

**Deteção, reconhecimento e alerta de objetos em vídeo**

Licenciatura em Engenharia Informática

Marco António da Silva Ferreira

Leiria, « mês » de 2024



**Deteção, reconhecimento e alerta de objetos em vídeo**

Licenciatura em Engenharia Informática

Válter Santos Pedrosa

Trabalho de Projeto da unidade curricular de Projeto Informático realizado sob a orientação do Professor Doutor Paulo Costa e do Professor Doutor Filipe Neves

Leiria, « mês » de 2024

# Dedicatória

Inserir aqui a dedicatória. Trata-se de um elemento **facultativo**.

Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória.

Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória.

# Agradecimentos

Inserir aqui os agradecimentos. Trata-se de um elemento **facultativo**.

Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos.

Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos.

# Resumo

Inserir aqui o resumo. Trata-se de um elemento **obrigatório**.

Deve começar sempre numa página ímpar. Se ocupar um número par de páginas (p. ex. 2), deve ajustar-se o texto para que a próxima secção (abstract) se inicie numa página ímpar. O resumo deve acabar com a lista de palavras-chave.

**No resumo deve dar-se nota das principais ideias do trabalho (objetivos e conclusões).**

Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo.

Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo. Texto do resumo.

**Palavras-chave:** deteção, yolo, reconhecimento, IA.

# Abstract

Please insert here the abstract in English. This is a **mandatory** element.

The abstract should always start in an odd page. If the length is a multiple of two, the text should be adjusted in order to the next section start also in an odd page. The abstract should end with a list of keywords.

Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English.

Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English. Please insert here the abstract in English.

**Keywords:** maximum of 6 words separated by “,”

Índice

Trata-se de um elemento **obrigatório**. Nota: **o índice nunca figura do índice.**

[Dedicatória ii](#_Toc168404872)

[Agradecimentos iii](#_Toc168404873)

[Resumo iv](#_Toc168404874)

[Abstract v](#_Toc168404875)

[Lista de Figuras ix](#_Toc168404876)

[Lista de tabelas xi](#_Toc168404877)

[Lista de siglas e acrónimos xii](#_Toc168404878)

[1. Introdução 1](#_Toc168404879)

[2. Análise teórica 2](#_Toc168404880)

[2.1. Representação de uma imagem digital 2](#_Toc168404881)

[2.2. Processamento de imagens 2](#_Toc168404882)

[2.2.1. Tipos de processamento de imagens 2](#_Toc168404883)

[2.2.1.1. Visualização 2](#_Toc168404884)

[2.2.1.2. Reconhecimento 3](#_Toc168404885)

[2.2.1.3. “Sharpening” e restauração 3](#_Toc168404886)

[2.2.1.4. Reconhecimento de padrões 3](#_Toc168404887)

[2.2.1.5. Recuperação 3](#_Toc168404888)

[2.3. Deteção de objetos 4](#_Toc168404889)

[2.3.1. Métodos tradicionais 4](#_Toc168404890)

[2.3.1.1. Viola-Jones Detector 5](#_Toc168404891)

[2.3.1.2. Histogram of Oriented Gradients Detector 5](#_Toc168404892)

[2.3.1.3. Deformable Part-based Model 5](#_Toc168404893)

[2.3.2. Métodos deep learning 5](#_Toc168404894)

[2.3.2.1. Redes Neuronais Convolucionais 6](#_Toc168404895)

[2.3.2.1.1. Camada convolucional 6](#_Toc168404896)

[2.3.2.1.2. Camada de pooling 8](#_Toc168404897)

[2.3.2.1.3. Camada totalmente conectadas 9](#_Toc168404898)

[2.3.2.2. VGG16 9](#_Toc168404899)

[2.3.2.3. ResNet 10](#_Toc168404900)

[2.3.2.3.1. One-stage detectors 11](#_Toc168404901)

[2.3.2.3.1.1. SSD 12](#_Toc168404902)

[2.3.2.3.1.2. YOLO 12](#_Toc168404903)

[2.3.2.3.1.3. RetinaNet 13](#_Toc168404904)

[2.3.2.3.1.4. DETR 14](#_Toc168404905)

[2.3.2.3.2. Two-stage detectors 14](#_Toc168404906)

[2.3.2.3.2.1. R-CNN 14](#_Toc168404907)

[2.3.2.3.2.2. Faster R-CNN 15](#_Toc168404908)

[3. Metodologias e tecnologias 17](#_Toc168404909)

[3.1. Metodologia de trabalho 17](#_Toc168404910)

[3.1.1. Método tradicional 17](#_Toc168404911)

[3.1.2. Metodologias ágeis 17](#_Toc168404912)

[3.1.2.1. Scrum 19](#_Toc168404913)

[3.1.2.2. XP 20](#_Toc168404914)

[3.1.2.3. Kanban 21](#_Toc168404915)

[3.1.2.4. Scrumban 22](#_Toc168404916)

[3.1.3. Metodologia de trabalho escolhida 22](#_Toc168404917)

[3.2. Funcionalidades da aplicação 23](#_Toc168404918)

[3.2.1. Análise de requisitos 23](#_Toc168404919)

[3.2.2. Mockups 24](#_Toc168404920)

[4. Implementação 29](#_Toc168404921)

[4.1. Escolha do modelo de deteção 29](#_Toc168404922)

[31](#_Toc168404923)

[4.2. Arquitetura da aplicação 31](#_Toc168404924)

[5. Testes 32](#_Toc168404925)

[5.1. Teste de margem ideal para definir posição do objeto 32](#_Toc168404926)

[5.1.1. Condições do teste 32](#_Toc168404927)

[5.1.2. Resultados obtidos: 33](#_Toc168404928)

[5.1.2.1. Class “backpack” 33](#_Toc168404929)

[5.1.2.2. Class “sports ball” 35](#_Toc168404930)

[5.1.2.3. Class “Person” 36](#_Toc168404931)

[5.1.3. Conclusão: 37](#_Toc168404932)

[5.2. Teste da arquitetura para recolher frames de dispositivos 38](#_Toc168404933)

[5.2.1. Arquiteturas propostas 38](#_Toc168404934)

[5.2.1.1. Grab & Retrieve 38](#_Toc168404935)

[5.2.1.2. Parallel 38](#_Toc168404936)

[5.2.1.3. Open & Close 38](#_Toc168404937)

[5.2.1.4. Open & Close threading 38](#_Toc168404938)

[5.2.2. Condições de teste 39](#_Toc168404939)

[5.2.3. Resultados obtidos 39](#_Toc168404940)

[5.2.3.1. Grab & Retrieve 39](#_Toc168404941)

[5.2.3.2. Parallel 39](#_Toc168404942)

[5.2.3.2.1. Parallel 5 sec 40](#_Toc168404943)

[5.2.3.2.2. Parallel 10 sec 41](#_Toc168404944)

[5.2.3.3. Open & Close 41](#_Toc168404945)

[5.2.3.4. Open & Close threading 42](#_Toc168404946)

[5.2.4. Discussão dos Resultados 42](#_Toc168404947)

[5.2.5. Conclusão 43](#_Toc168404948)

[6. Conclusões ou Conclusão 44](#_Toc168404949)

[Bibliografia ou Referências Bibliográficas 45](#_Toc168404950)

[Anexos 46](#_Toc168404951)

[Glossário 47](#_Toc168404952)

§

# Lista de Figuras

Elemento a figurar, **quando aplicável**.

[Figura 2.1 - Aplicação filtro sobre imagem. Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 7](#_Toc168404953)

[Figura 2.2 - Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 7](#_Toc168404954)

[Figura 2.3 - Movimento do Kernel Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 8](#_Toc168404955)

[Figura 2.4 - Estrutura do VGG16, fonte: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918 9](#_Toc168404956)

[Figura 2.5 - VGG16, fonte: https://medium.com/@socjon/tba-920219647404 10](#_Toc168404957)

[Figura 2.6 - Comparação das taxas de erro de teste e treino em CNN's com número de camadas diferente, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/ 10](#_Toc168404958)

[Figura 2.7 - Skip Connections, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/ 11](#_Toc168404959)

[Figura 2.8 – Algoritmo SSD, fonte: Algoritmo de deteção R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 12](#_Toc168404960)

[Figura 2.9 – Algoritmo YOLO, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 13](#_Toc168404961)

[Figura 2.10 - Algoritmo RetinaNet, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 14](#_Toc168404962)

[Figura 2.11 - Algoritmo de deteção R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 15](#_Toc168404963)

[Figura 2.12 - Algoritmo de deteção Faster R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 16](#_Toc168404964)

[Figura 3.1 - Mockup início 25](#_Toc168404965)

[Figura 3.2 - Mockup layout horizontal 25](#_Toc168404966)

[Figura 3.3 - Mockup adicionar dispositivo 26](#_Toc168404967)

[Figura 3.4 - Mockup adicionar dispositivo IP 26](#_Toc168404968)

[Figura 3.5 - Mockup definições dispositivo 27](#_Toc168404969)

[Figura 3.6 - Mockup emitir alerta para um dispositivo 27](#_Toc168404970)

[Figura 3.7 - Mockup dispositivo expandido 28](#_Toc168404971)

[Figura 3.8 - Mockup página alertas 28](#_Toc168404972)

[Figura 4.1 - Gráfico comparação YOLOv7 vs. YOLOv5 vs. YOLOR and Vit Transformers, fonte: https://viso.ai/deep-learning/object-detection/ 30](#_Toc168404973)

[Figura 4.2 - Gráfico comparação YOlOv7 e YOLOv8, fonte: https://docs.ultralytics.com/models/yolov8/ 31](#_Toc168404974)

[Figura 5.1 - Gráfico cantos backpack a 5M 33](#_Toc168404975)

[Figura 5.2 - Gráfico cantos backpack a 10M 34](#_Toc168404976)

[Figura 5.3 - Gráfico cantos backpack a 15M 34](#_Toc168404977)

[Figura 5.4 - Gráfico cantos bola a 5M 35](#_Toc168404978)

[Figura 5.5 - Gráfico cantos bola a 10M 35](#_Toc168404979)

[Figura 5.6 - Gráfico cantos bola a 12M 36](#_Toc168404980)

[Figura 5.7 - Gráfico cantos person a 5M 36](#_Toc168404981)

[Figura 5.8 - Gráfico cantos person a 35M 37](#_Toc168404982)

[Figura 5.9 - Gráfico cantos person a 70M 37](#_Toc168404983)

[Figura 5.10 - Gráfico performance parallel 39](#_Toc168404984)

[Figura 5.11 - Gráfico performance parallel 5 segundos 40](#_Toc168404985)

[Figura 5.12 - Gráfico performance parallel 10 segundos 41](#_Toc168404986)

[Figura 5.13 - Gráfico performance open & close 41](#_Toc168404987)

[Figura 5.14 - Gráfico performance open & close threading 42](#_Toc168404988)

# Lista de tabelas

Elemento a figurar, **quando aplicável**.

[Tabela 1.1 - Texto ilustrativo da tabela 1. 3](#_Toc92389035)

# Lista de siglas e acrónimos

Elemento a figurar, **quando aplicável**.

|  |  |
| --- | --- |
| ESTG | Escola Superior de Tecnologia e Gestão |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Cuidados na elaboração da lista de siglas e acrónimos:

* Ordenação alfabética;
* Apenas as que sejam relevantes para a leitura do texto.

Adicionar mais entradas à tabela, caso seja necessário (a tabela não tem contornos, mas está no texto).

# Introdução

Introdução deve conter resumidamente os seguintes elementos:

* O objeto do trabalho (o tema);
* A justificação ou a pertinência do tema;
* Os objetivos do trabalho (gerais e específicos, perguntas a responder ou hipóteses a testar);
* Os métodos e as técnicas utilizados;
* Estrutura do trabalho.

# Análise teórica

Neste capítulo vamos explorar os conceitos teóricos necessários para compreender por completo a teoria em que se baseia esta aplicação, como funciona e perceber as bases que justificam as escolhas tomadas durante o desenvolvimento da aplicação.

## Representação de uma imagem digital

Uma imagem é representada pelas suas dimensões (altura e largura) consoante o número de pixéis. Um pixel é um ponto na imagem que tem uma tonalidade, opacidade ou cor específica. Normalmente é representado de uma das seguintes maneiras:

* Grayscale – Um pixel é um inteiro compreendido entre 0 e 255 em que 0 é completamente preto e 255 completamente branco.
* RGB – Um pixel é composto por 3 inteiros compreendidos entre 0 e 255 em que cada um representa a intensidade de vermelho (R), verde (G) e azul (B).
* RGBA – É uma extensão de RGB que acrescenta um campo alfa, que reflete a opacidade do pixel.

## Processamento de imagens

O processamento de imagens consiste no processo de transformar uma imagem numa forma digital e executar certas operações para extrair informação útil sobre a mesma. Este processo requere sequências fixas de operações que são executadas a cada pixel de uma imagem sendo que o processador da imagem executa a primeira sequência de operações na imagem, pixel a pixel. Quando completa, começa a executar a segunda operação e assim sucessivamente. O valor de saída destas operações pode ser calculado a cada pixel da imagem.

### Tipos de processamento de imagens

Existem 5 tipos principais de processamento de imagens.

### Visualização

Este primeiro tipo de processamento envolve a conversão de dados na imagem para um formato visual que possa ser facilmente interpretado por humanos. Existem várias técnicas possíveis a ser utilizadas, nomeadamente melhoramento de contraste, segmentação ou criação de imagens tridimensionais a partir de dados de imagens bidimensionais. Um bom exemplo de aplicação desta técnica é na medicina, onde são aplicadas técnicas de visualização em MRI’s (Magnetic Resonance Imaging) para obter um diagnóstico ou explorar conectividade no cérebro de um indivíduo.

### Reconhecimento

No contexto de processamento de imagens, o reconhecimento refere-se à identificação e categorização de objetos ou padrões específicos dentro de uma imagem. Isso pode envolver a deteção de rostos humanos, identificação de caracteres de texto ou reconhecimento de objetos em cenas complexas. São utilizadas técnicas avançadas de machine learning e redes neurais convolucionais (CNNs) frequentemente para melhorar a precisão e a eficácia dos sistemas de reconhecimento de imagens. É neste tópico que este projeto se insere.

### “Sharpening” e restauração

“Sharpening” e restauração de imagens são processos usados para melhorar a qualidade de imagens que tenham sido distorcidas, quer seja por ruído, desfoque ou compressão. “Sharpening”, por um lado, foca em realçar detalhes e bordas na imagem, tornando os elementos mais nítidos e claros. A restauração, por outro, tenta recuperar a imagem original corrigindo distorções e removendo ruídos. Estes processos utilizam, geralmente, técnicas matemáticas e algoritmos de filtragem.

### Reconhecimento de padrões

O reconhecimento de padrões envolve a análise de imagens para identificar e categorizar padrões ou características específicas que podem ser repetidas ou significativas. Isto pode incluir a identificação de texturas, formas, ou regularidades em conjuntos de dados de imagem. Esta área é fundamental em aplicações como análise de imagens médicas, inspeção de qualidade de produtos em fábricas e análise de solo em imagens de satélite. Algoritmos de machine learning são frequentemente utilizados para treinar sistemas para reconhecer e interpretar esses padrões com alta precisão.

### Recuperação

A recuperação é uma técnica de computer vision que envolve encontrar imagens semelhantes a uma original numa grande base de dados. A recuperação de imagens é crucial em áreas como bancos de dados multimídia, sistemas de vigilância e arquivos digitais, onde o acesso rápido e preciso a imagens relevantes é essencial.

Existem duas técnicas para realizar a recuperação de imagens, CBIR e TBIR. CBIR significa Content-Based Image Retrieval e retrata que o modelo compara a imagem fornecida com as da base de dados através do conteúdo visual e as suas características intrínsecas, ou seja, analisa a cor, textura, forma entre outros atributos. TBIR significa Text-Based Image Retrieval e esta abordagem baseia-se em descrições textuais e metadados associados às imagens para realizar a busca. As imagens são indexadas com base em palavras-chave, legendas, anotações ou outras informações textuais fornecidas.

## Deteção de objetos

Como expresso anteriormente, este projeto enquadra-se na categoria de reconhecimento dentro dos vários tipos de processamento de imagens. Com este projeto pretendemos identificar objetos selecionados em imagens que são frames extraídos de câmaras conectadas à nossa aplicação. Existem dois tipos de processamento de imagens em computer vision: classificação de imagens e deteção de objetos. Os algoritmos de classificação apenas identificam a melhor classe que caracteriza a imagem designando uma “label” à imagem inteira baseado no seu conteúdo predominante, enquanto os algoritmos de deteção - implementados neste projeto - identificam quantos objetos de uma determinada categoria estão na imagem e onde cada objeto se encontra. Existem vários métodos relacionados à deteção de objetos, sendo que estes podem pertencer a abordagens tradicionais ou deep learning.

### Métodos tradicionais

Os métodos tradicionais de deteção de objetos envolvem geralmente extração de características e algoritmos de machine learning como Viola-Jones Detector, Histogram of Oriented Gradients (HOG) Detector e Deformable Part-based Model (DPM). Estas abordagens tradicionais contribuíram significativamente para o desenvolvimento de métodos de deteção de objetos, oferecendo “insights” na consideração de diferentes características e desafios de objetos.

### Viola-Jones Detector

Este algoritmo utiliza "Haar-like features" que são características formadas por elementos retangulares que capturam diferenças na intensidade dos píxeis. Estas características são combinadas em secções em cascata, onde cada estágio descarta rapidamente as áreas da imagem que não contêm elementos de interesse. As localizações que passam por todos os estágios são consideradas como contendo um objeto. A eficiência deste método reside na capacidade de filtrar rapidamente a maioria das áreas negativas (sem objeto), analisando detalhadamente apenas as áreas promissoras, o que o torna adequado para aplicações em tempo real.

### Histogram of Oriented Gradients Detector

Este método (HOG) funciona calculando gradientes de intensidade em células locais, que capturam orientações de bordas e contornos na imagem. Para cada célula são gerados histogramas dos gradientes, que são normalizados em blocos sobrepostos para garantir robustez contra variações de brilho e contraste. O vetor de atributos resultante representa a distribuição das arestas em toda a imagem e é utilizado como entrada para um classificador, como o SVM, para detetar objetos específicos.

### Deformable Part-based Model

O Deformable Part-based Model (DPM) é um método avançado de deteção de objetos que modela um objeto como uma coleção de partes interconectadas, onde cada parte pode mover-se ligeiramente em relação às outras. Este método utiliza um modelo hierárquico em que a aparência de cada parte e as relações espaciais entre as partes são capturadas por descritores de características, como o HOG. O DPM envolve um processo de aprendizagem que determina a melhor configuração das partes e as suas deformações permitidas, criando um modelo flexível que pode lidar com variações de pose, escala e forma do objeto. Esta técnica é particularmente eficaz para a deteção de objetos em cenários complexos, onde a rigidez dos modelos tradicionais anteriores pode falhar.

### Métodos deep learning

Machine learning (ML) é um ramo de IA e assenta na premissa que a máquina adquire dados e aprende com eles. Em vez de escrever código específico num programa com instruções específicas para completar uma tarefa, ML permite que o sistema aprenda e reconheça padrões por si próprio e faça previsões.

Deep learning é um campo especializado de machine learning assente em Redes Neuronais Concolucionais que envolve aprendizagem em diferentes estágios. Este método é caracterizado pela sua estrutura em camadas, cada camada adicionando conhecimento à anterior. Deste modo, o deep learning treina a máquina para tentar espelhar o funcionamento natural do cérebro humano.

Os métodos de deep learning têm ganho maior atenção e sucesso nos últimos anos. As Redes Neuronais Convolucionais (CNNs) são uma tecnologia chave no deep learning para deteção de objetos. Estas conseguem aprender a extrair automaticamente características relevantes de imagens e aprender padrões complexos que são representativos de diferentes categorias de objetos. Existem dois tipos de arquiteturas de deteção de objetos baseadas em CNNs, “one-stage detectors” e “two-stage detectors”.

### Redes Neuronais Convolucionais

Rede neuronal convolucional ou CNN (do inglês Convolutional Neural Network) é um algoritmo de Deep Learning. O pré-processamento exigido numa CNN é muito menor quando comparado a outros algoritmos de classificação. Enquanto em métodos primitivos os filtros são desenhados manualmente (hand-engineered), as CNN’s, com treino suficiente, têm a habilidade de aprender estes filtros/características. A arquitetura de uma rede neuronal convolucional é análoga à do padrão de conectividade entre neurónios no cérebro Humano e foi inspirada pela organização do córtex visual.

As redes neuronais convolucionais diferem das outras redes neuronais pela sua performance superior com entradas (inputs) de vídeo, áudio ou fala. Estas redes têm três camadas principais: camada convolucional, camada de “pooling” e camada “fully-connected”.

### Camada convolucional

Esta é a primeira camada de uma rede convolucional, é onde ocorre a maior parte do processamento e tem como objetivo reduzir as imagens numa forma mais fácil de processar sem perder características críticas que permitam manter uma boa precisão.

Uma imagem com texto, captura de ecrã, quadrado

Descrição gerada automaticamente

Figura 2.1 - Aplicação filtro sobre imagem. Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

No caso de imagens com vários canais, como é o caso do RGB, o cálculo da imagem “convolved” é da seguinte maneira:

Uma imagem com texto, diagrama, captura de ecrã, Esquema

Descrição gerada automaticamente

Figura 2.2 - Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Na figura acima podemos verificar, a verde, uma imagem de entrada de dimensões 5x5x1 e, a amarelo o filtro ou kernel (k) aplicado à imagem representado pela seguinte matriz ([1,0,1]; [0,1,0]; [1,0,1]).

Uma imagem com esboço, design, cubo

Descrição gerada automaticamente

Figura 2.3 - Movimento do Kernel Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Esta imagem representa o movimento em que o filtro opera sobre uma imagem. O filtro move-se da esquerda para a direita até atingir toda a largura passando para baixo para a esquerda e repete o processo até toda a imagem ser percorrida.

As CNN’s não necessitam de estar limitadas a uma única camada convolucional. Por norma, a primeira camada convolucional é responsável por capturar características de baixo nível como arestas ou cor. Com camadas adicionais, a arquitetura adapta-se para reconhecer características mais detalhadas possibilitando uma rede com uma compreensão completa das imagens.

### Camada de pooling

Na camada de pooling ocorre a redução da dimensionalidade, reduzindo o número de parâmetros no input. Tal como a camad aconvolucional, esta camada varre um filtro sobre toda a matriz de entrada com a diferenca que o filtro não contém qualquer peso. Ao invés, o filtro aplica uma função de agregação aos valores dentro do campo recetivo, preenchendo o vetor de saída. Existem dois tipos de pooling principais:

* **Max pooling**: Nesta abordagem o filtro escolhe o pixel com maior valor e envia-o para o vetor de saída.
* **Average pooling**: Neste método, o filtro calcula a média dos valores dos pixéis no campo recetivo e envia o valor para o vetor de saída

Embora se perca muita informação nesta camada, acaba por ser vantajoso para a CNN uma vez que reduz a complexidade, melhorando a eficiência e limita o risco de overfitting.

### Camada totalmente conectadas

Como mencionado anteriormente, os valores dos pixéis das imagens de entrada não estão diretamente relacionados à camada de saída em camadas parcialmente conectadas. Por outro lado, nesta camada, como o nome sugere, as camadas estão totalmente conectadas (fully-conected) pelo que cada nó na camada de saída está diretamente conectado a um nó da camada anterior.

Esta camada executa a tarefa de classificação com base nas características extraídas através das camadas anteriores e os seus diferentes filtros. Enquanto as camadas convolucionais e de pooling tendem a usar funções ReLu, esta última normalmente recorre a uma função de ativação “softmax” para classificar apropriadamente os inputs, produzindo uma probabilidade entre 0 e 1.

### VGG16

VGG16 é uma arquitetura de uma CNN desenvolvida pelo Visual Geometry Group e é considerada um dos melhores métodos de computer vision. A estrutura desta rede conta com 13 camadas convolucioanais, divididas em blocos onde cada bloco é seguido por uma camada de pooling do tipo Max Polling, contando com 5 destes elementos. Para concluir, a arquitetura conta com 3 camadas fully connected. A figura abaixo representa a arquitetura do VGG16.

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

Figura 2.4 - Estrutura do VGG16, fonte: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

O 16 no nome refere-se às 16 camadas, 13 convolucionais mais 3 fully connected, com peso (que possibilitam aprendizagem) das 21 que constituem o modelo.

As camadas de convolução aplicam filtros de 3x3 píxeis com stride de 1 e sempre o mesmo padding e maxpool layer de 2x2 com stride de 2. Estas configurações de filtros pequenos nas camadas de convolução permitem manter a resolução das características enquanto o pooling ajuda a reduzir o custo computacional, reduzindo para metade a resolução das características.

As seguintes (e últimas) camadas são as camadas fully connected. As primeiras duas integram 4096 canais cada e a terceira, por executar uma classificação “1000-way ILSVRC” contém 1000 canais, um para cada classe. A última camada é a camada de softmax que classifica as imagens nas diferentes classes.

Figura 2.5 - VGG16, fonte: https://medium.com/@socjon/tba-920219647404

### ResNet

Após a primeira versão de arquiteturas baseadas em CNN’s (AlexNet), de um modo geral, as arquiteturas seguintes consistiam sempre em camadas adicionais para reduzir a taxa de erro. Com o crescimento do número de camadas, verificou-se que a taxa de erro de treino e testes também aumenta (consultar figura abaixo). Percebeu-se que o problema estava relacionado com o gradiente tornar-se 0 ou demasiado grande, a este fenómeno chama-se “Vanishing gradient”.

Uma imagem com texto, file, Tipo de letra, diagrama

Descrição gerada automaticamente

Figura 2.6 - Comparação das taxas de erro de teste e treino em CNN's com número de camadas diferente, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

A Residual Network ou ResNet, propõe um conceito novo chamado Blocos Residuais. Este método propõe utilizar uma técnica (Skip Connections) que permite saltar camadas intermediárias conectando ativações entre camadas, formando Blocos Residuais. A arquitetura desta rede consiste num empilhamento destes Blocos Residuais.

Uma imagem com texto, diagrama, Tipo de letra, file

Descrição gerada automaticamente

Figura 2.7 - Skip Connections, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

A abordagem por trás desta rede é que, em vez das camadas aprenderem o mapeamento subjacente, permitimos que a rede se ajuste ao mapeamento residual. Por isso, em vez de aprender uma função de mapeamento H(x), os blocos residuais aprendem a função residual F(x), onde H(x) = F(x) + x, o que facilita o treino de redes muito profundas. A vantagem de adicionar este tipo de Skipping Connection é que se alguma camada prejudicar o desempenho da arquitetura, é ignorada pela regularização, o que possibilita o treino de uma rede neuronal profunda sem ocorrer o fenómeno Vanishing Gradient.

### One-stage detectors

Os One-stage detectores são desenhados para prever coordenadas da bounding box (caixa que delimita um objeto detetado) e probabilidades de classe para vários objetos numa única passagem pela rede. Estes modelos são conhecidos pela sua simplicidade e eficiência, uma vez que eliminam a necessidade de uma etapa separada para elaboração de propostas. O objetivo destes detectors é examinar amostras densas de possíveis localizações de objetos e prever a presença de um objeto e a sua bounding box associada numa única fase (single-shot). Por realizarem a deteção numa passagem única, estes algoritmos, são mais rápidos na deteção, o que os torna ideais para aplicações com deteção em tempo real. Esta abordagem sacrifica ligeiramente a precisão dos modelos para se focar na sua rapidez, e tendem a ter dificuldade a detetar objetos pequenos e tratar instâncias de objetos com variações de tamanho significativas. Os one-stage detectors mais populares são YOLO, SSD, RetinaNet e CenterNet.

### SSD

SSD, Single-Shot Detector, é um dos modelos mais rápidos para previsões multi-box de deteção de objetos para aplicações de real-time. Este modelo atinge uma taxa de frames por segundo (fps) quase 5x maior que o R-CNN. A sua arquitetura pode ser dividida em três componentes/etapas principais. A primeira etapa é a extração de características onde todos os principais mapas de características são selecionados recorrendo a uma CNN pré-treinada como VGG16 ou ResNet. A seguir este modelo conta com várias camadas convolucionais, cada uma responsável por detetar objetos de diferentes escalas. As camadas mais profundas são responsáveis pela deteção de objetos maiores, enquanto as camadas mais superficiais detetam objetos menores. Por último encontramos camadas de supressão não máxima para reduzir a taxa de erro causada por bounding-boxes repetidas uma vez que cada ponto em cada camada convolucional pode ter várias anchor boxes associadas.



Figura 2.8 – Algoritmo SSD, fonte: Algoritmo de deteção R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### YOLO

O YOLO (You Only Look Once) é um dos métodos mais populares para deteção de objetos da atualidade. Existem várias versões deste modelo, à data de hoje há registo até a versão 9. Este modelo utiliza um dos melhores arquétipos de redes neuronais que lhe conferem alta precisão e velocidade de processamento, características que justificam a sua popularidade.

O YOLO trata a deteção de objetos como um problema de regressão única, direto da imagem para as bounding boxes e probabilidades da classe. Ao contrário dos métodos que usam regiões propostas como os métodos da família R-CNN, este método divide a imagem numa grade e faz previsões para cada célula da grade.

A arquitetura do YOLO conta com três etapas principais para atingir o seu objetivo. Na primeira destas fases, o algoritmo define blocos residuais, isto é, divide a imagem numa grade de SxS células onde cada célula atua como um ponto central para previsão. Na segunda técnica, cada ponto central para uma previsão particular é considerado para a criação das bouding boxes. Embora as tarefas de classificação funcionem bem para cada grade, a segregação das bounding boxes para cada previsão feita é mais complexa. A terceira, e última técnica é implementada a métrica de IOU (intersection of union) para calcular a melhor bounding box para cada objeto.



Figura 2.9 – Algoritmo YOLO, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### RetinaNet

Introduzido em 2017, o RetinaNet tornou-se um dos melhores modelos com dingle shot detection que ultrapassava outros métodos populares. Quando foi lançado, as suas capacidades excediam os modelos SSD e YOLOv2, uma vez que mantinha a velocidade destes modelos e ainda competia com os da família R-CNN em termos de precisão. Este modelo foi desenvolvido para resolver um problema comum dos modelos de deteção de objetos, que é a deteção de objetos pequenos.

A arquitetura do modelo RetinaNet é construída com base em três padrões, nomeadamente o modelo ResNet, feature pyramid network (FPN) e perda focal. O ResNet é uma arquitetura de CNN desenhada para superar o problema de desaparecimento de gradientes durante o treino de redes muito profundas. A FPN é a principal inovação do RetinaNet e permite ao modelo detetar objetos em várias escalas uma vez que ajuda a combinar os recursos semanticamente ricos de imagens de resolução mais baixa com os recursos semanticamente fracos das imagens de resolução mais alta. Por fim, neste modelo a perda de entropia cruzada nos modelos anteriores é substituída pela perda focal que permite ao modelo concentrar-se nas regiões que são mais difíceis de classificar durante o treino, o que pode melhorar o desempenho em classes minoritárias ou em objetos pequenos.

Figura 2.10 - Algoritmo RetinaNet, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### DETR

### Two-stage detectors

Os two-stage detectors, ao contrário do one-stage, seguem um processo de duas etapas. Na primeira etapa, estes detetores geram um conjunto de propostas de regiões candidatas a localizações com objetos. Estas propostas são depois refinadas e classificadas na segunda etapa. Two-stage detectors tendem a ter uma precisão maior, mas a ser mais lentos quando comparados com os one-stage detectors, devido à etapa adicional de elaboração de propostas. Os two-stage detectors mais populares são R-CNN, SPPNet, Fast R-CNN, Faster R-CNN, FPN e S2ANet.

### R-CNN

O método R-CNN (Region-Based Convolutional Neural Networks) tenta extrair as características mais importantes da imagem permitindo focar apenas nas zonas selecionadas. Este processo de seleção permite gerar sub-segmentações (normalmente perto de 2000) e selecionar entradas candidatas para o algoritmo de deteção. Uma vez que o algoritmo de seleção termina com sucesso, cada proposta de região é redimensionada para um tamanho fixo e passa por uma CNN pré-treinada para extrair características. Por fim, as características extraídas são classificadas sobre presença de objetos e identificação da classe presente. Para obter o melhor resultado, a previsão é feita por um modelo de classificação enquanto um modelo de regressão é usado para classificar as bouding boxes para as regiões propostas.



Figura 2.11 - Algoritmo de deteção R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### Faster R-CNN

Embora o modelo anteriormente descrito (R-CNN) ser capaz de atingir resultados desejáveis, há alguns problemas associados, principalmente na velocidade do modelo. Foi então introduzido o modelo Fast R-CNN para tentar combater estes problemas associados à velocidade do modelo. Neste método toda a imagem é passada para a CNN pré-treinada em vez de sub-segmentações para ober um mapa de características e, posteriormente passa por uma camada adicional chamada RoI (Region-of-Interest) que recebe duas entradas do modelo pré-treinado e aplica o algoritmo Selective Search para devolver uma camada totalmente conectada com um output.

O modelo Faster R-CNN é uma das melhores versões da família R-CNN que melhora consideravelmente a velocidade do modelo quando comparada com os seus antecessores. Enquanto o R-CNN e o Fast R-CNN utilizam um algoritmo de Selective Search para calcular as propostas de região, o Faster R-CNN substitui este algoritmo com uma rede superior de propostas de região. Esta rede de propostas de região (RPN, Region Proposal, Network) calcula imagens de uma ampla variedade e diferentes escalas para produzir resultados eficazes e permite reduzir a margem de tempo computacional em 10ms por imagem. A RPN consiste numa camada convolucional que obtém os mapas de características essenciais de cada pixel e para cada mapa tem várias anchor boxes (anchor boxes são caixas predefinidas usadas como referência para calcular as bounding boxes) com várias escalas, tamanhos e proporções diferentes.



Figura 2.12 - Algoritmo de deteção Faster R-CNN, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

# Metodologias e tecnologias

## Metodologia de trabalho

Quando se fala em desenvolvimento de software, existem duas maneiras de abordar a sua gestão e planeamento de modo a tornar a gestão de projeto mais eficiente e eficaz. Por um lado, temos as metodologias tradicionais, como o modelo cascata, que apostam numa abordagem sequencial e planeamento detalhado desde o início do projeto. Enquanto, em contraste, as metodologias ágeis dão prioridade à flexibilidade, colaboração e adaptação contínua ao longo do projeto.

### Método tradicional

A abordagem tradicional de gestão de projetos, também conhecida como método cascata, segue um processo linear e sequencial onde só se passa à fase seguinte quando a anterior é completada. É uma abordagem estruturada e orientada a planos com foco no planeamento completo e documentação antecipada. Todos os projetos, independentemente da sua natureza, seguem o mesmo ciclo de vida que incluem as fases de viabilidade, planeamento, design, construção, teste, produção e suporte. O projeto é planeado todo no início e não há qualquer mudança de requisitos, uma vez que nesta abordagem assume-se que os requisitos são fixos e o tempo e custo variáveis. Como tal, não é adequado a projetos de grande escala em que os requisitos para todo o projeto não se conseguem definir numa fase inicial.

### Metodologias ágeis

Ao contrário do método anterior, o foco das metodologias ágeis é o trabalho em equipa, colaboração com o cliente e flexibilidade. É uma abordagem iterativa que se concentra mais na incorporação do cliente e lançamentos contínuos em cada iteração do desenvolvimento do projeto. O conceito chave desta abordagem é que aprofunda a evolução das mudanças e esforço colaborativo para obter resultados em vez de um processo predefinido. O planeamento adaptativo é a principal característica das metodologias ágeis que convence muitos gestores de projeto.

Na seguinte tabela podemos ver a comparação de alguns aspetos comuns a ambas as abordagens.

|  |  |  |
| --- | --- | --- |
| Parâmetros | Metodologias Ágeis | Gestão de projeto tradicional |
| Flexibilidade e adaptabilidade | Ênfase na flexibilidade e adaptabilidade | Ênfase no planeamento e previsibilidade |
| Requisitos do projeto | Prioriza a satisfação do cliente e software funcional | Prioriza o seguimento de um plano rigoroso e atender aos requisitos do projeto |
| Comunicação | Encoraja comunicação e colaboração cara-a-cara | Confia na documentação escrita. Comunicação formal |
| Fases de projeto e planificação | Sprints curtos para planeamento e entrega. Permite alterações e ajustes ao longo do projeto | Fases de projeto mais longas para planeamento e entrega. Tem um plano rígido que e difícil alterar |
| Funcionalidade de equipa | Equipas multifuncionais | Equipas com funções divididas |
| Organização | Equipas auto-organizadas | Equipas organizadas hierarquicamente |
| Estrutura organizacional | Iterativa | Linear |
| Escala de projeto | Grande escala | Escala pequena e média |
| Modelo de desenvolvimento | Modelo de entrega evolutivo | Modelo de ciclo de vida |
| Requisitos do utilizador | Inputs interativos | Claramente definidos antes de implementação |
| Envolvimento do cliente | Alto | Baixo |

Fonte: <https://www.knowledgehut.com/blog/agile/agile-project-management-vs-traditional-project-management> (com ajustes)

Optamos por seguir uma metodologia ágil por não saber de antemão os requisitos que vai ter a nossa aplicação, uma vez que esta abordagem permite-nos desenvolver a aplicação de maneira mais flexível e colaborativa. Vamos então analisar as principais metodologias ágeis e a forma como abordam a gestão dos projetos.

### Scrum

O scrum é a framework mais utilizada de todas as metodologias ágeis e caracteriza-se por etapas ou ciclos de desenvolvimento, conhecidos como sprints e pela maximização do tempo de desenvolvimento de um produto de sofware em direção ao Product Goal.

Uma equipa de Scrum é composta por 3 elementos com papeis bem definidos, o Product Owner, o Scrum Master e a equipa de desenvolvimento. O PO é o elemento que liga o cliente à equipa de desenvolvimento, assumindo a função garantir que a visão do cliente é entendida pela equipa de desenvolvimento e definindo os critérios de sucesso. É este que define o que deve ser feito, e em que ordem deve ser feito, atribuindo valor aos requisitos. O SM é referido como o facilitador do projeto uma vez que este certifica que as boas práticas da metodologia Scrum são seguidas, verifica relações saudáveis entre o PO e os programadores e elimina quaisquer obstáculos que impeçam uma colaboração produtiva. Por fim, a equipa de desenvolvimento, como o nome indica, é o conjunto de membros que trabalha em conjunto para criar, estar e lançar versões incrementais do produto final.

Além da equipa Scrum, esta metodologia conta com os seguintes eventos Scrum: Daily Scrum, Sprint, Sprint planning, Sprint review e Sprint retrospective.

O Daily Scrum consiste numa curta reunião diária que ocorre todos os dias no mesmo sítio à mesma hora. Nesta reunião a equipa avalia o trabalho realizado no dia anterior e planeia o que será feito nas próximas 24 horas.

A Sprint representa o período em que o trabalho deve estar concluído, não maior que um mês. Uma nova sprint começa quando a anterior acaba.

A Sprint planning meeting marca o início de uma nova sprint. Nesta reunião, toda a equipa Scrum participa e define objetivos, sendo que no final pelo menos um incremento de software utilizável deve ser produzido.

No fim de cada sprint, acontece a sprint review em que a equipa scrum mostra o trabalho realizado ao PO e stakeholders.

Por fim, a sprint retrospective, dá lugar à equipa para discutir o que correu bem e o que pode ser melhorado em relação à última sprint com o objetivo de atingir uma melhoria contínua.

No Scrum existem ainda 2 backlogs. O Product backlog que contém a lista das tarefas para sprints futuras e s Sprint Backlog que inclui as tarefas a realizar na sprint atual.

### XP

XP é outra metodologia ágil bastante presente no mundo dos projetos de desenvolvimento de software. Tem como objetivo criar sistemas de alta qualidade, assente nos princípios de proximidade com o cliente, testes constantes e ciclos de desenvolvimento curtos. Os 5 componentes básicos do XP são: comunicação, simplicidade, feedback, respeito e coragem. Esta abordagem aposta na velocidade e simplicidade com ciclos curtos de desenvolvimento e menos documentação.

No XP o cliente deve participar ativamente do processo de desenvolvimento. Tudo precisa da comunicação com o cliente. Este deve receber o melhor resultado possível a cada sprint, ver o progresso no sistema e ser informado de mudanças de planos para que possa dar a sua opinião.

Outra prática fundamental no XP é o pair programming. O pair programming consiste numa programação a pares num único computador em que um tem o papel de escrever o código enquanto o outro está sempre ao lado a rever. Desta maneira, erros e detalhes que possam passar despercebidos a um programador, o outro consegue cobri-los diminuído a probabilidade de falhas.

Para perceber mais facilmente o que difere o XP do Scrum podemos consultar a seguinte tabela:

|  |  |
| --- | --- |
| Scrum | Extreme Programming |
| Trabalho em iterações (sprints) podem ir até 1 mês | Trabalho em iterações de apenas 1-2 semanas |
| Modelos Scrum não permitem mudanças no seu cronograma ou diretrizes. | Permite mudanças nos cronogramas definidos |
| Ênfase na auto-organização | Ênfase em fortes práticas de engenharia |
| Equipa determina a sequência na qual o produto é desenvolvido | Equipa segue uma ordem de prioridade pré-determinada |
| A framework Scrum não está completamente descrita. Para adotá-la é necessário completá-la com métodos de trabalho, como XP ou Kanban | Pode ser aplicado diretamente a uma equipa. O XP é também conhecido pelas suas características “Ready-to-apply” |
| Não enfatiza as práticas de engenharia de software que os programadores devem usar | Enfatiza as técnicas de programação que os programadores devem seguir para garantir um melhor resultado |
| Requer que os programadores estejam conscientes da adoção de métodos de engenharia para garantir um melhor progresso ou qualidade | É muito rígido na adoção de métodos de engenharia, como pair programming, design simples, reestruturação, para garantir um melhor progresso ou qualidade. |
| O Product Owner define a prioridade das tarefas de acordo com os requisitos | O cliente define a prioridade das tarefas e analisa os lançamentos. |
| Existe flexibilidade para a equipa ajustar a prioridade das tarefas se necessário | A equipa não pode alterar a prioridade dos requisitos definidos pelo cliente |
| Valores: - Abertura  - Foco  - Compromisso | Valores: - Comunicação  - Simplicidade  - Feedback |
| Menor envolvimento do cliente no projeto | Maior envolvimento do cliente no projeto |

Fonte: https://www.geeksforgeeks.org/difference-between-scrum-and-xp/

### Kanban

A metodologia kanban visa uma melhoria contínua, flexibilidade no gerenciamento de tarefas, e fluxo de trabalho aprimorado. Por ser uma abordagem ilustrativa, o progresso de todo o projeto é facilmente compreendido em poucos segundos.

O Kanban, ao contrário do scrum, não possui um prazo definido para quando as tarefas devem ser concluídas. Em vez disso, esta metodologia é gerida pela prioridade dos requisitos num quadro Kanban. Este quadro é dividido em três estados para uma tarefa: pendente (to do), em desenvolvimento (doing) e concluída (done). As tarefas/requisitos são representadas por cartões visuais que transitam entre os estados até serem finalizadas. Assim, a equipa consegue ter uma noção visual do desenvolvimento do projeto.

### Scrumban

O Scrumban, como o nome sugere, é uma metodologia ágil que combina elementos do Scrum e Kanban. Esta abordagem envolve aplicar os princípios de visualização do fluxo de trabalho e processos flexíveis do Kanban a uma equipa da Framework Scrum. Porém, este princípio, remove alguns dos aspetos mais rígidos do Scrum e deixa as equipas formar uma abordagem personalizada para o desenvolvimento do projeto.

O Scrunban adota um quadro similar ao quadro Kanban na medida em que é a ferramenta principal de fluxo de trabalho com a diferença que permite adicionar as colunas necessárias ao quadro para marcar a fase de progresso para cada requisito.

O Scrum, define limites de tempo e tarefas por cada sprint, o contrário acontece no Kanban que se foca no fluxo de trabalho contínuo. No Scrumban é estabelecido um limite para a quantidade de trabalho que a equipa pode realizar num momento e este limite é o total de cartões no quadro.

Enquanto no Scrum cada tarefa é atribuída a um membro específico da equipa, no Scrumban o foco é estabelecer a ordem de prioridade das tarefas no quadro e cabe à equipa distribuir as tarefas a realizar pelos diversos membros.

Embora esta abordagem não tenha todas as reuniões típicas da estrutura Scrum, no Scrumban pode haver reuniões curtas durante as sprints para a equipa discutir planos e desafios encontrados.

### Metodologia de trabalho escolhida

Das metodologias ágeis apresentadas, excluímos já de partida o XP por não termos o cliente sempre presente no desenvolvimento da aplicação nem condições para aplicar o pair programming que esta abordagem requere. Das restantes, Scrum, Kanban e Scrumban decidimos escolher a última (Scrumbam) por oferecer o melhor de cada uma das metodologias em que esta se baseia enquanto descarta os aspetos menos favoráveis das mesmas. Deste modo conseguimos aproveitar não só a flexibilidade que o Kanban nos permite, mas também os princípios de planeamento Scrum, permitindo-nos adaptar nosso processo de desenvolvimento de acordo com as necessidades do mesmo.

Para aplicar a metodologia ágil Scrumban, realizamos uma reunião semanal com o Product Owner e Scrum Master, interpretados pelos professores orientadores. Nesta reunião discutimos o trabalho desenvolvido na última sprint que acaba nessa mesma reunião e determinamos o que deve ser feito para a próxima. Utilizamos a framework Jira Software da Atlassian que nos permite criar Issues que representam as tarefas/requisitos que a nossa aplicação tem ou vai ter. Aqui também conseguimos criar o quadro kanban que nos proporciona uma noção visual do desenvolvimento do trabalho e onde podemos colocar os Issues previamente criados. Conseguimos ainda criar e terminar as várias sprints ao longo do processo de desenvolvimento.

## Funcionalidades da aplicação

### Análise de requisitos

RF1. Adicionar dispositivo: Enquanto utilizador quero adicionar novos dispositivos à página de dispositivos para começar a captura e deteção nesses dispositivos.

RF2. Editar dispositivo: Enquanto utilizador quero editar dispositivos presentes na página de dispositivos para alterar as configurações dos mesmos.

RF3. Pausar captura: Enquanto utilizador quero parar a captura de frames de um dispositivo para parar de atualizar os frames desse dispositivo.

RF4. Retomar captura: Enquanto utilizador quero retomar a captura de frames de um dispositivo pausado para voltar a capturar frames desse dispositivo.

RF5. Capturar live: Enquanto utilizador quero capturar frames sem delay (ver em live) para ver em direto o estado de um dispositivo.

RF6. Expandir dispositivo: Enquanto utilizador quero expandir a janela de captura de um dispositivo para ver com maior detalhe os frames capturados do mesmo.

RF7. Remover dispositivo: Enquanto utilizador quero remover dispositivos da janela de dispositivos para remover dispositivos que já não tenha interesse.

RF8. Alterar layout: Enquanto utilizador quero alterar o layout da disposição dos dispositivos para escolher qual o layout que quero que os dispositivos estejam dispostos.

RF9. Alterar delay: Enquanto utilizador quero alterar a frequência com que um dispositivo captura e processa um frame para escolher o tempo que achar adequado para cada dispositivo.

RF10. Adicionar objetos a detetar: Enquanto utilizador quero adicionar objetos que quero detetar num dispositivo para escolher os objetos que considero relevantes para deteção no dispositivo.

RF11. Remover objetos a detetar: Enquanto utilizador quero remover objetos que já não quero detetar num dispositivo para retirar os objetos que já não tenha interesse a detetar num dispositivo.

RF12. Definir tempo para alerta de objeto: Enquanto utilizador quero definir o tempo máximo de um objeto parado antes de emitir um alerta para ser notificado quando um objeto estiver parado num dispositivo há mais tempo que o definido.

RF13. Filtrar alertas: Enquanto utilizador quero filtrar os alertas emitidos para visualizar apenas os alertas que se enquadran nos parâmetros que procuro.

RF14. Eliminar alertas: Enquanto utilizador quero eliminar os alertas emitidos para eliminar os alertas que considero irrelevantes ou que já foram analisados e tratados.

RF15. Exibir detalhes de alerta: Enquanto utilizador quero visualizar os detalhes de um alerta para analisar mais informações associadas ao alerta emitido.

RF16. Gravar configuração: Enquanto utilizador quero guardar os dispositivos e as suas configurações para quando sair da aplicação poder reabrir no estado em que estava.

RF17. Importar configuração: Enquanto utilizador quero importar uma configuração que tenha guardado previamente para poder importar os dispositivos e as suas configurações.

### Mockups

Neste capítulo apresentamos os mockups desenhados para aplicar os requisitos definidos na secção anterior.

Uma imagem com texto, multimédia, captura de ecrã, software

Descrição gerada automaticamente

Figura 3.1 - Mockup início

Uma imagem com texto, captura de ecrã, multimédia, Software de multimédia

Descrição gerada automaticamente

Figura 3.2 - Mockup layout horizontal

Uma imagem com texto, software, Software de multimédia, multimédia

Descrição gerada automaticamente

Figura 3.3 - Mockup adicionar dispositivo



Figura 3.4 - Mockup adicionar dispositivo IP



Figura 3.5 - Mockup definições dispositivo



Figura 3.6 - Mockup emitir alerta para um dispositivo



Figura 3.7 - Mockup dispositivo expandido



Figura 3.8 - Mockup página alertas

# Implementação

Este capítulo serve para justificas as escolhas tomadas a respeito da implementação das funcionalidades descritas no capítulo anterior.

## Escolha do modelo de deteção

Sendo esta aplicação uma aplicação de deteção de objetos e alerta, o primeiro passo é escolher o modelo a utilizar para a deteção dentro dos muitos existentes nos dias de hoje e apresentados no capítulo 2.3. Para isso vamos analisar resultados de alguns estudos comparativos de modelos de deteção de objetos.

A seguinte tabela foi retirada de uma pesquisa publicada em 2022 por Syed Sahil Abbas Zaidi e permite comparar a eficácia de vários modelos atuais destacando a precisão através dos parâmetros AP[0,5:0,95] e AP0.5 e a velocidade através do parâmetro FPS. AP significa Average Precision e quando seguida de 0.5 representa a precisão média de todas as classes quando preveem bounding boxes com IoU > 0.5 sendo uma métrica menos rigorosa, enquanto quando seguida de [0,5:0,95] avalia a média da precisão de múltiplos valores de IoU (de 0.5 a 0.95), fornecendo uma visão mais completa e rigorosa do desempenho do modelo. FPS representa o número de frames processados por segundo e avalia a velocidade do modelo. As linhas a cinzento identificam modelos com FPS > 30.

Uma imagem com texto, captura de ecrã, menu, número

Descrição gerada automaticamente

Uma vez que um dos requisitos para a nossa aplicação é função de real-time, focamo-nos apenas nas linhas a cinzento uma vez que apenas estas são indicadas para real-time atingindo pelo menos 30 FPS. Dentro destas, a que apresenta maior precisão tanto AP[0,5:0,95] como AP0.5 é o modelo YOLO-v4 que permite capturar 31 fps e mantém um AP0.5 de 64,90% e AP[0,5:0,95] de 43%. O modelo EfficientDet-D2 apresenta também AP[0,5:0,95] de 43% conseguindo 41.7 fps.

Num estudo mais recente (2024) conduzido por Gaudenz Boesch está presente o seguinte gráfico.



Figura 4.1 - Gráfico comparação YOLOv7 vs. YOLOv5 vs. YOLOR and Vit Transformers, fonte: https://viso.ai/deep-learning/object-detection/

Neste gráfico no eixo do X verifica-se o tempo de inferência que quanto menor, melhor e no eixo do Y a precisão média. Aqui verifica-se que a versão mais recente do YOLO apresentada (YOLOv7) é claramente favorita atingindo 56,8% de precisão, a maior de todos os modelos apresentados e é a que apresenta menor tempo de inferência, apenas igualando outras versões do YOLO (YOLOv5 e YOLOR) que atingem uma precisão menor.

Esta segunda comparação mostra que o modelo antes equiparado ao YOLO, o EfficientDet-D2, perde significativamente quando comparado a versões mais recentes do YOLO.

O YOLOv8 foi lançado em janeiro de 2023 e mostra melhorias relativamente ao seu antecessor (YOLOv7) tanto na precisão como na velocidade como se pode observar nos seguintes gráficos publicados pelo criador do modelo.

## YOLO algorithm comparison YOLOv7 vs YOLOv8

Figura 4.2 - Gráfico comparação YOlOv7 e YOLOv8, fonte: https://docs.ultralytics.com/models/yolov8/

Uma vez que o modelo YOLOv7 mostrou-se o favorito perante as necessidades da nossa aplicação e o modelo YOLOv8 é mais preciso e mais rápido que o anterior, vamos implementar o modelo YOLOv8 como o nosso modelo de deteção de objetos integrado na aplicação.

## Ambiente de desenvolvimento

Explicar Python e PyQT

## Arquitetura da aplicação

Este projeto pode ser dividido em duas grandes componentes, o ficheiro “yoloScript.py” que contém a chamada à função do YOLO e toda a lógica de deteção de objetos e compreende também os métodos relacionados com a emissão de alertas. A outra componente com o nome de “App.py“ abrange a lógica da aplicação que interage com o utilizador onde recorremos ao PyQT5.

Para a chamada à função de previsão do YOLO, enquadrada na funcionalidade deteção de objetos, escolhemos uma abordagem das quatro inicialmente consideradas: Grab & Retrieve, Parallel, Open & Close e Open & Close threading. Na secção de teste 5.2 explicamos em detalhe cada uma delas e o cenário de teste efetuado que nos permitiu escolher a melhor alternativa atendendo aos nossos requisitos. A arquitetura escolhida foi Parallel que para cada dispositivo de captura é criada uma thread que lê um frame, descodifica-o e envia para a função do YOLO para que possa detetar objetos na imagem. No mesmo teste verificamos ainda que o ideal é que esta chamada à função YOLO ocorra com intervalo mínimo de 10 segundos.

Estas duas componentes comunicam através de um elemento queue presente na biblioteca multiprocessing. Este elemento permite manter um canal aberto entre estes dois ficheiros que funciona como uma pilha manipulando itens da maneira First In First Out. A queue é criada no ficheiro App.py, enviada como parâmetro para o ficheiro yoloScript e é partilhada pelos dois permitindo a comunicação.

## Implementação dos requisitos funcionais

### RF1. Adicionar dispositivo

A função adicionar dispositivo, acedida pelo botão na página de dispositivos, cria um widget com os parâmetros definidos no painel de configuração, adiciona-o à lista de widgets global e ao dicionário de dispositivos. De seguida cria uma thread que acede ao método predict do ficheiro yoloScript que começa a capturar frames do dispositivo e processa-os com o YOLO para detetar objetos adicionados à lista de dispositivos a detetar na página de configuração. Por fim, caso este seja o primeiro dispositivo a ser adicionado, este método cria uma thread que é responsável pela leitura dos itens da queue que provém da deteção do outro ficheiro.

### RF2. Editar dispositivo

A função editar dispositivo, acedida através do botão de settings presente no widget do dispositivo, o sistema abre uma janela de configuração com os campos preenchidos com os parâmetros do dispositivo. A classe do widget depois fica à espera de que a janela ConfigurarDispositivo envie um sinal de que o utilizador terminou as alterações e altera os seus atributos para os recentes editados.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 4.3 – Código editar dispositivo

Finalmente chama-se o método “update\_obj\_to\_find” do ficheiro yoloScript para que a lista de objetos a detetar e o tempo para emitir alertas sejam atualizados também nos métodos deste ficheiro.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura 4.4 Código atualizar lista de objetos a detetar

### RF3. Pausar captura

Outro botão presente no widget de cada dispositivo é o botão de pausa. Este está associado ao método “stop\_button\_clicked” que começa por chamar o método “change\_stop” do ficheiro yoloScript que, com recurso à instrução with em python, altera o valor do dicionário stop\_dict na posição do dispositivo a ser pausado para true. A função que está continuamente a capturar frames para deteção (função predict), na próxima iteração, entra nos blocos de código contidos na condição “with stop\_lock: ” (representado nas figuras abaixo) e interrompe a captura de frames até que o stop\_dict para este dispositivo volte a ter o valor false. É utilizado um “Lock” (stop\_lock) para garantir que a operação seja thread-safe.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 4.5 - Código stop\_lock 1

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 4.6 - Código stop\_lock 2

Regressando ao ficheiro com o código da aplicação, após notificarmos a função de captura para parar de capturar, alteramos a imagem do widget para a última imagem capturada com um icon de pausa sobreposto.

### RF4. Retomar captura

Do mesmo modo que temos uma função para avisar a função predict que pare a captura, a mesma função avisa a última a remotar a captura. Para isto a basta chamar a função change\_stop na mesma, mas passando o valor false em vez de true. Com isto, o dicionário stop\_dict na posição do dispositivo readquire o valor false, deixando de entrar nos blocos de código dentro da condição “with stop\_lock: ” retomando à normal funcionamento da função.

O próximo passo é ir buscar o valor do delay na comboBox, porque este pode ter sido alterado, e chamar a função change\_delay do ficheiro yoloScript que, paralelamente ao funcionamento do stop\_dict e stop\_lock, utiliza o delay\_dict e delay\_lock para alterar a variável local delay na função predict acedendo ao valor presente no dicionário delay\_dict na posição do dipositivo.

### RF5. Capturar live

### RF6. Expandir dispositivo

### RF7. Remover dispositivo

### RF8. Alterar layout

### RF9. Alterar delay

### RF10. Adicionar objetos a detetar

### RF11. Remover objetos a detetar

### RF12. Definir tempo para alerta de objeto

### RF13. Filtrar alertas

### RF14. Eliminar alertas

### RF15. Exibir detalhes de alerta

### RF16. Gravar configuração

### RF17. Importar configuração

# Testes

No capítulo de testes vamos apresentar testes conduzidos por nós que no decorrer do desenvolvimento do projeto achamos necessários.

## Teste de margem ideal para definir posição do objeto

O YOLO, após detetar um objeto, desenha uma bounding box que delimita o objeto detetado. Mesmo que o input não se altere, no caso deste projeto em que o input são camaras de vídeo, significa objeto na mesma posição, esta box pode variar ligeiramente devido a vários fatores. Estes fatores compreendem: variabilidade de rede, anchor boxes ou Non-Maximum Supression (NMS). Variabilidade de rede porque, a rede YOLO, como qualquer rede neuronal de deep lerning, tem variabilidade inerente nas suas previsões, devido à variação de valores inicias de peso, processos de otimização e descendente estocástico de gradiente (Stochastic Gradient Descent - SGD). Entende-se por Anchor boxes as formas e tamanhos predefinidos pelo algoritmo usadas para prever as bouding boxes. Se as anchor boxes não corresponderem exatamente à forma e tamanho dos objetos na imagem, pode também levar a variações nas bounding boxes. Por fim, NMS é uma técnica de pós-processamento usada pelo YOLO para filtrar as bounding boxes previstas. Neste processo é escolhida a box com maior grau de confiança e suprime-se as boxes sobrepostas com menor grau de confiança. O limite, isto é, o valor de confiança mínimo necessário para que uma bounding box seja considerada válida, usado nesta técnica pode afetar que bounding boxes são selecionadas, resulta em variações nas previsões finais.

Posto isto, submetemo-nos à criação de um cenário de testes para definir qual a margem ideal a definir para compensar a variação da posição da bounding box em objetos parados. Deste modo garantimos uma deteção de movimento mais rigorosa, e consequentemente alertas de segurança mais precisos e confiáveis.

### Condições do teste

Para avaliar qual a margem que confere a melhor exatidão dos resultados, foram simulados 17 cenários de testes variando os objetos a detetar e a distância a que se encontram do dispositivo de gravação. A posição da camara manteve-se a mesma durante todo o procedimento bem como o cenário onde se encontraram os objetos a detetar. Este cenário consistia num dia de sol, céu limpo com presença de sombra onde os objetos foram detetados.

Os cenários de teste foram os seguintes:

* Mochila (backpack) a 5,10 e 15 metros
* Bola (sports ball) a 5,10 e 12 metros
* Pessoa (person) a 5,10,15,20,25,30,35,40,50,60 e 70 metros

### Resultados obtidos:

Os gráficos seguintes representam, à esquerda, a distância entre os cantos inferior esquerdo (canto 1) do frame anterior e o atual e a distância do canto superior direito (canto 2) entre o frame anterior e atual à direita.

### Class “backpack”

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.1 - Gráfico cantos backpack a 5M

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.2 - Gráfico cantos backpack a 10M

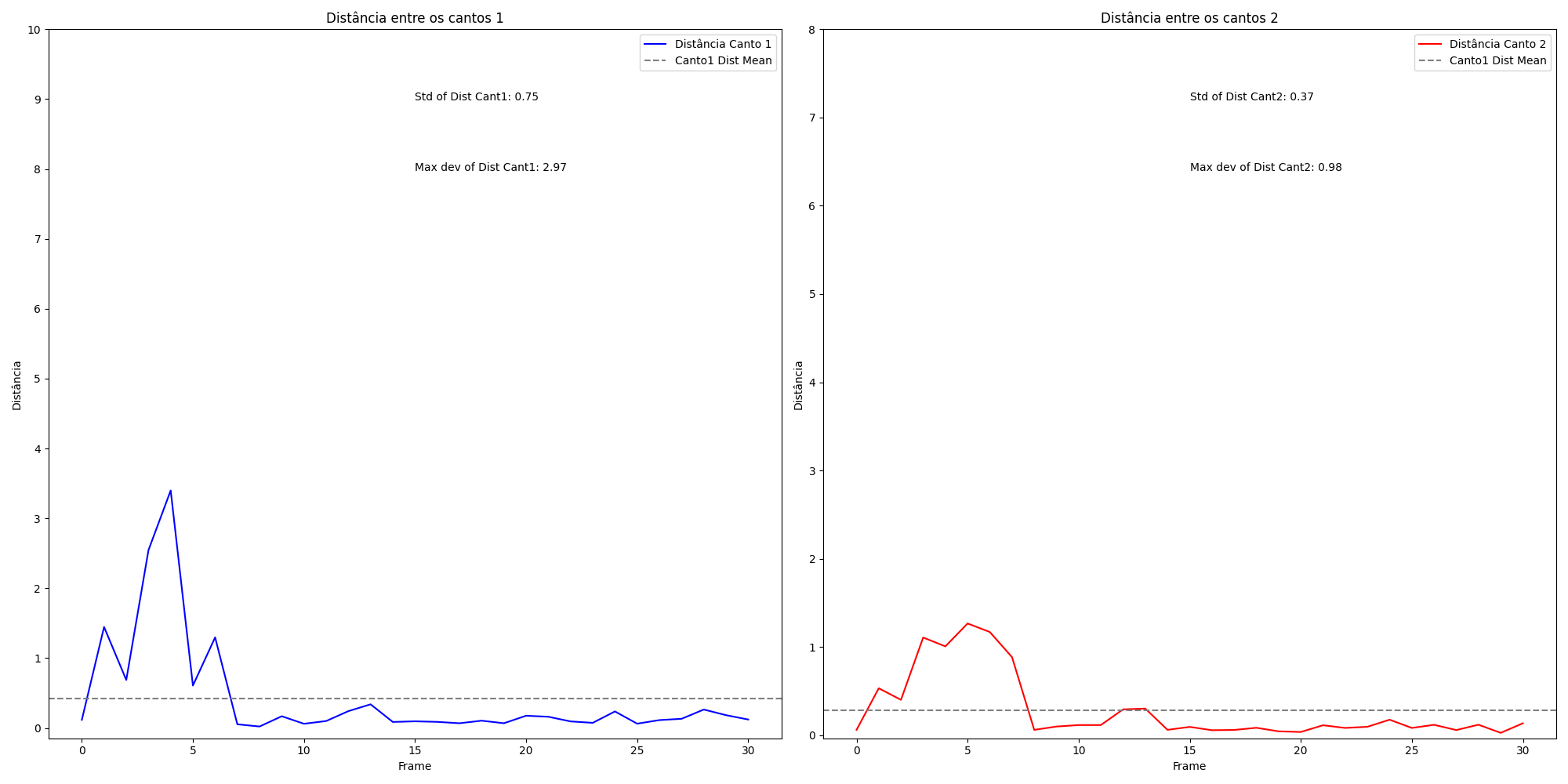


Figura 5.3 - Gráfico cantos backpack a 15M

### Class “sports ball”

Uma imagem com diagrama, file, texto, Gráfico

Descrição gerada automaticamente

Figura 5.4 - Gráfico cantos bola a 5M

Uma imagem com texto, file, Gráfico, diagrama

Descrição gerada automaticamente

Figura 5.5 - Gráfico cantos bola a 10M

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.6 - Gráfico cantos bola a 12M

### Class “Person”

Uma imagem com file, texto, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.7 - Gráfico cantos person a 5M

Uma imagem com file, diagrama, texto, Gráfico

Descrição gerada automaticamente

Figura 5.8 - Gráfico cantos person a 35M

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.9 - Gráfico cantos person a 70M

### Análise dos resultados

Após teste dos vários cenários, seguidos da análise e discussão dos resultados obtidos, consideramos 4 px como margem ideal a aplicar para desprezar as possíveis variações das bounding boxes geradas pelo modelo de deteção usado em objetos parados. Isto deve-se ao facto dos maiores desvios obtidos serem na ordem dos 3px, sendo o máximo deles 3.15 no caso de teste backpack a 5 metros. Sendo 4 o próximo inteiro, 4px cobre todos os desvios que obtivemos na fase de testes.

## Teste da arquitetura para recolher frames de dispositivos

Serve o presente teste para testar qual a arquitetura que melhor se adequa à nossa aplicação e aos seus requisitos. Para isso temos 4 modelos para teste sendo eles:

### Arquiteturas propostas

### Grab & Retrieve

Neste modelo, após inicializarmos as câmaras com o videCapture de cada dispositivo, criamos uma thread de captura por dispositivo. Esta thread apenas agarra (grab) continuamente frames do dispositivo para posterior descodificação e análise do frame. Na thread principal, de X em X segundos (definido pelo utilizador) chamamos a função retrieveFrames que percorre o vetor das câmaras previamente inicializadas e recolhe (retrieve) o último frame agarrado (grabbed) pela thread de captura do dispositivo associado descodificando-o. De seguida enviamos cada frame para a função YOLO para o processar.

### Parallel

Nesta arquitetura, para cada dispositivo de captura selecionado, é criada uma thread que lê, descodifica e envia o frame ao YOLO para detetar objetos na imagem.

### Open & Close

Neste padrão (ausente de threads) cada dispositivo de captura é inicializado, captura o frame, processa-o para deteção através do YOLO e fecha o dispositivo, seguindo para o próximo continuamente, N vezes, sendo N o número de dispositivos selecionado.

### Open & Close threading

Neste modelo, inicializa-se os dispositivos e cria-se uma thread por dispositivo para ler um frame. Após todas as threads terem lido um frame, cria-se uma thread por frame para o processar no YOLO. Finalmente, depois das threads terminarem o processamento, fecham-se os dispositivos inicializados no início.

### Condições de teste

Para comparar os vários modelos de implementação, utilizamos a biblioteca “psutil” do python que nos permite avaliar a % de CPU usage bem como a % de RAM usage durante todo o processo. Para avaliar os cenários, utilizamos as mesmas 5 câmaras, das quais 2 locais e 3 por IP e definimos 40 iterações para avaliar o tempo de execução.

### Resultados obtidos

### Grab & Retrieve

Neste padrão não foi possível concluir os testes porque ao usar câmaras IP em vez de locais no videoCapture, o retrieve não consegue descodificar (retrieve) o frame agarrado (grabbed) anteriormente, pelo que excluímos das opções.

### Parallel

Uma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamente

Figura 5.10 - Gráfico performance parallel

Ao analisar a percentagem elevada de CPU usage nesta arquitetura, acrescentamos dois novos casos de teste para tentar diminuir a % de CPU usage sacrificando tempo de execução. Para isso, adicionamos um time.sleep de 5 e 10 segundos.

### Uma imagem com texto, file, captura de ecrã, diagrama Descrição gerada automaticamenteParallel 5 sec

Figura 5.11 - Gráfico performance parallel 5 segundos

### Uma imagem com texto, file, diagrama, Gráfico Descrição gerada automaticamenteParallel 10 sec

Figura 5.12 - Gráfico performance parallel 10 segundos

### Uma imagem com texto, file, Gráfico, diagrama Descrição gerada automaticamenteOpen & Close

Figura 5.13 - Gráfico performance open & close

### Open & Close threading

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.14 - Gráfico performance open & close threading

### Análise dos Resultados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | Tempo execução | Máx – Init CPU % | Máx – Init RAM % | Mean – Init CPU % | Mean – Init RAM % |
| Parallel | 1m 44s | 100 | 10,80 | 85,88 | 10,03 |
| Parallel 5 sec | 5m 24s | 100 | 8,50 | 34,55 | 7,67 |
| Parallel 10 sec | 7m 39s | 35,20 | 8,30 | 20,46 | 4,11 |
| Open & Close | 17m 58s | 28,30 | 19,50 | 9,86 | 13,53 |
| Open & Close threading | 10m 25s | 32,40 | 13 | 18,14 | 8,77 |

Através dos dados retirados podemos criar a seguinte tabela:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Menor tempo execução | Maior tempo  Execução | Menor CPU usage | Maior CPU usage | Menor RAM usage | Maior RAM usage |
| Parallel | Open & Close | Open & Close | Parallel | Parallel 10 sec | Open & Close threading |

Neste projeto, priorizamos o menor custo computacional em relação ao tempo de execução. Dito esto, excluímos o modelo Parallel pois utiliza o esforço máximo do CPU e o Open & Close threading por representar o maior esforço sobre a RAM.

Entre o Parallel 5 sec e o Parallel 10 sec, o último representa o melhor comparativamente porque demora 1,42 vezes mais para concluir o teste mas utiliza menos 1,69 vezes o CPU e 1,87 vezes menos a RAM.

O Open & Close é o que apresenta menor custo para o processador (9,86%) mas demora demasiado tempo em relação ao Parallel 10 sec que utiliza 20,46% do processador, o que representa 2,08 vezes mais que o anterior, mas acaba o teste 2,35 vezes mais cedo, o que representa um rácio maior de tempo / CPU usage, pelo que também exclui o Open & Close em comparação ao Parallel 10 sec.

### Conclusão

Após análise e discussão dos resultados obtidos, escolhemos a arquitetura Parallel 10 sec como a mais adequada para a nossa aplicação por ser o mais equilibrado quando aplicamos os nossos critérios.

# Conclusões ou Conclusão

Inserir aqui as conclusões ou conclusão. Trata-se de um elemento **obrigatório**.

A conclusão:

* Deve ser sucinta;
* Não deve conter informações ou ideias novas;
* Deve permitir concluir se se atingiram os objetivos enunciados na introdução.

Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão.

Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão.

# Bibliografia ou Referências Bibliográficas

Inserir aqui a bibliografia ou referências bibliográficas. Trata-se de um elemento **obrigatório**.

Notas: o sistema a adotar para a apresentação das referências bibliográficas e as suas citações deve:

* Respeitar uma norma estabelecida;
* Seguir as práticas mais disseminadas na área em causa;
* Ser empregue de modo uniforme em todo o documento.

Bibliografia – quando se coloca toda a bibliografia consultada;

Referências bibliográficas – quando se faz referência apenas à bibliografia citada.

# Anexos

Elemento a figurar, **quando aplicável**.

# Glossário

Elemento a figurar, **quando aplicável**.