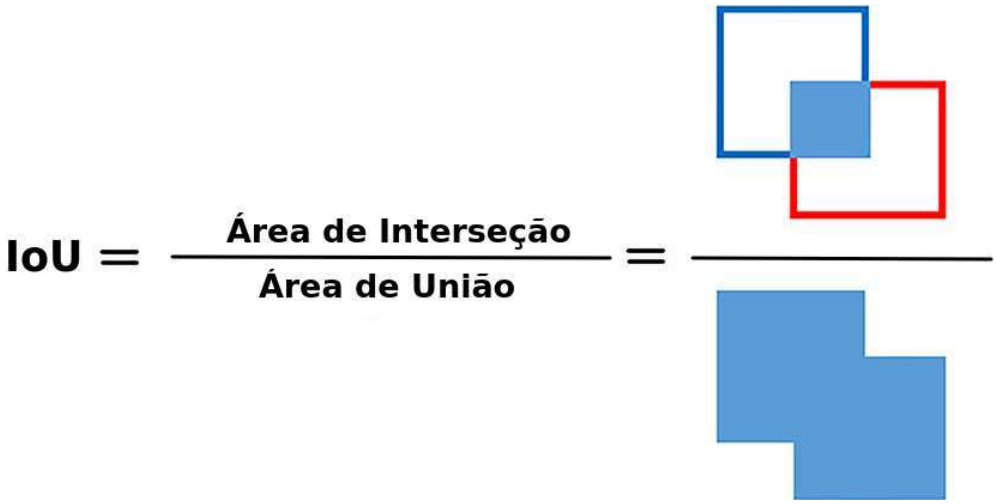



**Figura 5 -** Representação da Interseção pela União para detecção de objetos.

**Fonte**: Adaptado de [73].

O valor da IoU pode variar entre 0 e 1, dessa forma, quanto maior for o valor da interseção, melhor é o desempenho do modelo. Essa métrica pode ser descrita de duas formas diferentes, realizando o cálculo das áreas, como foi demonstrado na Figura 5 e, também pode ser representada pela Equação 6.

Utiliza-se valores de frequências de classificação, mencionados anteriormente, onde a área de interseção pode ser representada pelo número de exemplos classificados como verdadeiro positivo (TP), e a área de união pela soma dos verdadeiros positivos (TP), com os falsos positivos (FP) e falsos negativos (FN).

$$IoU = \frac{TP}{TP + FP + FN} \qquad (6)$$

# 3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo apresentar uma seleção de pesquisas que, de alguma forma, trouxeram contribuições para a literatura no uso de CNNs para detecção de objetos em vídeos. A busca por estes trabalhos foi realizada em portais acadêmicos como o Google Acadêmico [74], SciELO [75] e IEEE Xplore [76].

Para a etapa de pós-processamento realizado no vídeo de entrada, responsável pela aproximação do objeto detectado e recorte para a criação de um novo vídeo de saída, não foi encontrada nenhuma pesquisa que se assemelhasse a este trabalho. Desta forma, este capítulo abordará apenas pesquisas relacionadas ao processo de detecção de objetos em vídeos, utilizando diversas abordagens.

Dewi et al. (2019) [77] propuseram uma nova estrutura de rede da YOLO para a detecção, chamada YOLO-GAN, ou YOLO *Generative Adversarial Network.* Rede neural *adversarial* (do inglês, *Generative Adversarial Network* - GAN) [34] é um *framework* de aprendizagem profunda no qual dois modelos, sendo um generativo e um discriminativo, são treinados simultaneamente. O modelo discriminativo visa julgar se uma amostra é de dados reais ou dados falsos, enquanto o modelo generativo captura a distribuição de alguns dados alvo com o objetivo de confundir o modelo discriminativo [78].

Na metodologia proposta por Dewi et al. (2019) [77], foi adotada uma arquitetura baseada na GAN [34], que gera uma nova imagem a partir do processo Gerador. O processo Gerador supervisionado aprende cada instrumento musical entendido como semelhante a outros instrumentos musicais e o discriminador consiste em uma arquitetura de classe YOLO.

O modelo é responsável por aprender a diferença de confiança entre a saída do gerador e discriminador para definir um limiar de confiança para detectar se a imagem é da classe correta ou semelhante, e, por fim, a YOLO é responsável por fazer a detecção dando o rótulo de cada instrumento e a caixa delimitadora.

Foi utilizada uma base de dados criada por Yao e Fei-Fei (2010) [79], chamada PPMI, que consiste em imagens de pessoas tocando instrumentos e contém 12 classes. Os resultados obtidos foram comparados ao desempenho da YOLOv2, utilizando as mesmas condições de treinamento e teste, medindo a acurácia para cada classe. O sistema proposto supera a YOLOv2 para a detecção de instrumentos similares.

Slizovskaia, Gomez e Haro (2017) [80] propuseram um método para reconhecer instrumentos musicais em vídeos gerados pelo usuário, utilizando modelos para a tarefa de classificação de instrumentos musicais multimodais, combinando áudio e imagem para o treinamento e validação. Para o reconhecimento em imagem foi utilizada a arquitetura convolucional profunda do Inception V3 desenvolvido pelo Google [81].

Para o desenvolvimento do modelo multimodal foram treinados individualmente os modelos de representação de áudio e vídeo, e, em seguida, foram explorados os atributos aprendidos das últimas camadas das redes para treinar e avaliar o modelo conjunto.

Os vídeos utilizados neste trabalho foram extraídos das bases de dados Fudan-Columbia Video Dataset (FCVID) [82] e o YouTube-8M [83], com 13 classes e um total de 60 mil vídeos com uma duração total de aproximadamente 4 mil horas.

No trabalho proposto por Corovic (2018) [84] é apresentado um sistema inteligente de rastreamento de veículos, onde foi feita uma detecção e classificação de veículos em vídeos de câmeras de segurança. Neste trabalho, foi utilizada uma arquitetura aprimorada da YOLOv3 [51] e o vídeo é processado a cada quadro, para que seja feita a detecção de todos os objetos.

Foram utilizadas diferentes bases de dados para o treinamento do modelo, tais como, Pascal VOC [73], *Open Image* [85] e COCO *dataset* [86] , além de uma base de dados personalizada. Para a criação desse banco de dados, foi utilizado um kit de ferramentas, OIDv4, que faz o download de imagens do website Open Images Dataset [85] e usa comandos predefinidos para a criação dessa base. De acordo com os resultados obtidos, apesar de haver problemas com falsos positivos, o sistema proposto obteve desempenho satisfatório.

Abhinand et al. (2021) [87] propuseram um sistema de alerta de emergência em estações de metrô que detecta quando uma pessoa cruza a linha amarela de segurança sem autorização.

Inicialmente, para detectar se uma pessoa cruzou a faixa, é feita a detecção da faixa em si, utilizando uma técnica de filtragem de cor para recuperar suas coordenadas. Dessa forma, o modelo faz a detecção das pessoas presentes no vídeo para comparar com as coordenadas da faixa. Após essa detecção, é feita outra detecção para verificar se o trem está parado na estação. Caso o trem não seja detectado por dois quadros consecutivos, o alerta é emitido.

O modelo utilizado no desenvolvimento deste trabalho foi o modelo pré-treinado YOLOv3 com a base de dados COCO *dataset*, realizando detecção em vídeos em tempo real.

Na metodologia desenvolvida por Aung, Bobkov e Tun (2021) [88] é apresentado um modelo híbrido de detecção de faces em vídeos em tempo real, dividido em três etapas. Na primeira etapa, é feita a rotulação das imagens da base de dados, enquanto que a segunda etapa é responsável pela extração dos padrões, ou *features*, e, por fim, é feita a detecção da face.

As imagens utilizadas no trabalho foram extraídas da base de dados FDDB [89] e a rotulação foi feita utilizando uma ferramenta manual. Para a extração de *features*, foi utilizada a arquitetura adaptada da CNN VGG-16 pré-treinada, contendo 41 camadas diferentes.

Por fim, na última etapa, a saída da rede VGG-16 é combinada com o algoritmo YOLOv2 para detecção de faces. Foi utilizada a arquitetura original na YOLOv2, usando o tamanho da imagem de entrada de 480x480 com codificação RGB.

Na Tabela 1 é possível ver uma comparação entre os métodos e objetivos dos diferentes trabalhos relacionados apresentados nesta seção e o presente trabalho.

| Autores | Descrição | Metodologia | Base de Dados |
|---|---|---|---|
| Dewi et al. (2019) | Detecção de instrumentos musicais com um novo modelo proposto, baseado na YOLO, utilizando apenas imagens para o treinamento. | YOLO-GAN | PPMI |
| Slizovskaia, Gomez e Haro (2017) | Reconhecimento através de imagem e áudio de instrumentos musicais em vídeos gerados pelo usuário utilizando um modelo multimodal. | Inception V3, CNN Profunda, FCN. | FCVID e YouTube-8M. |
| Corovic et al. (2018) | Detecção e classificação de veículos através de câmeras de segurança. | YOLOv3 | Pascal VOC, COCO dataset e Open Images Dataset |

| | | | |
|---|---|---|---|
| Abhinand et al. (2021) | Detecção de pessoas em um sistema de alerta de emergência em estações de metrô para evitar que a linha de segurança da estação seja ultrapassada indevidamente. | YOLOv3 | Pascal VOC |
| Aung, Bobkov e Tun (2021) | Detecção de faces, utilizando um modelo híbrido com duas arquiterutas. Para a extração de features foi utilizada a VGG-16 e para a detecção foi utilizado o modelo YOLOv2 | VGG-16 e YOLOv2 | FDDB dataset |
| Trabalho Atual | Detecção de instrumentos musicais em vídeo, utilizando o modelo YOLOv4, destacando um instrumento para a criação de um novo vídeo dando foco ao instrumento detectado. | YOLOv4 | Imagenet |

**Tabela 1 -** Comparação entre os diferentes trabalhos relacionados e o presente trabalho.

**Fonte**: Do Autor.

Neste capítulo, foi possível observar diferentes abordagens para o problema de detecção de objetos, novas propostas de modelos e como diferentes tipos de representação de dados influenciam na escolha e aplicação de diferentes metodologias.

Para o presente trabalho a metodologia escolhida foi a arquitetura YOLO, que se destaca diante dos outros modelos para detecção de objetos, mencionados acima, pois utiliza uma base de dados de apenas imagens para treinar o modelo.

# 4 METODOLOGIA

A metodologia proposta foi dividida em três etapas, sendo a primeira para o desenvolvimento de um sistema que, utilizando um modelo treinado da YOLO, seja capaz de identificar e detectar objetos e fazer um procedimento semelhante a um *zoom in* no vídeo, para dar maior visibilidade a esse objeto. Após a implementação, foi elaborada uma validação para esse *pipeline*, com um modelo pré-treinado, onde o problema de detecção foi minimizado, ou seja, os objetos detectados não fazem parte do escopo de instrumentos musicais. A segunda etapa apresentará o método de validação do *pipeline* e, por fim, a terceira etapa está focada no treinamento de uma CNN para a detecção de instrumentos musicais.

## 4.1 Dependências e *Hardware*

O hardware utilizado neste trabalho foi um Notebook Acer com processador Intel® CoreTM i7-7500 2.7 GHz, com capacidade de disco de 2 TB e 8GB de memória RAM e uma GPU NVIDIA GeForce® 940MX. Ademais, foi utilizada para treino e teste uma ferramenta em nuvem disponibilizada pelo *Google*, o *Google Collaboratory*, com uma GPU NVIDIA Tesla T4.

## 4.2 Implementação do *Pipeline*

Nesta etapa, serão apresentados os recursos e métodos utilizados para o desenvolvimento do algoritmo proposto.

Os parâmetros de configuração da rede também são parametrizados no nosso sistema, de forma que seja possível realizar a detecção de qualquer objeto, dependendo apenas da configuração da rede passada pelo usuário.

As etapas de execução do sistema podem ser representadas na Figura 6, onde é exibido um fluxograma de funcionamento do código, que serão detalhadas nas subseções subsequentes.


**Figura 6 -** Fluxograma de funcionamento do código desenvolvido neste trabalho.

**Fonte**: Do Autor.

### 4.2.1 Fragmentação do vídeo em quadros

Primeiramente, é feita a extração de todos os quadros do arquivo de entrada, seguindo a taxa de FPS correspondente ao vídeo, sendo esses quadros em formato .JPG. Em conseguinte, foi feita a extração do áudio para um arquivo de mídia no formato desse mesmo vídeo, que, dependendo dele, podendo ser .AAC, por exemplo.

Para a extração dos quadros, foi utilizado o *framework* multimídia FFmpeg, onde foram testados diferentes parâmetros para a extração, descritos na Seção 4.3, com o intuito de avaliar a capacidade de detecção do modelo, baseado na qualidade da imagem extraída.

### 4.2.2 Detecção dos objetos

Na etapa de detecção dos objetos utilizando o modelo, inicialmente, foi testada a detecção em todos os quadros dos vídeos de entrada, porém, a detecção dura em média 0,5 segundo por imagem, ou seja, considerando um vídeo de 2 minutos, com 30 FPS, demoraria cerca de uma hora apenas na etapa de detecção, sem contar as demais.

Dessa forma, para otimizar a execução do programa, foi definida uma estratégia de detecção fracionada, ou seja, sendo feita a cada *N* quadros, sendo esse valor parametrizado pelo usuário.

Para os nossos testes, foram considerados dois intervalos, de 12 e 24 quadros, e, avaliando tempo de execução e fluidez, foi escolhido o intervalo de 12 quadros para a detecção. Caso haja uma alteração muito brusca entre detecções, como uma mudança de posicionamento de câmera nesse intervalo, que corresponde a meio segundo, não geraria uma perda relevante para a fluidez.

Essa detecção gera um arquivo, no formato .JSON, de todos os objetos identificados e as suas respectivas posições em cada imagem. Cada detecção aponta o ponto inicial nos eixos X e Y da imagem e a largura e altura da detecção, bem como a confiança do modelo para a classe detectada. Essas detecções são conhecidas como caixas delimitadoras.

### 4.2.3 Corte das caixas delimitadoras

O objeto de detecção é parametrizado para o usuário, através dele escolhe-se o objeto que deseja extrair do vídeo, e, a partir dos dados colhidos no .JSON, é realizada a extração de uma caixa por imagem.

Como o modelo utilizado neste trabalho não diferencia objetos de uma mesma classe na mesma detecção, não há como diferenciá-los na etapa de recorte. Por isso, se houver duas ou mais detecções do mesmo objeto no mesmo quadro, é feita uma verificação para selecionar a caixa mais à esquerda da imagem, garantindo que, na maioria dos casos, o mesmo objeto vai ser selecionado, desde que não haja mudança no posicionamento da câmera.

Para melhorar a fluidez do vídeo, foi feita uma estabilização entre os *pixels* das detecções, que ocorrem a cada 12 quadros, como justificado na Subseção 4.2.2, tanto nas coordenadas, quanto no tamanho. Dessa forma, caso haja variação entre o tamanho e posição das caixas de detecção, o recorte deve acontecer gradualmente entre elas.

A estabilização leva em consideração as posições do quadro atual *N* e do seguinte *N + 12*. Caso, no quadro seguinte, não tenha nenhuma detecção do objeto escolhido, o algoritmo busca até dois quadros à frente *N + 36* para fazer a estabilização. Essa verificação é feita a cada passo. Caso não tenha nenhuma detecção por quatro quadros consecutivos, o recorte é feito gradualmente para o tamanho original do quadro.

### 4.2.4 Criação do vídeo de saída

Todos os recortes extraídos na etapa anterior são agrupados em um novo vídeo, no formato .MP4, com o áudio extraído do vídeo de entrada.

Utilizando a ferramenta FFmpeg, foi feita uma cópia do vídeo de entrada, redimensionando-a para 15% do seu tamanho, seguindo as proporções de largura e altura, para que não houvesse distorção. Em seguida, foi feita a sobreposição, no canto superior esquerdo do vídeo de saída, com a cópia redimensionada.

O intuito dessa sobreposição foi que o usuário pudesse comparar o vídeo recortado resultante do sistema com o vídeo original de entrada.

Esse valor foi definido após testes com as porcentagens de redimensionamento de 10%, 15%, 20%, 25% e 30%, sendo 15% a taxa que proporcionou melhor visualização e causou menor obstrução do objeto recortado no vídeo final.

## 4.3 Parâmetros FFmpeg

Ao longo dos testes, foi possível perceber que a qualidade dos quadros extraídos influenciava diretamente no poder de detecção da rede pois, quando havia muitos artefatos na imagem, a rede não conseguia identificar o objeto desejado. Dessa forma, para suavizar as imagens, foram testadas diferentes combinações de parâmetros com a biblioteca FFmpeg para extrair os quadros, bem como, foram definidos parâmetros para gerar o vídeo final. A proposta é sempre trabalhar com imagens e vídeos com melhor resolução, para diminuir a presença de artefatos.

Os parâmetros testados nesta etapa do trabalho foram:

- *crf* ou Fator de taxa constante (do inglês, *Constant Rate Factor*) - esse valor representa uma variedade de parâmetros de compactação, que determinam o nível

de qualidade. Este método permite que o codificador tente obter uma determinada qualidade de saída para todo o arquivo. O parâmetro *CRF* varia entre 0 e 51, onde 0 é considerado sem perdas, 23 é o padrão e 51 é a pior qualidade. Um valor mais baixo geralmente leva a uma qualidade mais alta. De acordo com a documentação da biblioteca, 17 ou 18 é considerado como visualmente sem perdas, por isso, o valor escolhido para a criação do vídeo de saída foi 18 [90];

- *qscale:v* - esse parâmetro direciona o nível de qualidade na extração com uma escala que varia entre 1 e 30. Este é um modo de fluxo de transferência de bits variável (do inglês, *bitrate*), de forma que não seja fixado um valor específico de *bitrates* para a extração. 1 é a qualidade mais alta e maior tamanho de arquivo e 31, a qualidade mais baixa e menor tamanho de arquivo. De acordo com a documentação, apesar de o valor 1 ser a melhor qualidade de extração, o resultado é visualmente idêntico ao 2. Caso esse parâmetro não seja especificado, é definido um valor padrão de 200k de *bitrate*, gerando uma imagem de baixa qualidade;
- *preset* - esse parâmetro fornece a velocidade de codificação para a taxa de compactação. Quanto mais lenta a codificação, melhor será a compactação, resultando em uma melhor qualidade. Os valores de *preset* disponíveis em ordem decrescente de velocidade são: *ultrafast*, *superfast*, *veryfast*, *faster*, *fast*, *medium*, *slow*, *slower*, *veryslow*. Onde *medium* foi o valor escolhido para a criação do vídeo de saída, pois é o padrão.
- *vframes* - esse parâmetro é utilizado para forçar a resolução do vídeo de saída. Como as imagens que compõem o vídeo de saída possuem tamanhos diferentes, esse parâmetro garante que as imagens sejam redimensionadas para o mesmo tamanho, podendo causar distorção. A resolução atribuída foi 1920x720.

Foram avaliados 4 cenários diferentes, todos com a mesma configuração para gerar o vídeo final, variando apenas os parâmetros na extração dos quadros do vídeo. O propósito é gerar imagens melhores para que o modelo treinado consiga identificar mais objetos e na classe correta, ponderando alguns pontos, que serão mencionados nas seções subsequentes.

Parâmetros de extração:

- Cenário 1: *-q:v 1* - cenário para avaliar a detecção nas imagens extraídas com maior qualidade possível;

- Cenário 2: *-q:v 2* - cenário com o valor mínimo recomendado, pois o resultado é visualmente idêntico ao 1;
- Cenário 3: *-q:v 5* - um valor "intermediário" que oferece menor qualidade, mas pouco perceptível.
- Cenário 4: sem especificar o parâmetro de qualidade.

## 4.4 Validação do *Pipeline*

Para a validação do *pipeline* proposto neste trabalho, foi utilizado um modelo de detecção de 80 classes pré-treinado e disponibilizado por Alexey [6]. Esse modelo foi treinado com o MS COCO (*Microsoft Common Objects in Context*), um conjunto de dados de detecção, segmentação e detecção de pontos-chave em grande escala, com mais de 330 mil imagens em seu acervo [91].

Para a validação, parte-se do pressuposto que a etapa de detecção funciona de forma eficiente, para que fosse possível testar as demais. Dessa forma, foi avaliada a confiança média do modelo, ao longo das detecções, em cada cenário proposto para extração dos quadros do vídeo.

Outro aspecto avaliado na validação foi a heurística desenvolvida na etapa de recorte, com a estabilização dos *pixels* e o desempenho do sistema ao lidar com problemas de oclusão parcial ou total e movimentação do objeto ao longo do vídeo. Sendo assim, foi utilizado este modelo para detecção de objetos arbitrários.

O modelo escolhido para a validação não apresenta no seu escopo a detecção de instrumentos musicais, por isso, as classes escolhidas foram os animais gato e cachorro, pois seria mais fácil avaliar o aspecto da oclusão e movimentação. Foram testados 7 vídeos de entrada[1], descritos em detalhes abaixo, com 2 objetos de detecção diferentes.

- Vídeo 01: O objeto de detecção deste vídeo é o cachorro. É um vídeo de uma exposição internacional de cães, com câmera dinâmica, onde o objeto de detecção, em alguns momentos, tem oclusão parcial e total e também sai do enquadramento da câmera, permanecendo fora por alguns segundos. A resolução do vídeo é de 640*x*360 e o tempo de duração do vídeo é de 301 segundos.

[1] Link para os vídeos

- Vídeo 02: O objeto de detecção deste vídeo é o cachorro. É um vídeo de câmera doméstica, e o cenário é estático. O objeto de detecção se movimenta por todo o ambiente, ficando parcial e totalmente ocluso, em alguns momentos, e também sai do enquadramento da câmera. A resolução do vídeo de entrada é de 1280*x*720 e o tempo de duração do vídeo é de 158 segundos.
- Vídeo 03: O objeto de detecção deste vídeo é o cachorro. É um vídeo gravado em *smartphone*, onde o próprio usuário aproxima e afasta a imagem com *zoom* da câmera, com oclusão parcial em vários momentos, e com telas pretas com texto ao longo do vídeo. A resolução do vídeo é de 1280*x*720 e o tempo de duração do vídeo é de 213 segundos.
- Vídeo 04: O objeto de detecção deste vídeo é o cachorro. É um vídeo gravado por uma câmera doméstica, e o cenário permanece estático durante todo o vídeo. O cachorro se movimenta pelo ambiente, ficando parcial e totalmente ocluso, em alguns momentos, e também sai do enquadramento da câmera. A resolução do vídeo de entrada é de 1280*x*720 e o tempo de duração do vídeo é de 81 segundos.
- Vídeo 05: O objeto de detecção deste vídeo é o cachorro. É um vídeo de câmera doméstica e o cenário permanece estático ao longo dele. Neste vídeo existem dois cachorros, que estão em constante movimento, e permanecendo por alguns segundos em oclusão parcial e total e também saindo do enquadramento do vídeo. A resolução do vídeo é de 1920*x*1080 e o tempo de duração do vídeo é de 233 segundos.
- Vídeo 06: Este vídeo tem dois objetos de detecção, cachorro e gato, e foi gravado com um *smartphone*, com pouco movimento de câmera. Ambos os animais estão sempre no enquadramento, ocasionalmente com oclusão parcial. A resolução do vídeo é de 1920*x*1080 e o tempo de duração do vídeo é de 66 segundos.
- Vídeo 07: Este vídeo tem dois objetos de detecção, cachorro e gato, e foi gravado com um *smartphone*. A câmera e os animais se movimentam ao longo do cômodo, fazendo com que os animais fiquem com oclusão parcial e/ou total em vários momentos do vídeo. A resolução do vídeo de entrada é de 1920*x*1080 e tem 209 segundos de duração.

A Tabela 2 sintetiza as informações dos vídeos de entrada, em relação ao objeto de detecção, à resolução por quadro, ao tamanho original por *pixel* por frame, ao tempo total

do vídeo em segundos e à quantidade total de quadros avaliados na detecção, sendo eles, apenas 1/12 do total de quadros do vídeo.

| Entrada | Objeto de Detecção | Descrição do Vídeo | Tempo de duração | Resolução de Entrada | Tamanho do Vídeo Original | Total de frames avaliados |
|---|---|---|---|---|---|---|
| Vídeo 01 | Cachorro | Vídeo de uma exposição de cães. Câmera dinâmica e cachorro em movimento. | 301 | 230400 | 2073600000 | 750 |
| Vídeo 02 | Cachorro | Vídeo de câmera de segurança doméstica. Câmera estática e cachorro andando pelo ambiente. | 158 | 921600 | 4340736000 | 393 |
| Vídeo 03 | Cachorro | Vídeo de câmera de celular. Câmera dinâmica e cachorro andando pelo ambiente. | 213 | 921600 | 5861376000 | 530 |
| Vídeo 04 | Cachorro | Vídeo de câmera de segurança doméstica. Câmera estática e cachorro andando pelo ambiente. | 81 | 921600 | 1360281600 | 123 |
| Vídeo 05 | Cachorro | Vídeo de câmera de segurança doméstica. Câmera estática e cachorro andando pelo ambiente. | 233 | 2073600 | 12078720000 | 485 |
| Vídeo 06 | Cachorro | Vídeo de câmera de celular. Câmera dinâmica com pouco movimento, gato e cachorro interagindo entre si. | 66 | 2073600 | 4043520000 | 163 |
| | Gato | | | | | |
| Vídeo 07 | Cachorro | Vídeo de câmera de celular. Câmera dinâmica, gato andando pelo ambiente e cachorro imóvel na maior parte do vídeo. | 209 | 2073600 | 13001472000 | 523 |
| | Gato | | | | | |

**Tabela 2**: Informações dos vídeos de entrada utilizados para validação do *pipeline*, expondo o objeto de detecção, breve descrição, tempo de duração em segundos, resolução de entrada por quadro, tamanho do vídeo original por *pixel* no vídeo inteiro e total de quadros avaliados na detecção dos vídeos.
**Fonte**: Do Autor.

## 4.5 Treinamento da Rede para Detecção Personalizada

Nesta seção, é detalhado o conjunto de dados e as configurações utilizadas na CNN, com a YOLOv4, para a realização do treinamento para detecção de instrumentos musicais. Em conseguinte, como foi feito o tratamento da base de dados utilizada e sobre a divisão para o conjunto de treinamento e teste. Por fim, serão apresentadas as métricas utilizadas para avaliar o modelo.

### 4.5.1 Base de dados

No presente trabalho foi utilizada a base de dados do projeto ImageNet [43], utilizada em pesquisas de softwares de reconhecimento visual de objetos. Esta base possui

um acervo com mais de 14 milhões de imagens e 20.000 classes, todas rotuladas manualmente. Essas imagens não são propriedade da ImageNet, e trata-se de um banco de anotações de URLs de imagens de terceiros que está disponível gratuitamente.

Desde 2010, o projeto ImageNet realiza uma competição anual, o Desafio de Reconhecimento Visual de Grande Escala do ImageNet (do inglês, *ImageNet Large Scale Visual Recognition Challenge* - ILSVRC). No ILSVRC, as equipes avaliam seus algoritmos sobre uma parcela da base desconhecida aos competidores e competem em várias tarefas de reconhecimento visual. A base utilizada neste trabalho foi a publicada mais recentemente, disponibilizada para o evento ILSVRC 2017.

As imagens utilizadas neste trabalho pertencem ao domínio público e vêm de diversas origens para fazer parte do ImageNet, dessa forma, as imagens podem possuir resoluções diferentes. Porém, não foi necessário fazer um pré-processamento dessas imagens, pois a YOLOv4 é responsável por redimensioná-las para um tamanho pré-definido, sendo assim, todas as imagens são redimensionadas para o tamanho 416 x 416 (largura x altura).

A priori, a base contém apenas 3 classes de instrumentos, sendo elas Guitarra com 121 imagens, Violão com 311 imagens, Microfone com 341 imagens.

Cada imagem da base de dados possui um rótulo ou anotação própria, em formato Pascal VOC, que é o disponibilizado pela ImageNet. O formato Pascal VOC foi originalmente criado para o desafio Pascal Visual Object Classes (VOC) [73], que consiste em realizar o reconhecimento e detecção de categorias de objetos. Esse formato é amplamente utilizado para rótulos de detecção de objetos, mas não é um formato reconhecido pelo modelo.

Em conseguinte, foi utilizada uma plataforma online, Roboflow [92], para realizar a conversão desses rótulos para o formato YOLO Darknet TXT, que consiste em um arquivo de texto por imagem (contendo as anotações e uma representação numérica do rótulo) e um mapa de rótulos que mapeia os IDs numéricos para strings legíveis por humanos. As anotações são normalizadas para ficarem dentro do intervalo [0, 1].

### 4.5.2 Conjunto de Treinamento e Teste

Para o treinamento da rede foram utilizadas um total de 773 imagens, cada uma delas podendo possuir mais de um elemento ou classe rotulada. A base foi dividida em

duas partes, denominadas de treinamento e teste. A amostra de treinamento é usada para treinar a rede, enquanto que a amostra de teste é utilizada para medir o desempenho do modelo.

A divisão foi feita de forma aleatória, através da plataforma que realizou a conversão, Roboflow [92], e seguiu a seguinte proporção: 70% dos dados para treinamento e 30% dos dados para teste, totalizando 550 e 223 imagens para cada etapa, respectivamente. As proporções das classes para cada conjunto seguem os valores apresentados na Tabela 3.

| Conjunto | Classes | | |
|---|---|---|---|
| | Guitarra | Violão | Microfone |
| Treinamento | 87 | 219 | 244 |
| Teste | 34 | 92 | 97 |

**Tabela 3 -** Disposição de cada classe para os conjuntos de treinamento e teste.

**Fonte**: Do Autor.

### 4.5.3 Rede Neural Convolucional

Para a detecção dos instrumentos foi utilizada a YOLOv4, uma biblioteca de AM desenvolvida em linguagem C que utiliza Redes Neurais, baseada na Darknet-53.

As configurações para o treinamento da rede devem ser modificadas de acordo com os dados utilizados. Neste trabalho, foram utilizadas 3 classes e, por isso, foram feitas as alterações detalhadas abaixo:

- *batch* = 64, que representa o número de amostras, ou lote, que serão usadas antes de atualizar os parâmetros do modelo interno;
- *subdivisions* = 16, os lotes serão divididos em 16 mini lotes, onde cada mini lote deve conter 4 amostras;
- *max_batches* = 15.000. Essa variável representa o número total de iterações, ou épocas.
- *steps* = 12.000, 13.500. Esses valores indicam em qual lote as taxas de aprendizagem serão atualizadas;
- *filters* = 24, onde foi alterado o número de filtros apenas nas camadas convolucionais.

Alexey (2020) [6] sugere a definição desses parâmetros, sendo o *batch* e o *subdivisions* valores fixos, definidos em 64 e 16, respectivamente. E, para os parâmetros *max_batches*, *steps* e *filters*, são definidos valores baseados no número total de classes para que o modelo está sendo treinado.

Para o parâmetro *max_batches* deve ser definido um valor de acordo com o número total de classes (nº de classes * 2.000) e, para o problema atual, seriam 6.000 iterações. Porém, o autor também sugere que o melhor valor para a perda média (avg loss) seja abaixo de 0,5, porém, com 6.000 iterações, a perda média ainda estava em 0,61.

Dessa forma, optou-se por prosseguir com o treinamento até 15.000 iterações, com o intuito de reduzir o valor da perda média. O menor valor para a perda média foi na iteração 13.000, chegando a 0,52. Na Figura 7, observa-se a evolução da perda média ao longo do treinamento a cada 1.500 iterações.

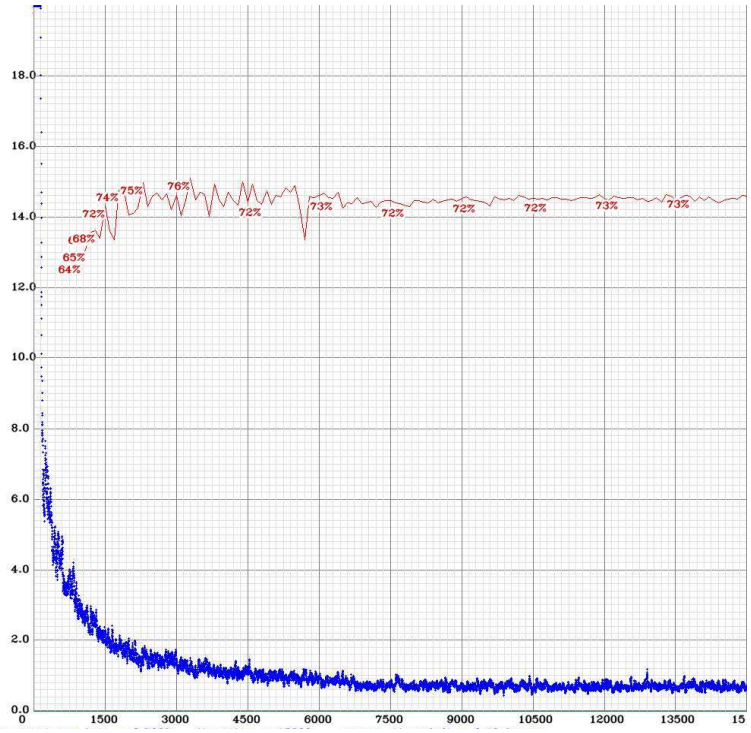



**Figura 7 -** Desempenho do modelo no treinamento, avaliando a perda média e o mAP. Nos eixos x e y são representados a quantidade de iterações do modelo e a perda média, respectivamente. A linha azul representa a evolução da perda média e a linha vermelha os valores obtidos pela métrica mAP.

**Fonte**: Do Autor.

No parâmetro *steps* são atribuídos dois valores que correspondem a 80% e 90% do *max_batch*. E, por fim, o parâmetro *filters* deve ser ajustado de acordo com o número de classes, respeitando a proporção [(nº de classes + 5) * 3].

Esses parâmetros foram definidos para o treinamento da rede, onde foram utilizados pesos pré-treinados da YOLOv4 como ponto de partida e, a cada 1.000 iterações, os pesos são atualizados. Dessa forma, os pesos utilizados para realizar as detecções em vídeo foram os correspondentes a iteração 13.000.

## 4.6 Aplicação em instrumentos musicais

Para a etapa de aplicação em instrumentos musicais, foi utilizado o modelo treinado apresentado na Seção 4.5. Foram testados 6 vídeos de entrada[2], que serão descritos em detalhes abaixo, com os três objetos de detecção, sendo eles, Guitarra, Violão e Microfone.

- Vídeo 01: Os objetos de detecção neste vídeo são o violão e o microfone. É um vídeo de apresentação em uma espécie de palco, onde o instrumentista permanece sempre sozinho e com o enquadramento nele e há a aparição de apenas um violão e um microfone ao longo do vídeo. Este vídeo foi gravado na vertical com câmera de *smartphone*, com a gravação trêmula. Os objetos permanecem enquadrados na maior parte do vídeo, sem sofrer oclusão. A resolução do vídeo é de 720*x*406 e o tempo de duração é de 229 segundos.
- Vídeo 02: Os objetos de detecção neste vídeo são o violão e o microfone. É um vídeo gravado em um programa de televisão, onde o posicionamento de câmera varia bastante. Há vários microfones ao longo do vídeo e, em vários momentos, ambos o violão e o microfone não estão enquadrados ou permanecem com oclusão parcial. A resolução do vídeo é de 1920*x*1080 e o tempo de duração do vídeo é de 267 segundos.
- Vídeo 03: Os objetos de detecção neste vídeo são o violão e o microfone. Este vídeo foi gravado em estúdio, com dois instrumentistas, mas apenas um violão foi capturado na gravação. Ocasionalmente, o posicionamento da câmera muda para o

[2] Link para os vídeos

microfone, tirando o enquadramento do segundo objeto. A resolução do vídeo é de 1080*x*720 e o tempo de duração é de 524 segundos.

- Vídeo 04: Os objetos de detecção neste vídeo são o violão, a guitarra e o microfone. Apesar do instrumento presente neste vídeo ser o violão, o modelo não reconheceu bem a classe correta, por isso, foi testado o desempenho de recorte do *pipeline* para as classes violão e guitarra. As condições similares ao Vídeo 01, com a diferença de ter sido gravado na horizontal. Os objetos permanecem enquadrados na maior parte do vídeo, sem sofrer oclusão. A resolução do vídeo é de 1920*x*1080 e o tempo de duração é de 175 segundos.
- Vídeo 05: Os objetos de detecção neste vídeo são o violão e o microfone. Este vídeo foi gravado em um espetáculo musical e o posicionamento da câmera foi horizontal. Esta gravação foi feita em câmera de celular, com filmagem trêmula, e os objetos a serem detectados estão presentes em quase todos os quadros e, ocasionalmente, aparecendo com oclusão parcial. A resolução do vídeo é de 1280*x*720 e o tempo de duração é de 201 segundos.
- Vídeo 06: O objeto de detecção é a guitarra. Este vídeo foi gravado em um espetáculo musical, com condições similares ao Vídeo 05, onde o objeto a ser detectado permanece na maior parte do tempo enquadrado e, ocasionalmente, com oclusão parcial ou total. A resolução do vídeo é de 1920*x*1080 e o tempo de duração é de 375 segundos.

A Tabela 4 sintetiza as informações dos vídeos de aplicação em instrumentos musicais, em relação ao objeto de detecção, à resolução por quadro, ao tamanho original por *pixel* considerando todos os quadros do vídeo, ao tempo total do vídeo em segundos e à quantidade total de quadros avaliados na detecção, sendo eles, apenas 1/12 do total de quadros do vídeo.

| Entrada | Objetos de Detecção | Descrição do Vídeo | Tempo de duração | Resolução de Entrada | Tamanho do Vídeo Original | Total de frames avaliados |
|---|---|---|---|---|---|---|
| Vídeo 01 | Violão e Microfone | Vídeo gravado na vertical em celular com gravação trêmula e o objeto a ser detectado sempre em quadro | 229 | 292320 | 12078720000 | 485 |
| Vídeo 02 | Violão e Microfone | Vídeo gravado na horizontal com câmeras em movimento. Em vários momentos, o objeto a ser detectado não está em quadro | 267 | 2073600 | 16547328000 | 665 |
| Vídeo 03 | Violão e Microfone | Vídeo gravado na horizontal com câmeras em movimento. Em vários momentos, o objeto a ser detectado não está em quadro | 524 | 777600 | 12072960000 | 1294 |
| Vídeo 04 | Violão, Guitarra* e Microfone | Vídeo gravado na horizontal em celular com gravação trêmula e o objeto a ser detectado sempre está em quadro | 175 | 2073600 | 10886400000 | 438 |
| Vídeo 05 | Violão e Microfone | Vídeo gravado na horizontal em celular com gravação trêmula e o objeto a ser detectado está em quadro em quase todos os quadros e, quando em quadro, ocasionalmente aparece com oclusão parcial | 201 | 921600 | 5557248000 | 503 |
| Vídeo 06 | Guitarra | Vídeo gravado na horizontal em celular com gravação trêmula e o objeto a ser detectado sempre está em quadro, em poucos momentos com oclusão parcial ou total | 375 | 2073600 | 46656000000 | 1875 |

*A Guitarra entrou nessa detecção pois o modelo não reconheceu bem o instrumento com a classe correta

**Tabela 4 -** Informações dos vídeos de entrada utilizados para os testes na aplicação em instrumentos musicais, expondo o objeto de detecção, breve descrição, tempo de duração em segundos, resolução de entrada por quadro, tamanho do vídeo original por *pixel* no vídeo inteiro e total de quadros avaliados na detecção dos vídeos.

**Fonte**: Do Autor.

## 4.7 Avaliação dos vídeos

Para avaliar os vídeos de saída, foram considerados o tempo de execução do *pipeline*, levando em conta cada etapa, a confiança média do modelo para o objeto de detecção e o percentual de informação excluída do vídeo original para o vídeo de saída. Descartando em cada quadro o excedente que estava externo às caixas delimitadoras nas detecções, esse percentual foi calculado através da proporção do tamanho em *pixels* do vídeo original, em relação à somatória do tamanho em *pixels* dos quadros que compõem o vídeo de saída.

# 5 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados e discutidos os resultados obtidos a partir da metodologia descrita no capítulo anterior. Primeiramente, são apresentados os resultados da validação do *pipeline* do programa. Logo após, são discutidos os resultados obtidos na etapa de treinamento da rede, como acurácia, precisão, revocação e perda média e IoU. Em seguida, são apresentados os resultados referentes à etapa de detecção em vídeo e o *zoom in* na aplicação de instrumentos musicais.

## 5.1 Validação do *Pipeline*

Considerando o desempenho geral do software, em decorrência da variação dos cenários, foi possível avaliar dois aspectos distintos nos vídeos de saída[3]. O primeiro, foi o desempenho do modelo em relação à quantidade de objetos encontrados em quadros, à confiança média do modelo para a classe detectada e o percentual de informação excedente. O segundo aspecto foi em relação ao tempo de execução para cada etapa do sistema.

Na Tabela 5 é possível perceber que o Cenário 4, que corresponde ao cenário sem parâmetro de qualidade, majoritariamente, tem menos objetos encontrados na etapa de detecção, com exceção do Vídeo 06, detectando a classe gato, que apresenta uma quantidade similar aos demais cenários.

[3] Link para os vídeos

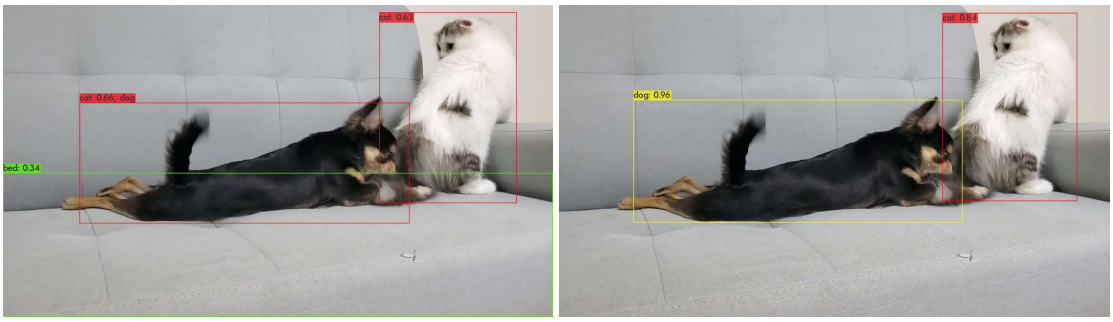
| Entrada | Objeto de detecção | Cenários | Quatidade de objetos encontrados | Tamanho do vídeo cortado | Confiança média | Percentual de informação excedente |
|---|---|---|---|---|---|---|
| Vídeo 01 | Cachorro | 1 | 550 | 595260004 | 78,68% | 71,29% |
| | | 2 | 551 | 599952983 | 77,67% | 71,07% |
| | | 3 | 523 | 610678781 | 74,84% | 70,55% |
| | | 4 | 20 | 1925991165 | 51,42% | 7,12% |
| Vídeo 02 | Cachorro | 1 | 138 | 2385063858 | 81,34% | 45,05% |
| | | 2 | 138 | 2384261489 | 80,44% | 45,07% |
| | | 3 | 138 | 2373363530 | 79,96% | 45,32% |
| | | 4 | 47 | 3327216053 | 69,41% | 23,35% |
| Vídeo 03 | Cachorro | 1 | 335 | 2874485209 | 78,14% | 50,96% |
| | | 2 | 334 | 2832407600 | 78,47% | 51,68% |
| | | 3 | 314 | 3027314290 | 78,92% | 48,35% |
| | | 4 | 38 | 5018951836 | 45,12% | 14,37% |
| Vídeo 04 | Cachorro | 1 | 60 | 470572591 | 75,75% | 65,41% |
| | | 2 | 61 | 465972300 | 75,83% | 65,74% |
| | | 3 | 54 | 470968871 | 77,12% | 65,38% |
| | | 4 | 6 | 1097382465 | 53,06% | 19,33% |
| Vídeo 05 | Cachorro | 1 | 507 | 1151537989 | 67,79% | 90,47% |
| | | 2 | 494 | 1218124792 | 68,21% | 89,92% |
| | | 3 | 503 | 1162444930 | 67,63% | 90,38% |
| | | 4 | 237 | 3910565523 | 66,10% | 67,62% |
| Vídeo 06 | Cachorro | 1 | 226 | 1846332959 | 75,82% | 54,34% |
| | | 2 | 219 | 1812906692 | 75,96% | 55,17% |
| | | 3 | 206 | 1890000118 | 77,37% | 53,26% |
| | | 4 | 154 | 1817686855 | 68,26% | 55,05% |
| | Gato | 1 | 125 | 1755701280 | 75,82% | 56,58% |
| | | 2 | 127 | 1743079168 | 75,96% | 56,89% |
| | | 3 | 132 | 1756336559 | 77,37% | 56,56% |
| | | 4 | 126 | 2051702766 | 68,26% | 49,26% |
| Vídeo 07 | Cachorro | 1 | 522 | 11350742987 | 76,77% | 12,70% |
| | | 2 | 521 | 11328465056 | 77,17% | 12,87% |
| | | 3 | 522 | 11359832942 | 76,91% | 12,63% |
| | | 4 | 436 | 10457500484 | 60,56% | 19,57% |
| | Gato | 1 | 379 | 4997903384 | 76,77% | 61,56% |
| | | 2 | 378 | 4979280177 | 77,17% | 61,70% |
| | | 3 | 392 | 5017269598 | 76,91% | 61,41% |
| | | 4 | 363 | 6055018132 | 60,56% | 53,43% |

**Tabela 5 -** Desempenho do *pipeline* para os vídeos de entrada em relação a cada cenário, apresentando a quantidade de objetos encontrados, o tamanho do vídeo cortado, a confiança média em todos os quadros do modelo para o objeto detectado e o percentual de informação excedente removido no vídeo final.

**Fonte**: Do Autor.

Para esse caso, apesar de ter encontrado mais objetos, o Cenário 4 obteve a menor confiança em relação aos outros 3. Isso se dá porque a imagem, por ter qualidade inferior e *pixels* bem aparentes (granulada), prejudica a detecção do objeto e compromete a precisão do modelo, fazendo com que a confiança caia e podendo até gerar detecções erradas, como é possível perceber na Figura 8, onde temos o 252º quadro extraído do Vídeo 06, e na Figura 9, onde temos o 1620º quadro extraído do Vídeo 01. Em ambas as figuras, a imagem (a) corresponde ao Cenário 4 e a imagem (b) ao Cenário 1.

Na Figura 8, no Cenário 4, é possível perceber que, além do modelo não conseguir identificar as classes corretamente, tendo duas classes na mesma caixa delimitadora, ele ainda manteve a confiança baixa, em relação à detecção do mesmo quadro com o Cenário 1.




(a) (b)

**Figura 8 -** Detecção do 252º quadro do vídeo 06, onde as detecções estão marcadas por caixas delimitadoras nas cores verde, vermelho e amarelo. Na figura (a) há três classes detectadas, cama em verde com precisão de 0.34, gato e cachorro em vermelho com precisão de 0.66 e gato em vermelho com precisão de 0.63. Na figura (b) há duas classes detectadas, cachorro em amarelo com precisão de 0.96 e gato em vermelho com precisão de 0.84.

**Fonte**: Quadro extraído do vídeo 06.

A Figura 9 representa dois quadros extraídos do Vídeo 01, em cenários diferentes, os Cenários 4, à esquerda, e 1, à direita. No Cenário 4, é possível perceber que o modelo não conseguiu identificar corretamente a classe esperada e manteve sua confiança baixa. Em contrapartida, no Cenário 1, em que a imagem está suavizada, com uma redução considerável de *pixels* aparentes, temos as duas classes detectadas corretamente, Pessoa e Cachorro, respectivamente.


(a) (b)

**Figura 9 -** Detecção do 1620º quadro do vídeo 01, onde as detecções estão marcadas por caixas delimitadoras marcadas nas cores verde, roxo e amarelo. Na figura (a) há uma classe detectada, barco em verde com precisão 0.36. Na figura (b) há duas classes detectadas, pessoa em roxo com precisão 0.52 e cachorro em amarelo com precisão 0.83.

**Fonte**: Quadro extraído do vídeo 01.

Ainda considerando a confiança, em todos os vídeos, os Cenários 1, 2 e 3 obtiveram maior confiança média na detecção, comparados ao Cenário 4, mas, entre si, não foi possível perceber um comportamento padrão.

Na maioria dos casos, o Cenário 1 foi o que obteve o maior percentual de informação excedente removido no vídeo de saída. Considerando que ele detém os quadros com melhor qualidade de imagem, esse percentual elevado pode estar associado ao fato de que, quanto maior a precisão na detecção, mais precisa será a caixa delimitadora e o corte final será menor, descartando mais informações desnecessárias no quadro.

A Tabela 6 exibe o tempo de execução para cada vídeo em cada cenário. Avaliando os cenários em decorrência do tempo de execução, é possível perceber que o Cenário 4, na maioria dos casos, teve um tempo de extração maior em relação aos demais cenários, com exceção do Vídeo 07, detectando a classe cachorro, em que o Cenário 1 demorou 9.37 segundos a mais para finalizar.

Desta forma, é possível afirmar que o Cenário 4 não obteve resultados satisfatórios. Além de ser o que realiza a extração com pior qualidade de imagem, é o que demanda mais tempo de execução para a extração dos quadros, na maioria dos testes.

| Entrada | Objeto de Detecção | Cenários | Tempo de execução total | Tempo de extração das imagens | Tempo de detecção | Tempo de recorte das imagens | Tempo de criação do vídeo |
|---|---|---|---|---|---|---|---|
| Vídeo 01 | Cachorro | 1 | 15min28.97s | 2,05% | 39,52% | 5,36% | 53,07% |
| | | 2 | 15min22.46s | 1,62% | 39,21% | 5,16% | 54,01% |
| | | 3 | 15min46.66s | 1,77% | 39,13% | 4,49% | 54,61% |
| | | 4 | 13min24.78s | 2,41% | 43,83% | 6,62% | 47,14% |
| Vídeo 02 | Cachorro | 1 | 09min01.73s | 5,88% | 38,78% | 19,61% | 35,73% |
| | | 2 | 08min50.37s | 5,12% | 39,64% | 18,42% | 36,82% |
| | | 3 | 08min43.63s | 4,71% | 39,17% | 18,23% | 37,89% |
| | | 4 | 08min10.46s | 6,68% | 40,90% | 19,94% | 32,48% |
| Vídeo 03 | Cachorro | 1 | 13min15.91s | 5,10% | 34,22% | 16,40% | 44,28% |
| | | 2 | 13min03.28s | 4,69% | 34,58% | 14,97% | 45,76% |
| | | 3 | 13min25.20s | 4,58% | 33,66% | 15,50% | 46,26% |
| | | 4 | 13min21.09s | 5,49% | 33,19% | 15,78% | 45,54% |
| Vídeo 04 | Cachorro | 1 | 03min14.95s | 6,88% | 34,90% | 14,49% | 43,73% |
| | | 2 | 03min06.46s | 5,72% | 35,17% | 13,91% | 45,20% |
| | | 3 | 03min06.22s | 4,95% | 35,37% | 14,85% | 44,83% |
| | | 4 | 02min33.98s | 7,24% | 40,90% | 19,08% | 32,78% |
| Vídeo 05 | Cachorro | 1 | 15min18.78s | 8,69% | 30,40% | 22,00% | 38,91% |
| | | 2 | 14min49.81s | 8,07% | 31,17% | 20,33% | 40,43% |
| | | 3 | 14min43.58s | 7,31% | 31,13% | 20,15% | 41,41% |
| | | 4 | 14min33.30s | 9,88% | 30,06% | 21,62% | 38,44% |
| Vídeo 06 | Cachorro | 1 | 05min47.26s | 8,14% | 27,55% | 23,79% | 40,52% |
| | | 2 | 05min39.26s | 7,58% | 27,93% | 22,79% | 41,70% |
| | | 3 | 05min37.34s | 6,86% | 27,26% | 23,22% | 42,66% |
| | | 4 | 05min37.70s | 8,82% | 26,55% | 19,97% | 44,66% |
| | Gato | 1 | 05min47.34s | 8,06% | 27,72% | 23,04% | 41,18% |
| | | 2 | 05min40.80s | 7,43% | 28,08% | 22,18% | 42,31% |
| | | 3 | 05min41.73s | 7,13% | 27,79% | 22,43% | 42,65% |
| | | 4 | 05min40.35s | 8,77% | 26,46% | 20,79% | 43,98% |
| Vídeo 07 | Cachorro | 1 | 20min35.49s | 9,11% | 24,91% | 33,95% | 32,03% |
| | | 2 | 19min44.41s | 8,34% | 25,96% | 32,58% | 33,12% |
| | | 3 | 19min31.79s | 7,51% | 25,66% | 32,35% | 34,48% |
| | | 4 | 19min50.85s | 8,67% | 24,14% | 25,45% | 41,74% |
| | Gato | 1 | 19min19.58s | 9,99% | 26,99% | 26,30% | 36,72% |
| | | 2 | 18min35.13s | 8,85% | 27,59% | 25,34% | 38,22% |
| | | 3 | 18min36.74s | 7,68% | 26,90% | 24,28% | 41,14% |
| | | 4 | 19min09.21s | 9,22% | 25,12% | 20,54% | 45,12% |

**Tabela 6 -** Desempenho do *pipeline* para os vídeos de entrada em relação a cada cenário, apresentando o tempo de execução total e os percentuais de cada etapa.

Dentre os três primeiros cenários, o comportamento foi similar quando comparados ao número de objetos detectados, o tempo de execução para a extração dos quadros e o percentual de informação excedente, mas, ao avaliarmos a confiança média do modelo em relação à classe detectada, o Cenário 1 obteve os melhores resultados. Em razão disto, o Cenário 1 foi o escolhido para os testes realizados na detecção aplicada a instrumentos musicais.

Ainda na Tabela 6, é possível perceber que o maior tempo de processamento se concentra nas etapas de detecção, recorte dos objetos e criação do vídeo de saída.

## 5.2 Aplicação em instrumentos musicais

### 5.2.1 Treinamento da rede

A rede foi treinada com 3 classes, como apresentado na Subseção 4.5. Na Tabela 7, serão apresentadas as métricas alcançadas para cada classe, ao fim do treinamento.

| Classes | Precisão | Precisão Média | Verdadeiros Positivos | Falsos Positivos |
|---|---|---|---|---|
| Guitarra | 67,50% | 61,90% | 27 | 13 |
| Violão | 87,70% | 87,94% | 107 | 15 |
| Microfone | 86,23% | 62,27% | 94 | 15 |

**Tabela 7 -** Desempenho do modelo avaliando diferentes métricas e frequências de classificação.

**Fonte**: Do Autor.

Através desses valores, é possível perceber que a precisão média na detecção das classes Guitarra e Microfone é mais baixa que na classe Violão. Para ambas as classes, pode-se atribuir esse resultado à quantidade de imagens na base de dados que as compõem, tendo apenas 121 e 478 imagens para as classes guitarra e microfone, respectivamente.

Em adição a esse problema, existe ainda inconsistência na rotulação dessas imagens pois, a base utilizada contém imagens previamente rotuladas e, em alguns casos, essa rotulação foi feita de forma errada. Essa falha dificulta o treinamento da rede, portanto, se faz necessário verificar a rotulação existente em busca de possíveis falhas.

Como é possível perceber na Figura 10, a imagem à esquerda tem a rotulação correta, enquanto que a imagem à direita foi rotulada de forma errada.

**Figura 10 -** Imagens do banco de dados pertencentes à classe Guitarra.

**Fonte**: Acervo da Imagenet [43].

Ademais, para a classe Microfone, é possível atribuir esse valor à qualidade das imagens, pois elas são de difícil identificação e baixa resolução, além de muitas não serem representativas. A Figura 11 exemplifica algumas dessas situações.

**Figura 11 -** Imagens do banco de dados pertencentes à classe Microfone.
Fonte: Acervo da Imagenet [43].

Para a classe Guitarra, foi encontrado ainda um problema de semelhança com as imagens da classe Violão. As diferenças entre guitarras e violões são consideravelmente pequenas, sendo necessário aumentar o acervo para as duas classes, para melhorar o desempenho na detecção de ambas.

Para a classe Violão, considerando o acervo utilizado, o resultado pode ser considerado satisfatório para a etapa de detecção.

### 5.2.2 Detecção em vídeo e *zoom In*

Considerando o desempenho geral do software, utilizando apenas os parâmetros de extração do Cenário 1, em todos os vídeos de saída[4], foram avaliados os mesmos aspectos evidenciados na Seção 5.1.

Na Tabela 8, é possível perceber que o percentual de informação excedente variou entre 24,91% e 98,22%, correspondentes aos Vídeos 03 e 01, respectivamente.

A causa desse baixo percentual, na detecção da classe violão no Vídeo 01, pode ser justificada pela característica do vídeo, que, em vários momentos, o violão não se encontra em quadro, além dos problemas de detecção, pois o modelo nem sempre consegue detectar o objeto.

Avaliando a detecção da classe violão, é possível perceber que no Vídeo 01, como o violão se manteve enquadrado ao longo de todo o vídeo, o modelo conseguiu detectá-lo

[4] Link para os vídeos

em quase 70% dos quadros avaliados. Em contrapartida, o Vídeo 01 obteve a segunda menor confiança média em suas detecções, resultante da baixa resolução do vídeo de entrada.

| Entrada | Objeto de Detecção | Quatidade de objetos encontrados | Tamanho do vídeo cortado | Confiança média | Percentual de informação excedente |
|---|---|---|---|---|---|
| Vídeo 01 | Violão | 330 | 3766170379 | 71,21% | 68,82% |
| | Microfone | 461 | 215378880 | 97,86% | 98,22% |
| Vídeo 02 | Violão | 253 | 11886274136 | 85,42% | 28,17% |
| | Microfone | 562 | 2369106891 | 85,62% | 85,68% |
| Vídeo 03 | Violão | 373 | 9065854256 | 87,78% | 24,91% |
| | Microfone | 909 | 1690808585 | 86,54% | 86,00% |
| Vídeo 04 | Guitarra | 413 | 4853459695 | 90,41% | 55,42% |
| | Microfone | 386 | 226670177 | 77,44% | 97,92% |
| | Violão | 83 | 8338972862 | 58,62% | 23,40% |
| Vídeo 05 | Violão | 303 | 3358820814 | 82,23% | 39,56% |
| | Microfone | 107 | 2353690863 | 59,29% | 57,65% |
| Vídeo 06 | Guitarra | 1707 | 26870270505 | 88,82% | 42,41% |

**Tabela 8 -** Desempenho do *pipeline* para os vídeos de entrada, apresentando a quantidade de objetos encontrados, o tamanho do vídeo cortado, a confiança média em todos os quadros do modelo para o objeto detectado e o percentual de informação excedente removido no vídeo final.

**Fonte**: Do Autor.

Através da Tabela 8, também é possível identificar a relação direta entre a resolução do vídeo de entrada e a confiança média do modelo na detecção, onde, quanto maior a resolução, maior a confiança média.

Ainda na Tabela 8, é possível perceber que a classe microfone obteve bons resultados a respeito ao percentual de informação excedente, que está relacionado ao tamanho do objeto em relação ao quadro do vídeo. Ademais, alcançou bons resultados na confiança média, com uma média de 81%, na detecção.

Apesar dos resultados apresentados na Tabela 8, os vídeos de saída não ficaram visualmente satisfatórios, pois não ficaram fluidos e, por ser um objeto pequeno em relação ao quadro, pois, quando foi feito o corte, a imagem perdeu bastante resolução.

No Vídeo 04, apesar do instrumento correto presente no vídeo ser o violão, também foi testada a detecção da classe guitarra, pois, o modelo não detectou corretamente o violão, apesar do objeto estar presente em muitos quadros e ter boa visibilidade. Esse

resultado pode ser atribuído à falha na rotulação das imagens na base de dados, como mencionado na subseção 5.2.1.

A confiança média está associada ao desempenho do modelo na detecção. Quando a confiança alcança valores mais altos, pode indicar melhores caixas delimitadoras. Desta forma, com caixas delimitadoras bem definidas, o objeto recortado tem melhor qualidade de conteúdo.

A Tabela 9 mostra o tempo de execução para cada vídeo de entrada, todos seguindo os parâmetros de extração do Cenário 1. Através desta tabela, é possível concluir que, apesar do Cenário 1 ter apresentado os maiores percentuais de tempo para a extração dos quadros, o tempo para a extração representa o menor percentual, em relação às outras etapas, levando em média 7% do tempo de execução total do *pipeline*.

| Entrada | Objeto de Detecção | Tempo de execução total | Tempo de extração das imagens | Tempo de detecção | Tempo de recorte das imagens | Tempo de criação do vídeo |
|---|---|---|---|---|---|---|
| Vídeo 01 | Violão | 15min28.97s | 7,89% | 21,60% | 21,78% | 48,73% |
| | Microfone | 15min46.89s | 8,25% | 19,20% | 15,92% | 56,63% |
| Vídeo 02 | Violão | 19min17.76s | 8,28% | 22,91% | 30,98% | 37,83% |
| | Microfone | 24min51.87s | 9,84% | 25,35% | 24,07% | 40,74% |
| Vídeo 03 | Violão | 20min28.79s | 4,34% | 28,76% | 19,69% | 47,21% |
| | Microfone | 26min15.33s | 7,08% | 20,51% | 15,58% | 56,83% |
| Vídeo 04 | Guitarra | 11min05.47s | 7,26% | 24,69% | 27,14% | 40,91% |
| | Microfone | 13min45.13s | 9,31% | 20,62% | 21,22% | 48,85% |
| | Violão | 11min17.23s | 6,76% | 24,27% | 31,56% | 37,41% |
| Vídeo 05 | Violão | 15min18.78s | 3,62% | 18,94% | 19,36% | 58,08% |
| | Microfone | 12min48.81s | 4,81% | 20,26% | 13,23% | 61,71% |
| Vídeo 06 | Guitarra | 05min47.26s | 6,59% | 30,44% | 28,63% | 34,34% |

**Tabela 9 -** Desempenho do *pipeline* para os vídeos de entrada, apresentando o tempo de execução total e os percentuais de cada etapa.

Na Figura 12, extraída do Vídeo 03 de saída, ainda é possível perceber que a oclusão, mesmo sendo parcial, dificultou a detecção do objeto pelo modelo, mesmo ele estando enquadrado.

**Figura 12 -** Imagem extraída do Vídeo 03 após o processamento do *pipeline*, onde o objeto encontra-se em quadro, mas não foi detectado pelo modelo.

**Fonte**: Do Autor.

Temos o mesmo efeito na Figura 13, extraída do Vídeo 04 em condições similares, mas com resolução do vídeo de entrada melhor, com oclusão parcial do violão. Nesse vídeo, o modelo faz algumas detecções, como é possível perceber em poucas interpolações feitas, porém, em diversos momentos, o vídeo permanece estático, sem recortes. Podemos associar essa falha à capacidade de detecção do modelo.

**Figura 13 -** Imagens extraídas do Vídeo 04 após o processamento do *pipeline*. À esquerda o objeto encontra-se em quadro, mas não foi detectado pelo modelo. À direita o objeto encontra-se em quadro e foi feito o recorte.

**Fonte**: Do Autor.

# 6 CONSIDERAÇÕES FINAIS

Este trabalho apresenta uma proposta de desenvolvimento de um sistema de recorte em vídeo inteligente que, utilizando um modelo de detecção baseado na YOLOv4, pode fazer a detecção de um objeto, e, de forma automatizada, realizar seu destaque.

No sistema desenvolvido, foi implementado um *pipeline*, que consiste nas etapas de fragmentação do vídeo, detecção dos objetos utilizando um modelo treinado, recorte de um objeto escolhido e criação do vídeo final a partir desses recortes.

Para a validação deste *pipeline*, partiu-se do pressuposto que a detecção funciona de forma eficiente, para que fosse possível avaliar o comportamento do sistema nas demais etapas. Dessa forma, foi utilizado um modelo pré-treinado da YOLOv4, detectando classes arbitrárias, sendo elas gato e cachorro.

Foram testadas diferentes combinações de parâmetros de qualidade dos quadros na extração, avaliando a confiança do modelo para cada cenário. Além desse aspecto, também foi avaliado o comportamento do sistema na estabilização dos *pixels* na etapa de recorte. O desenvolvimento prezou majoritariamente pela resolução e fluidez do vídeo de saída, tempo de execução e suavização dos quadros do vídeo.

Para a aplicação em instrumentos musicais, foi criada uma base de dados de objetos pertencentes a três classes especificamente. O foco do trabalho não é a avaliação da detecção dos objetos em questão, mas é parte do processo. Dessa forma, o modelo foi treinado a partir dessa base, obtendo uma precisão média de teste de 61,90%, 87,94% e 62,27% para guitarra, violão e microfone, respectivamente. Essa métrica avalia o desempenho do modelo com o conjunto de teste.

Nos testes aplicados no *pipeline* com instrumentos musicais, o modelo apresentou uma confiança média acima de 71%, 88% e 59% para as classes violão, guitarra e microfone. A confiança média foi a métrica utilizada para avaliar o desempenho do modelo nos diferentes cenários, onde foi possível perceber que imagens com qualidade superior obtêm maior confiança. Foi possível perceber portanto que, quanto melhor o parâmetro de extração, maior é a quantidade de objetos detectados pelo modelo, como, também, a detecção tem maior precisão e qualidade. Pode-se observar uma pequena perda de qualidade em relação à resolução do vídeo original, mas não aconteceu perda significativa de conteúdo do vídeo devido ao intervalo dos quadros na detecção.

Ao analisar os resultados obtidos, é possível afirmar que a proposta do trabalho foi alcançada, pois, todos os objetivos apresentados foram atingidos. Sendo eles a criação de um banco de dados aplicado ao contexto de instrumentos musicais, o desenvolvimento de um sistema capaz de realizar a aproximação de objetos em vídeo de forma automatizada e a avaliação de combinações de parâmetros que impactam na resolução de extração de quadros para avaliar a capacidade de detecção do modelo.

Considerando os resultados obtidos na etapa de treinamento da rede, a principal limitação se deu pela quantidade e qualidade das imagens de treinamento. Para o desenvolvimento deste trabalho foi treinado um modelo utilizando imagens estáticas para aplicar a detecção em vídeos, o que pode atrapalhar o desempenho do modelo na detecção. Para trabalhos futuros, pretende-se fazer o treinamento de um modelo utilizando vídeos.

O modelo utilizado neste trabalho é a versão 4 da YOLO. Para trabalhos futuros, pode-se avaliar o desempenho de versões mais recentes, bem como, o desempenho de outros modelos de detecção de objetos e estudar a substituição do modelo utilizado para detecção. Ademais, pretende-se estudar e aprofundar os impactos das métricas que correspondem ao desempenho do modelo.

Na etapa de corte, como mencionada na metodologia, é selecionada sempre a detecção mais à esquerda da imagem. Para trabalhos futuros, pode-se pensar em uma CNN que seja capaz de detectar e reconhecer os objetos, diferenciando-os para que o corte seja feito sempre no mesmo objeto.

A avaliação de desempenho do sistema foi estritamente visual e não foi considerada nenhuma métrica para avaliar o vídeo gerado no final. Por isso, para a próxima etapa, deverão ser avaliadas métricas de desempenho, tais como Fusão de Avaliação de Vídeo Multimétodo (do inglês, *Video Multimethod Assessment Fusion* - VMAF) e Similaridade estrutural (do inglês, *Structural similarity* - SSIM), métricas que comparam a qualidade entre dois vídeos, utilizando como valor de referência o vídeo original. Por fim, além das métricas mencionadas acima, pretende-se também fazer uma avaliação da fluidez do vídeo baseada no fator humano, utilizando o *feedback* do usuário.

Para trabalhos futuros, deve-se ainda considerar o processamento paralelo de alguns métodos, como, por exemplo, a detecção e recorte, para otimizar o tempo de execução.

**REFERÊNCIAS**


[1] AMIT, Yali; FELZENSZWALB, Pedro. Object Detection. **Springer**, p. 537-542, 2014.

[2] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep Learning. **Nature**, vol. 521, p. 436–44, 2015.

[3] SCHMIDHUBER, Jürgen. Deep Learning in Neural Networks: An Overview. **Neural Networks**, vol. 61, p. 85-117, 2015.

[4] ARAÚJO, Ricardo Matsumura de. Aprendizagem de máquina em sistemas complexos multiagentes: estudo de caso em um ambiente sob racionalidade limitada. Dissertação (mestrado em computação). Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

[5] REDMON, Joseph. et al. You only look once: Unified, real-time object detection. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 779-788, 2016.

[6] BOCHKOVSKIY, Alexey. Yolov4: Optimal speed and accuracy of object detection. **Computer Vision and Pattern Recognition**. arXiv:2004.10934, 2020.

[7] BISHOP, Christopher. Pattern Recognition and Machine Learning. 2.ed. Nova Iorque. Spring. 2006

[8] GILDA, Shlok. Notice of Violation of IEEE Publication Principles: Evaluating machine learning algorithms for fake news detection. 2**017 IEEE 15th Student Conference on Research and Development (SCOReD)**, p. 110-115, 2017.

[9] PENG, Tianrui; HARRIS, Ian; SAWA, Yuki. Detecting Phishing Attacks Using Natural Language Processing and Machine Learning. **2018 IEEE 12th International Conference on Semantic Computing (ICSC)**, p. 300-301, 2018.

[10] S. LARABI Marie-Sainte. et al. Arabic Natural Language Processing and Machine Learning-Based Systems. **IEEE Access**, vol. 7, p. 7011-7020, 2019.

[11] SHARMIN, Sadia; ZAMAN, Zakia. Spam Detection in Social Media Employing Machine Learning Tool for Text Mining. **2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)**, p. 137-142, 2017.

[12] TANDEL, Nishtha H.; PRAJAPATI, Harshadkumar B.; DABHI, Vipul K.. Voice Recognition and Voice Comparison using Machine Learning Techniques: A Survey. **2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)**, p. 459-465, 2020.

[13] GANAPATHIRAJU, Aravind; HAMAKER, Jonathan E.; PICONE, Joseph. Applications of support vector machines to speech recognition. **IEEE Transactions on Signal Processing**, vol. 52, no. 8, p. 2348-2355, 2004.

[14] YING, Yong-Qian; WOO, Peng-Yung. Speech recognition using fuzzy logic. **IJCNN'99. International Joint Conference on Neural Networks**, vol. 5, p. 2962-2964, 1999.

[15] ADANKON, Mathias M.; CHERIET, Mohamed. Model selection for the LS-SVM: Application to handwriting recognition. **Pattern Recognition**, vol. 42, no. 12, p. 3264-3270, 2009.

[16] DENG, Li. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. **IEEE Signal Processing Magazine**, vol. 29, no. 6, p. 141-142, 2012.

[17] SENG, Kah Phooi. et al. Computer Vision and Machine Learning for Viticulture Technology. **IEEE Access**, vol. 6, p. 67494-67510, 2018.

[18] KHAN, Asharul Islam; AL-HABSI, Salim. Machine Learning in Computer Vision. **International Conference on Computational Intelligence and Data Science**, vol 167, p. 1444-1451, 2020.

[19] FELZENZWALB, Pedro F. et al. Object detection with discriminatively trained part-based models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 32, no. 9, p. 1627, 2010.

[20] SUNG, K.-K.; POGGIO, T. Example-based learning for view-based human face detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 20, no. 1, p. 39–51, 1998.

[21] DOLLAR, Piotr. et al. Pedestrian detection: An evaluation of the state of the art. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 34, no. 4, p. 743, 2012.

[22] PEPPA, Maria Valasia. et al. URBAN TRAFFIC FLOW ANALYSIS BASED ON DEEP LEARNING CAR DETECTION FROM CCTV IMAGE SERIES. **ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, vol. XLII-4, p. 499-506, 2018.

[23] DU, Xinxin; ANG, Marcelo H.; RUS, Daniela. Car detection for autonomous vehicles: LIDAR and vision fusion approach through deep learning framework. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, p. 749-754, 2017.

[24] OU, Zhonghong. et al. A CNN-Based Supermarket Auto-Counting System. **Algorithms and Architectures for Parallel Processing**, p. 359-371, 2017.

[25] SANTANA, Clodomir Joaquim de. et al. A Solution for Counting Aedes aegypti and Aedes albopictus Eggs in Paddles from Ovitraps Using Deep Learning. **IEEE Latin America Transactions**, vol. 17, no. 12, p. 1987-1994, 2019.

[26] JIA, Yangqing. et al. Caffe: Convolutional architecture for fast feature embedding. **Computer Vision and Pattern Recognition**, arXiv:1408.5093, 2014.

[27] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E.. Imagenet classification with deep convolutional neural networks. **Association for Computing Machinery**, vol. 60, no. 6, p. 84-90, 2012.

[28] CAO, Zhe. et al. Realtime multi-person 2d pose estimation using part affinity fields. **Computer Vision and Pattern Recognition**, arXiv:1611.080502017, 2017.

[29] YANG, Zhenheng; NEVATIA, Ramakant. A multi-scale cascade fully convolutional network face detector. **23rd International Conference on Pattern Recognition (ICPR)**, P. 633-638, 2016.

[30] GIRSHICK, Ross. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. **2014 IEEE Conference on Computer Vision and Pattern Recognition**, p. 580-587, 2014.

[31] GIRSHICK, Ross. Fast r-cnn. **2015 IEEE International Conference on Computer Vision (ICCV)**, p. 1440-1448, 2015.

[32] REN, Shaoqing. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **Advances in Neural Information Processing Systems 28 (NIPS 2015)**, p. 91–99, arXiv:1506.01497, 2015.

[33] COMASCHI, Francesco. et al. RASW: A run-time adaptive sliding window to improve Viola-Jones object detection. **2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)**, 2013.

[34] DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**, vol. 1, p. 886-893, 2005.

[35] CHEN, Dong. et al. Joint cascade face detection and alignment. **Springer International Publishing (ECCV)**, vol. 8694, 2014.

[36] DCHEN, Dong. et al. Supervised transformer network for efficient face detection. **Springer International Publishing (ECCV)**, vol. 9909, p. 122-138, 2016.

[37] RIBEIRO, David. et al. A real-time pedestrian detector using deep learning for human-aware navigation. **2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)**, arXiv:1607.04441, p. 165-171, 2016.

[38] YANG, Fan; CHOI, Wongun; LIN, Yuanqing. Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 2129-2137, 2016.

[39] LONG, Jonathan; SHELHAMER, Evan; DARREL, Trevor. Fully convolutional networks for semantic segmentation. **Computer Vision and Pattern Recognition**, arXiv:1411.4038, 2015.

[40] JUNG, Chanho; KIM, Changick. A Unified Spectral-Domain Approach for Saliency Detection and Its Application to Automatic Object Segmentation. **IEEE Transactions on Image Processing**, vol. 21, no. 3, p. 1272–1283, 2012.

[41] GIDARIS, Spyros; KOMODAKIS, Nikos. Object detection via a multi-region and semantic segmentation-aware cnn model. **2015 IEEE International Conference on Computer Vision (ICCV)**, p. 1134-1142, 2015.

[42] ZHU, Yukun. et al. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 4703-471, 2015.

[43] RUSSAKOVSKY, Olga; DENG, Jia. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision**, arXiv:1409.0575, 2014, p. 211–252, 2015.

[44] LI, Chen. et al. CNN based post-processing to improve HEVC. **2017 IEEE International Conference on Image Processing (ICIP)**, p. 4577-4580, 2017.

[45] MA, Di; ZHANG, Fan; BULL David R.. MFRNet: A New CNN Architecture for Post-Processing and In-loop Filtering. **IEEE Journal of Selected Topics in Signal Processing**, vol 2, n. 15, p. 378-387, 2021.

[46] SANTAMARIA, Maria. et al. Content-adaptive convolutional neural network post-processing filter. **2021 IEEE International Symposium on Multimedia (ISM)**, p. 99-106, 2021.

[47] WEN, Liwu; DING, Jinshan; XU, Zhong. Multiframe Detection of Sea-Surface Small Target Using Deep Convolutional Neural Network. **IEEE Transactions on Geoscience and Remote Sensing**, vol. 60, p. 1-16, 2022.

[48] CHOI, Dong Yoon. et al. CNN-based pre-processing and multi-frame-based view transformation for fisheye camera-based AVM system. **2017 IEEE International Conference on Image Processing (ICIP)**, p. 4073-4077, 2017.

[49] HUANG, Yingping. et al. Learning Optical Flow with R-CNN for Visual Odometry. **2021 IEEE International Conference on Robotics and Automation (ICRA)**, p.14410-14416, 2021.

[50] TIAN, Long. et al. Unsupervised Learning of Optical Flow With CNN-Based Non-Local Filtering. **IEEE Transactions on Image Processing**, vol. 29, p. 8429-8442, 2020.

[51] REDMON, Joseph; FARHADI, Ali. YOLOv3: An incremental improvement. **Computer Vision and Pattern Recognition**, arXiv:1804.02767, 2018.

[52] SANG, J.et al. An improved YOLOv2 for vehicle detection. **Sensors 2018**, vol. 18, n. 12, 2018.

[53] REDMON, Joseph; FARHADI, Ali. YOLO9000: Better, Faster, Stronger. **Computer Vision and Pattern Recognition**, arXiv:1612.08242, 2016.

[54] LIQUAN, Zhao; SHUAIYANG, LI. Object detection algorithm based on improved YOLOv3. Electronics, vol. 9 p. 537, 2020.

[55] LI, Chuyi. et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. **Computer Vision and Pattern Recognition**, arXiv:2209.02976, 2022.

[56] WANG, Chien-Yao; BOCHKOVSKIY. Alexey; LIAO, Hong-Yuan Mark. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. **Computer Vision and Pattern Recognition**, arXiv:2207.02696, 2022.

[57] LONG , Xiang. et al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. **Computer Vision and Pattern Recognition**, arXiv:2007.12099, 2020.

[58] HUANG, Xin. et al. PP-YOLOv2: A Practical Object Detector. **Computer Vision and Pattern Recognition**, arXiv:2104.10419, 2021.

[59] GHIASI, Golnaz; LIN, Tsung-Yi; LE, Quoc V.. Dropblock: A regularization method for convolutional networks. **Advances in Neural Information Processing Systems**, arXiv:1810.12890, 2018.

[60] MULLER, Rafael; KORNBLITH, Simon; HINTON, Geoffrey. When Does Label Smoothing Help?. **33rd Conference on Neural Information Processing Systems (NeurIPS 2019)**, 2019.

[61] YUN, Sangdoo. et al. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. **IEEE/CVF International Conference on Computer Vision (ICCV)**, arXiv:1905.04899, 2019.

[62] ZHONG, Zhun. et al. Random Erasing Data Augmentation. **Computer Vision and Pattern Recognition**, arXiv:1708.04896, 2017.

[63] SRIVASTAVA, Nitish. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, vol. 15, no. 56, p. 1929-1958, 2014.

[64] MISRA, Diganta. Mish: A Self Regularized Non-Monotonic Activation Function. **Computing Research Repository**, arXiv:1908.08681, 2019.

[65] ZEILER, Matthew. et aI. On Rectified Linear Units For Speech Processing. **2013 IEEE International Conference on Acoustics, Speech and Signal Processing**, p. 3517-3521, 2013.

[66] RAMACHANDRAN, Prajit; ZOPH, Barret; LE, Quoc V.. Swish: A selfgated activation function. **Neural and Evolutionary Computing**, arXiv:1710.05941, 2017.

[67] ELFWING, Stefan; UCHIBE, Eiji; DOYA, Kenji. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. **Machine Learning**, vol. 107,p. 3-11, 2018.

[68] HUANG, Gao. et al. Densely Connected Convolutional Networks. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 2261-2269, 2017.

[69] HE, Kaiming. et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 37, no. 9, p. 1904–1916, 2015.

[70] LIU, Shu. et al. Path aggregation network for instance segmentation. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, p. 8759–8768, 2018.

[71] CIRESAN, Dan; MEIER, Ueli; SCHMIDHUBER, Jurgen. Multi-column deep neural networks for image classification. **2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. p. 3642–3649, 2012.

[72] THOMSEN, V. “Precision and The Terminology of Measurement”. **The Physics Teacher**, Vol. 35, pp.15-17, Jan. 1997.

[73] EVERINGHAM, Mark. et al. The PASCAL Visual Object Classes Challenge: A Retrospective. **International Journal of Computer Vision**, vol. 111, no. 1, p. 98-136, 2015.

[74] Google acadêmico. Disponível em: <https://scholar.google.com.br/>. Acesso em: Outubro de 2022.

[75] Scielo. Disponível em: <https://scielo.org//>. Acesso em: Outubro de 2022.

[76] IEEE Xplore. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso em: Outubro de 2022

[77] DEWI, Christine. et al. Similar Music Instrument Detection via Deep Convolution YOLO-Generative Adversarial Network. **2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)**, p. 1-6, 2019.

[78] WEI, Gang. et al. Progressive generative adversarial networks with reliable sample identification. **Pattern Recognition Letters**, vol. 130, p. 91-98, 2019.

[79] YAO, Bangpeng Yao; FEI-FEI, Li. Grouplet: A Structured Image Representation for Recognizing Human and Object Interactions. Computer Science Department. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 9-16, 2010.

[80] SLIZOVSKAIA, Olga; GOMEZ, Emilia; HARO, Gloria. Musical Instrument Recognition in User-generated Videos using a Multimodal Convolutional Neural Network Architecture. **2017 ACM on International Conference on Multimedia Retrieval**, p. 226–232, 2017.

[81] SZEGEDY, Christian. et al. Rethinking the Inception Architecture for Computer Vision. **Computer Vision and Pattern Recognition**, 2016.

[82] JIANG, Yu-Gang. Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 40, no. 2, p. 352-364, 2015.

[83] KOTHARI, Nisarg. YouTube8M: A Large-Scale Video Classification Benchmark. **Computer Vision and Pattern Recognition**, arXiv:1609.08675, 2016.

[84] COROVIC, Aleksa. et al. The Real-Time Detection of Traffic Participants Using YOLO Algorithm. **2018 26th Telecommunications Forum (TELFOR)**, p. 1-4, 2018.

[85] Open Images Dataset. Disponível em: <https://storage.googleapis.com/openimages/web/index.html>.

[86] COCO Dataset. Disponível em: < http://cocodataset.org>.

[87] A., Abhinand. et al. Detection of Moving Objects in a Metro Rail CCTV Video Using YOLO Object Detection Models. **Data Management, Analytics and Innovation, Proceedings of ICDMAI 2021**, vol. 1 p.183-195, 2021.

[88] AUNG, Htet; BOBKOV, Alexander V.; TUN, Nyan Lin. Face Detection in Real Time Live Video Using Yolo Algorithm Based on Vgg16 Convolutional Neural Network. 2021 **International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)**, p. 697-702, 2021.

[89] JAIN, Vidit; LEARNED-MILLER, Erik. FDDB: A Benchmark for Face Detection in Unconstrained Settings. University of Massachusetts, Amherst, 2010.

[90] BIENIK, Juraj; UHRINA, Miroslav; KORTIS, Peter. Impact of Constant Rate Factor on Objective. **Video Quality Assessment**, n. 4, vol. 15, 2017.

[91] LIN, Tsung-Yi. et al. Microsoft COCO: Common Objects in Context. Computer Vision and Pattern Recognition, arXiv:1405.0312, 2015

[92] Roboflow. Disponível em: <https://roboflow.com/>. Acesso em: Janeiro de 2022.