|  |
| --- |
| **UNIVERSIDADE AUTÓNOMA DE LISBOA**  **LUÍS DE CAMÕES** |
| **DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS**  **LICENCIATURA EM ENGENHARIA INFORMÁTICA** |
|  |
| **RECONHECIMENTO/CLASSIFICAÇÃO AUTOMÁTICA DE ALVOS POR ALGORITMOS DE DEEP LEARNING EM OPENCV ATRAVÉS DE INTERFACE WEB** |
| Relatório de Projeto para obtenção do grau de Licenciado em Engenharia Informática |
|  |
| Autor/a: Bruno Miguel Almeida Silva,  Diogo Miguel da Silva Mendes,  Reinaldo João Ramos Coelho,  Ricardo Manuel dos Santos Cardoso  Orientador/a: Dr. Mário Marques da Silva  Número do/a candidato/a: 30003696, 30003865, 30003670, 30002299 |
| **julho de 2021**  **Lisboa** |

(Página em Branco)

# Dedicatória

Bruno Silva: “Obrigado à minha família e aos meus amigos. Dedico este projeto a eles.”

Diogo Mendes: “Quero dedicar este projeto, que representa toda uma jornada pela Licenciatura em Engenharia Informática aqui na UAL, aos meus pais pelo apoio que me deram e que permitiram que a minha viagem sinuosa pelo ensino superior finalmente chegasse ao fim.

Também quero agradecer ao grupo de colegas que me acompanhou ao longo destes 3 anos. Um grupo que sempre trabalhou e se ajudou mutuamente para que todos atingíssemos os nossos objetivos. A todos vós, obrigado pelo vosso apoio dentro e fora da sala de aula.”

Reinaldo Coelho: “Em primeiro lugar, quero dedicar este projeto final de curso da licenciatura em Engenharia Informática da Universidade Autónoma de Lisboa à minha família e amigos, em especial, aos meus pais e à minha irmã, por todo o suporte que me deram ao longo destes 22 anos de vida e estes 3 anos que estou na Universidade, um grande obrigado, do fundo do coração. Sem eles, não tinha chegado a este patamar que estou hoje a atingir. Dedico também este trabalho aos meus colegas Bruno Silva, Diogo Mendes e Ricardo Cardoso que estiveram comigo na realização do mesmo e a amizades que criei durante estes 3 anos de Universidade que estiveram a realizar outros projetos finais de curso e, se algum dia, estiverem a ler isto, saberão bem a quem estou a referir, um grande obrigado, ajudaram-me bastante ao longo deste tempo e a arranjar forças para continuar os estudos até ao fim. Sem eles, se calhar não estava aqui hoje a terminar o curso, mais uma vez, um muito obrigado por esta caminhada difícil, mas recompensadora e pelos bons momentos que passei com vocês. Por fim, quero dedicar este trabalho à minha avó paterna, que, entretanto, falecera no passado dia 10 de maio, a qual agradeço por tudo que fez por mim e esteja a descansar a paz e a olhar por todos nós e com certeza que ficaria muito feliz e orgulhosa de ver o seu neto a terminar o Ensino Superior.”

Ricardo Cardoso: “Obrigado Sofia.”

# Agradecimentos

Em nome do grupo, queremos agradecer, em primeiro lugar, às nossas famílias pelo suporte que nos deram ao longo da nossa estadia no Ensino Superior.

Agradecemos também à UAL por nos ter possibilitado entrar no curso de Engenharia Informática e darmos uma oportunidade promissora para o nosso futuro, depois da Universidade, na entrada para o mercado de trabalho.

Um especial obrigado ao nosso orientador, o professor Mário Marques da Silva, por estar sempre disponível para nos ajudar ao longo do semestre, quando tínhamos alguma dúvida ou sugestão sobre o projeto, pela sua colaboração nas reuniões que ocorreram com a Leitek, sobre a discussão que se irá realizar ou até sobre este relatório agora escrito.

Um agradecimento à LEITEK INNOVATIVE SOLUTIONS por nos ter esclarecido as nossas dúvidas nas acesas discussões nas reuniões e pelo fornecimento de ferramentas para a conclusão da realização do projeto. Em especial, para o Engenheiro Artur Pedro Morais, que esteve sempre presente nas reuniões que ocorreram, de forma remota. Ao programador, Tiago Mendes, que demonstrou sempre disponível para ajudar-nos no código, ainda que não tivéssemos assim grandes dúvidas na realização do código para o programa. Não esquecemos da Rita Capela, que apesar de ter aparecido poucas vezes nestas reuniões, também se mostrou disponível para ajudar.

Queremos também agradecer aos amigos que fizemos nesta estadia na UAL e que vão estar sempre nos nossos corações e que tudo lhes corra bem no futuro.

Agradecemos, por fim, a todos os professores que tivemos ao longo destes três anos desta jornada difícil, um grande obrigado por todos os conhecimentos que nos passaram ao longo das várias Unidades Curriculares que, assim, permitiu estarmos aqui a terminar o curso.

# Epígrafe

“Design não é apenas o que parece e o que se sente. Design é como funciona.” – Steve Jobs.

“Não podemos resolver os nossos problemas com o mesmo pensamento que tínhamos quando os criamos.” – Albert Einstein.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.” – Arthur Schopenhauer.

“O conhecimento faz-nos responsáveis.” – Che Guevara.

“Só um sentido de invenção e uma necessidade intensa de criar levam o homem a revoltar-se, a descobrir e a descobrir-se com lucidez.” – Pablo Picasso.

“Quanto mais aumenta o nosso conhecimento, mais evidente fica a nossa ignorância.” – John F. Kennedy.

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes.” – Martin Luther King.

# 

# Resumo (Português)

Este projeto consistia em Reconhecer e Classificar, de forma automática, alvos através de um algoritmo de Deep Learning em OpenCV através de uma Interface Web. Pretendia-se recorrer aos serviços do Microsoft Azure para permitir um processamento rápido via cloud e adaptar os módulos de processamento de imagem para a identificação dos objetos, no qual foi escolhido o YOLOv3/Darknet para este efeito. Numa segunda fase, pretendia-se construir uma página web com recurso ao Azure que permitisse um acesso de um streaming de vídeo ou de imagem através do seu upload, apresentar os resultados do reconhecimento e classificação na página web, no próprio feed do vídeo ou da imagem e no final, guardar os dados numa base de dados para consultas posteriores. Foi utilizado CUDA, da NVIDIA, para melhorar a performance da deteção do algoritmo, que permite usar uma GPU para este efeito invés de uma CPU.

Estes objetivos requeridos foram na sua maioria alcançados com sucesso com a exceção da parte que envolve o Azure, devido a alguns problemas alheios a nós, o que nos obrigou a ter que utilizar as nossas máquinas físicas.

Através da utilização das nossas máquinas físicas, o uso do algoritmo do YOLOv3, CUDA, uma base de dados PostgreSQL e código escrito em Python, SQL e HTML como linguagens principais, beneficiando de algumas ferramentas para a construção do mesmo como Flask, os resultados que obtivemos foram bastante positivos e vão em linha para aquilo que era pedido no enunciado.

O projeto foi proposto pela LEITEK INNOVATIVE SOLUTIONS em parceria com a Universidade Autónoma de Lisboa, que fora realizado pelos alunos Bruno Silva, Diogo Mendes, Reinaldo Coelho e Ricardo Cardoso, sob o orientador o professor Mário Marques da Silva durante a Unidade Curricular de Laboratório de Projeto.

**Palavras-chave:** Deep Learning; OpenCV; Azure; PostgreSQL; YOLOv3; Flask.

# 

# Abstract (Inglês)

This project consisted of Recognizing and Classifying, automatically, targets through a Deep Learning algorithm in OpenCV through a Web Interface. It was intended to use Microsoft Azure services to allow fast processing via the Cloud and adapt the processing modules of image for object identification, in which YOLOv3/Darknet was chosen for this purpose. In a second phase, the intention was to build a web page using Azure that would allow access to a video or image streaming through its upload, presenting the results of recognition and classification on the web page, in the video feed or in the image and at the end, save the data in a database for later consultation. NVIDIA’s CUDA was used to improve the performance of the detection algorithm, which allows to use a GPU instead of a CPU for this purpose.

These required goals were mostly successfully achieved except for the part involving Azure, due to some problems we cannot controlled, which forced us to use our physical machines.

With the use of our machines, the YOLOv3 algorithm, CUDA, a PostgreSQL database and code written in Python, SQL, and HTML as main languages, benefiting from some tools like Flask, the results that we got were positive and inline for what was asked by the utterance.

The project was proposed by LEITEK INNOVATIVE SOLTUONS in partnership with Universidade Autónoma de Lisboa, which was developed by the students Bruno Silva, Diogo Mendes, Reinaldo Coelho, and Ricardo Cardoso, under the advisor Professor Mário Marques da Silva, during the subject of Laboratory of Project.

**Keywords:** Deep Learning; OpenCV; Azure; PostgreSQL; YOLOv3; Flask.

# 

# Abstrakt

Dieses Projekt bestand in der automatischen Erkennung und Klassifizierung von Zielen durch einen Deep-Learning-Algorithmus in OpenCV über ein Webinterface.Es sollte Microsoft Azure-Dienste verwenden, um eine schnelle Verarbeitung über die Cloud zu ermöglichen und die Verarbeitungsmodule des Bildes zur Objektidentifikation anzupassen, in adapt welches YOLOv3/Darknet dafür gewählt wurde. In einer zweiten Phase sollte mit Azure eine Webseite erstellt werden, die den Zugriff auf ein Video- oder Bildstreaming durch den Upload ermöglicht, die Ergebnisse der Erkennung und Klassifizierung auf der Webseite, im Videofeed oder im Bild präsentiert und Am Ende speichern Sie die Daten in einer Datenbank für eine spätere Abfrage. NVIDIAs CUDA wurde verwendet, um die Leistung des Erkennungsalgorithmus zu verbessern, der es ermöglicht, für diesen Zweck eine GPU anstelle einer CPU zu verwenden.

Diese erforderlichen Ziele wurden größtenteils erfolgreich erreicht, mit Ausnahme des Teils mit Azure, aufgrund einiger Probleme außerhalb von uns, die uns zwangen, unsere physischen Computer zu verwenden.

Durch den Einsatz unserer physischen Maschinen, die Verwendung des YOLOv3-Algorithmus, CUDA, einer PostgreSQL-Datenbank und Code, der in Python, SQL und HTML als Hauptsprachen geschrieben wurde, sowie der Vorteile einiger Tools für den Aufbau als Flask, waren die Ergebnisse, die wir erhielten, ziemlich positiv und gehen Sie online für das, was in der Erklärung gefragt wurde.

Das Projekt wurde von LEITEK INNOVATIVE SOLUTIONS in Zusammenarbeit mit der Autonomen Universität Lissabon vorgeschlagen und von den Studenten Bruno Silva, Diogo Mendes, Reinaldo Coelho und Ricardo Cardoso unter dem Berater Professor Mário Marques da Silva während der Lehrplaneinheit des Labors von Projekt.

**Schlüsselwörter:** Deep Learning; OpenCV; Azure; PostgreSQL; YOLOv3; Flask.

# Índice

[Dedicatória 3](#_Toc77277673)

[Agradecimentos 4](#_Toc77277674)

[Epígrafe 5](#_Toc77277675)

[Resumo (Português) 6](#_Toc77277676)

[Abstract (Inglês) 7](#_Toc77277677)

[Abstrakt 8](#_Toc77277678)

[Índice 9](#_Toc77277679)

[Lista de Fotografias/Ilustrações 11](#_Toc77277680)

[Lista de Abreviaturas 12](#_Toc77277681)

[Lista de Siglas e Acrónimos 12](#_Toc77277682)

[1 Introdução 13](#_Toc77277683)

[2 Entidades Envolvidas 15](#_Toc77277684)

[2.1 Universidade Autónoma de Lisboa 15](#_Toc77277685)

[2.2 LEITEK INNOVATIVE SOLUTIONS 16](#_Toc77277686)

[3 Conceitos Teóricos e Ferramentas utilizadas no projeto 17](#_Toc77277687)

[3.1 Machine Learning 17](#_Toc77277688)

[3.2 Deep Learning 18](#_Toc77277689)

[3.3 YoloV3 e Darknet 18](#_Toc77277690)

[3.4 OpenCV 19](#_Toc77277691)

[3.5 CUDA 20](#_Toc77277692)

[3.6 Microsoft Azure 20](#_Toc77277693)

[3.7 Python 21](#_Toc77277694)

[3.8 SQL 22](#_Toc77277695)

[3.9 CSS, JavaScript e HTML 22](#_Toc77277696)

[3.10 PostgreSQL 23](#_Toc77277697)

[3.11 Flask 23](#_Toc77277698)

[3.12 SQLAlchemy 24](#_Toc77277699)

[3.13 Bootstrap 24](#_Toc77277700)

[3.14 Ubuntu 24](#_Toc77277701)

[3.15 TeamViewer 25](#_Toc77277702)

[4 Fases de Desenvolvimento do Projeto 26](#_Toc77277703)

[4.1 Primeira Fase: Protótipo e Planeamento 26](#_Toc77277704)

[4.2 Segunda Fase: O CUDA 30](#_Toc77277705)

[4.3 Terceira Fase: A Base de Dados 33](#_Toc77277706)

[4.4 Quarta Fase – A Galeria 37](#_Toc77277707)

[4.5 Quinta Fase – A reta final 38](#_Toc77277708)

[5 Funcionamento do Código da VidereApp 40](#_Toc77277709)

[6 Conclusões 45](#_Toc77277710)

[7 Trabalho futuro 47](#_Toc77277711)

[Bibliografia 49](#_Toc77277712)

[Créditos 52](#_Toc77277713)

[Anexo 01 – Código do projeto 53](#_Toc77277714)

# Lista de Fotografias/Ilustrações

[Figura 1 - Rosto de Ricardo Cardoso a ser reconhecido pelo algoritmo com uma eficácia de 52% usando o YOLOTiny. 28](#_Toc77277577)

[Figura 2 – Rosto de Diogo Mendes reconhecido pelo algoritmo com uma eficácia de 100% usando o YOLO. 28](#_Toc77277578)

[Figura 3 – Rosto de Reinaldo Coelho reconhecido pelo algoritmo usando o YoloTiny. Aqui verifica-se que o algoritmo ainda não consegue reconhecer uma caixa de um videojogo. 29](#_Toc77277579)

[Figura 4 – Uma nova experiência de Reianldo Coelho com outra caixa de videojogo mostra alguma dificuldade do algoritmo em detetar corretamente pessoas em uma caixa. Pode-se ver que o algoritmo consegue reconhecer como uma pessoa, o jogador Cristiano Ronaldo, já o jogador Marcus Rashford, o algoritmo reconhece-o como uma garrafa. 29](#_Toc77277580)

[Figura 5 – Quarto de Bruno Silba, onde o algoritmo reconhece uma televisão, cadeira e uma garrafa com uma eficácia acima de 90%, mostrando que o algoritmo consegue identificar mais objetos além de pessoas utilizando o YOLO. 30](#_Toc77277581)

[Figura 6 – Máquina Virtual do Azure com GPUs que usámos para o CUDA. 32](#_Toc77277582)

[Figura 7 – Especificações da VM utilizada no Azure, com SO Ubuntu 18.04, CPU Intel Xeon Platinum 8171M com 4 núcleos de velocidade 2,095GHz, 16GB de RAM. 32](#_Toc77277583)

[Figura 8 – CUDA compilado com sucesso no Azure, através do nvcc. 32](#_Toc77277584)

[Figura 9 – Compilação do OpenCV com o CUDA na máquina do Azure. 33](#_Toc77277585)

[Figura 10 - Teste de Bruno Silva realizado com sucesso sobre deteção de duas câmaras em simultâneo, apresenta em ambos uma eficácia de 94% usando o YOLO com CUDA na máquina local. 33](#_Toc77277586)

[Figura 11 - Desenho da Base de Dados realizada, contém as classes frames, object, objects\_found, objects\_video, vídeos, stream\_urls e utilizadores 34](#_Toc77277587)

[Figura 12 - Protótipo da Interface Web, página das câmaras 36](#_Toc77277588)

[Figura 13 - Protótipo da Interface Web, deteção de pessoas pelo YOLO através de upload de vídeo. A página também apresenta uma barra para o brilho e para o contraste. 36](#_Toc77277589)

[Figura 14 – Protótipo da Interface da Galeria. 37](#_Toc77277590)

[Figura 15 – Barre de progresso do processamento de um vídeo no site. 38](#_Toc77277591)

[Figura 16 – Página da Galeria contendo Vídeos e Imagens em separado. Em ambos, podemos ver os resultados da deteção com bastante eficácia. 39](#_Toc77277592)

# Lista de Abreviaturas

|  |  |
| --- | --- |
| ID | Identificador |
| Leitek | LEITEK INNOVATIVE SOLUTIONS |
| ML | Machine Learning |
|  |  |

# Lista de Siglas e Acrónimos

|  |  |
| --- | --- |
| ACID | Atomicidade, Consistência, Isolamento e Durabilidade |
| API | Application Programing Interface/Interface de Programação de Aplicação |
| CSS | Cascading Style Sheet |
| CPU | Central Processing Unit/Processador |
| CUDA | Compute United Device Architecture |
| cuDNN  DOM  IA/AI  GB  GHz  GPGPU  GPU  HTML  OpenCV  RAM  RGB  SQL  SO/OS  PC  UAL  UI  URL  UTC  UX  VM/MV  YOLO | CUDA Deep Neural Network  Document Object Model  Inteligência Artifical  GigaByte  GigaHertz  General Purpose Processing computing ou Graphics Processing Units  Graphics Processing Units/Placa Gráfica  HyperText Markup Language  Open Source Computer Vision Library  Random Access Memory  Red, Green, Blue  Structured Query Language  Sistema Operativo  Personal Computer/Computador Pessoal  Universidade Autónoma de Lisboa  User Interface  Uniform Resource Locator  Coordinated Universal Time  User Experience  Máquina Virtual  You Only Look Once |

# Introdução

Para definirmos o que é Inteligência Artificial, temos de começar pelo que significa inteligência, o que não é uma tarefa fácil. Inteligência é um termo demasiado amplo e vago por si só, às vezes envenenado por motivos nefastos ou pela falta de introspeção no que realmente significa inteligência. Essencialmente, toda a gente tem uma noção vaga do que é inteligência, mas quando se trata de definir a palavra, as noções irão divergir para vários temas: lógica, compreensão, mestria, resolução de problemas, raciocínio, dedução, indução, intuição e até mesmo instinto e emoção.

Para nós, que programamos máquinas, não estamos interessados na maioria desses termos. Queremos ensinar apenas um software a resolver um problema sozinho, com o menor input possível do utilizador. Inteligência Artificial no contexto deste trabalho, trata-se de algoritmos e cadeias de código que evoluem por si só, naturalmente e exponencialmente, treinando e ficando cada vez mais eficazes e eficientes.

Em relação a este projeto, temos o interesse de fazer uma máquina aprender a identificar certos objetos, denominados de alvos na caracterização do projeto, numa stream de vídeo. Entretanto a lógica é semelhante a qualquer outro modelo de inteligência artificial: processamento, treino, e aprendizagem, sem qualquer influência do utilizador no processo. Assim, é importante distinguir a diferença entre a utilização de inteligência artificial e de uma abordagem de programação clássica, em que Inteligência Artificial responde e processa qualquer input a partir do código que foi desenvolvido no início, sem qualquer modificação do programador.

Partindo deste conceito de Inteligência Artifical no contexto deste projeto, o nosso objetivo, proposto pela UAL e pela parceira da Universidade, a LEITEK INNOVATIVE SOLUTIONS, é, conforme está no título deste relatório, Reconhecimento/Classificação automática de alvos por algoritmos de Deep Learning em OpenCV através de Interface Web. O projeto visa a utilização de algoritmos através de técnicas de Deep Learning para classificar e reconhecer vários objetos, em tempo real, numa interface web, onde nesta, haja a possibilidade de processamento via Cloud, que detete e classifique de forma aplicada, a partir de uma fonte ou várias fontes via streaming de vídeo ou via upload de imagens para a interface web. O OpenCV será utilizado no sentido de tirar proveito dos módulos de processamento de imagem, em específico, no reconhecimento de objetos em tempo real. Para o processamento via Cloud, vai ser utilizado o Microsoft Azure, e a interface web será construída numa Máquina Virtual. De uma forma simples, o programa consiste em configurar uma Máquina Virtual que possibilite uma computação adequada e ajustada aos resultados que se pretendem, seguindo-se de seguida, um acesso a um streaming de vídeo ou upload de uma imagem, nessa fonte de vídeo ou de imagem, apresentar os resultados do reconhecimento/classificação na página web, e, por fim, armazenar os dados dos resultados numa base de dados PostgreSQL para consulta posterior. Espera-se também que o sistema seja capaz de efetuar o reconhecimento de vários objetos e avaliar a taxa de falsos positivos.

Ao longo do relatório serão enunciados alguns pontos fundamentais: O ponto 2 refere-se às Entidades Envolvidas no projeto realizado, a UAL e a parceira, Leitek. Neste ponto fala-se um pouco sobre cada uma das entidades e os objetivos. O ponto 3 é dedicado a conceitos teóricos, servindo um pouco de introdução para o tema do projeto e as ferramentas que utilizamos ao longo do desenvolvimento do projeto. O ponto 4, é a secção mais longa deste relatório onde falamos sobre os vários passos do desenvolvimento do projeto e um pouco sobre a sua execução. O ponto 5 é dedicado a uma pequena explicação do funcionamento do código realizado no projeto, o ponto 6 é referente às conclusões tiradas do projeto e o ponto 7 ao eventual trabalho futuro que possa ocorrer. Por fim, segue-se uma secção dedicada à Bibliografia e o código escrito, sob a forma de anexo.

Ao longo deste documento e durante o desenvolvimento do projeto até ao seu término, é referido várias vezes o nome Videre. Videre significa “Ver” em latim, onde contexto, ver é um olho que, de certa forma, simboliza uma câmara, que é um dos principais objetivos da realização do projeto, deteção de alvos através de câmaras onde é guardado os resultados em uma base de dados. Este nome também foi escolhido de forma a simplificar o nome do projeto, que consideramos que era bastante extenso.

# Entidades Envolvidas

No ponto a seguir, falaremos sobre as entidades que estiveram envolvidas connosco no projeto, a UAL e a LEITEK INNOVATIVE SOLUTIONS.

## Universidade Autónoma de Lisboa

A Universidade Autónoma de Lisboa (UAL) foi inaugurada a 13 de dezembro de 1985, é a universidade privada mais antiga do país, com mais de 30 anos de experiência ao serviço da comunidade académica e estudantil. Com um corpo docente de excelência e uma vasta oferta formativa composta por dezenas de licenciaturas, mestrados, pós-graduações e doutoramentos [2] em áreas como Arquitetura, Ciências da Comunicação, Ciências e Tecnologias, Ciências Económicas e Empresariais, Direito, História, Artes e Humanidades, Psicologia ou Relações Internacionais [1]. Está localizada no Palácio dos Condes do Redondo, em Lisboa. A UAL pertence ao Grupo CEU, também fundada em 1985, uma marca que agrega diferentes estabelecimentos de ensino superior, universitário e politécnico, assim como de ensino especializado e artístico. Entre os estabelecimentos de ensino que o Grupo CEU tem, para além da UAL, a Escola de Pós-Graduações Autónoma Academy ou a Escola Superior de Enfermagem São Francisco das Misericórdias (ESESFM) [3].

A UAL contém no seu estabelecimento de ensino, serviços como bibliotecas, bares, alojamento para estudantes, cantinas, seguro escolar, bolsas de mérito, apoios a estágios, entre vários outros.

A UAL, apresenta como missões, enquanto instituição de ensino superior a qualificação de alto nível dos seus estudantes, a sua formação cultural, artística, tecnológica e científica, num quadro de referência nacional e internacional. A UAL pretender valorizar a atividade dos seus investigadores, docentes, estudantes e trabalhadores, estimulando a sua formação intelectual e profissional. Promove e organiza ações de apoio e difusão da cultura humanística, artística, científica e tecnológica, com vista á sua compreensão pública. Persegue o enriquecimento humano e social da sua comunidade, dentro de um quadro de valores humanistas, personalistas, de tolerância, de cooperação interna e internacional e de respeito pelos direitos do Homem. No quadro da legalidade instituída, a UAL rege-se pelos princípios da autonomia académica, cultural, científica e pedagógica e da participação de docentes e discentes na sua gestão académica [2].

## LEITEK INNOVATIVE SOLUTIONS

A LEITEK INNOVATIVE SOLUTIONS é uma empresa privada, pertencente ao ecossistema de empresa do setor de Base Industrial e Tecnológica da Defesa (IDD), que responde às necessidades de inovação dos seus clientes, fornecendo e implementando soluções complexas e de estado de arte, nas áreas de deteção precoce e simulação de incêndios florestais, aeronaves não tripuladas, segurança e serviços de mapeamento com aeronaves não tripuladas [4].

Em todas as áreas de atividade a LEITEK faz uso extensivo de Inteligência Artificial e Machine Learning, quer nos sistemas que comercializa, quer nos sistemas que são produto de desenvolvimento próprio, como, por exemplo o sistema Dronesavior e ScanCrowd [4].

A LEITEK desenvolve muita das suas atividades na investigação e desenvolvimento, comercialização e prestação de serviço aéreo com aeronaves não tripuladas fazendo parte de organizações de referências como o DRONE REGIM da UVS International e Associação Portuguesa de Aeronaves Não Tripuladas, de sua sigla APANT. A empresa é parceira e distribuidora exclusiva da INSIGHT ROBOTICS / ROBOTICS CATS para países como Portugal, Espanha e outros do Sul da Europa, onde presta suporte técnico especializado nesta área [4].

A empresa está focada na comercialização e desenvolvimento de soluções em setores de atividade com bastante potencial de crescimento aliada às oportunidades de negócio trazidas pela acelerada transformação digital no planeta e pelo avanço tecnológico, assim como, na prestação de serviços com Aeronaves não Tripuladas e processamento de dados com analítica avançada [5].

A LEITEK tem como missão proporcionar a entidades públicas e clientes privados as melhores soluções inovadoras e agregar valor em todos os projetos nas áreas tecnológicas de: Vídeo Vigilância da Floresta; Deteção Precoce de Incêndio Florestal; Aeronaves não tripuladas e tecnologias associadas; Defesa e Segurança; Segurança operacional em aeroportos e Prestação de Serviços de Aeronaves Não Tripuladas [5]. Tem como visão futura ser uma empresa de referência e ser fornecedor líder em Portugal e no Sul da Europa, nas áreas em específico, da Deteção de Incêndios Florestais e Serviços de Aeronaves Não Tripuladas, no sentido de contribuir para um mundo em transformação, mais seguro e sustentável [5].

# Conceitos Teóricos e Ferramentas utilizadas no projeto

No capítulo que se segue, vamos falar de dois conceitos teóricos que achámos fundamentais para a realização do projeto: Machine Learning e Deep Learning, além das ferramentas que utilizámos ao longo do desenvolvimento do projeto. Chegou-se a utilizar algumas ferramentas de forma indireta, como jQuery, JSON ou Popper. Porém, como foram utilizadas apenas de forma indireta não vão ser referidas nos pontos a seguir, já que serviu apenas para suporte para as outras ferramentas principais.

## Machine Learning

Machine Learning, informalmente conhecido como ML, é um tipo de inteligência artificial que permite que aplicações de software possam ser mais precisas e eficazes no processamento e na previsão de resultados, apesar de nenhum código específico tenha sido escrito para processar os dados. Isto significa que em vez de adaptarmos o código a cada um dos cenários possíveis para controlar o fluxo do processamento, simplesmente damos um objetivo ao programa que será o guia para ver se as interações estão mais perto do objetivo ou não.

Em termos técnicos, ML nada é mais que a interação de algoritmos que criam de forma automática modelos de representação de conhecimento com base num conjunto de dados. Sendo estes modelos de representação o que o algoritmo já aprendeu e melhorou por si mesmo.

E antes de poder ser usado, um algoritmo é deixado a processar um conjunto de dados de treino para afinar a sua precisão. Após o treino, o modelo tem potencial para efetuar previsões de qualidade em situações posteriores e que estejam relacionadas com padrões anteriores. Basicamente informação nova, mas do mesmo tipo que a que foi usada para treino. Seja texto, rostos humanos ou animais, um dos exemplos que pode ser um tipo de dado analisado por algoritmos ML.

No entanto, ML está limitado na sua única função e apenas faz uma coisa bem. Isto significa que para outras funções mais complexas, ML simplesmente não chega. Precisamos de algo mais complexo.

Estes métodos de análise de dados estão bastante enraizados no nosso dia-a-dia, alguns usos de Machine Learning, são, por exemplo, as pesquisas na Web, as recomendações da Amazon ou o Google Tradutor [16].

## Deep Learning

Deep Learning é um tipo de machine learning que permite treinar PCs para realizar tarefas como se atuasse como um ser humano, permite o reconhecimento de imagem, previsões e reconhecimento da fala. Ao contrário de alguns algoritmos ML, que organizam os dados para serem executados através de equações predefinidas, o Deep Learning configura parâmetros básicos sobre os dados e treina o PC para aprender sozinho através do reconhecimento de padrões em várias camadas de processamento [6]. Alguns exemplos de softwares que utilizam Deep Learning, temos o caso da Cortana, da Microsoft, que está disponível no Windows 10, porém, não funciona em Portugal ou a Siri, da Apple, disponíveis nos aparelhos da companhia. Quer a Cortana, quer a Siri, são uma assistente inteligente que ajuda o utilizador a realizar algumas tarefas no Sistema Operativo.

## YoloV3 e Darknet

O cerne do projeto era detetar e classificar objetos em tempo real e era necessário usarmos um algoritmo que o fizesse de forma rápida e precisa. Já nos estava especificado no enunciado que enumerava o âmbito do projeto, a existência do YOLO.

O YOLO, uma sigla para You Only Look Once (Apenas Olhas Uma Vez), é o algoritmo mais rápido, preciso e estável disponível para processar o que nós precisávamos. E não só, também existia uma vasta biblioteca de tutoriais e documentação que nos iria ajudar imenso.

O YOLO funciona através da aplicação de uma rede neural simples à imagem, chamada de Darknet. Cada nó dessa rede tem apenas um objetivo, comparar uma região da imagem com um conjunto de dados de um dataset. Dividindo a imagem em regiões e caixas delimitadores facilita a previsão das probabilidades de o objeto estar na imagem ou não [18].

Em termos de performance, existe uma versão mais fraca do YOLO, chamada de Tiny YOLO. Esta é usada para máquinas menos potentes e portáteis, desde CPUs de portátil, até Raspberry Pis [19].

O Darknet enquadra-se nesta questão como quase um adendo e gestor do algoritmo do YOLO. No fundo, o YOLO são blocos de algoritmo a ser corridos pelo Darknet, que depois de os organizar e gerir, monta-nos o seu output, em forma de imagem.

No entanto, por cada instrução em que o Darknet é chamado, o Yolo/Darknet fazem output de apenas uma única imagem de cada vez. E para processarmos streams de vídeo das câmaras, era necessária uma biblioteca que, a partir do input de um ficheiro de vídeo, processasse esta rede neural várias vezes.

O algoritmo do YOLOv3/Darknet está disponível no GitHub de pjreddie em: <https://github.com/pjreddie/darknet>.

## OpenCV

Open Source Computer Vision Library é o nome completo de OpenCV, que é uma biblioteca de funções para visão computacional em tempo real, criada originalmente pela Intel em junho de 2000. Este campo de estudo, a visão computacional, trata de desenvolver métodos para robots e computadores poderem processar imagens e vídeos como informação. Sendo escrita em C++, satisfaz a necessidade de celeridade e de portabilidade destas funções, ajudando no desenvolvimento moderno de software para drones e robôs que têm como objetivo a análise de imagens da forma mais rápida e leve possível.

Dito isto, o OpenCV tem interfaces para Java e Python, não se limitando apenas a C++, que funcionam em todos os sistemas operativos principais do mercado. A nossa escolha foi a interface para Python, sob a licença gratuita de código aberto Apache 2.

Tudo o que o OpenCV é capaz de fazer inclui áreas de aplicação com sistemas de reconhecimento facial, realidade aumentada, captura de movimento, reconhecimento gestual, interação máquina-pessoa, classificador de Naive-Bayes, algoritmo do k nearest neighbor, redes neurais artificiais, deteção de objeto, sendo a última a de interesse especial para o nosso projeto [7].

No entanto, era necessário que este processamento fosse o mais rápido possível, portanto necessitamos não só do processamento do nosso processador, mas também do nosso processador gráfico. E desde 2011 que o OpenCV traz funcionalidades de aceleração de GPU para operações em tempo real. Agora só era necessária uma plataforma para ligarmos o OpenCV com uma placa gráfica.

O OpenCV contém um GitHub oficial, que pode ser acedido em: <https://github.com/opencv/opencv> , documentação online e tutoriais online disponibilizada de forma gratuita e cursos de aprendizagem de IA através de pagamento. Contém parcerias com várias empresas, tais como a Intel, o Microsoft Azure ou a Google [8].

## CUDA

CUDA (Compute United Device Architecture) é uma plataforma de computação paralela e uma API (Application Programming Interface) criada pela Nvidia, lançada a 23 de junho de 2007. CUDA é suportado por GPUs que sejam da Nvidia, a sua licença pertence também à Nvidia e é do tipo GPGPU (General Purpose Processing computing on Graphics Processing Units). CUDA permite aos engenheiros e desenvolvedores de software para usar uma GPU que seja compatível com CUDA para uso de processamento geral sobre a forma de GPGPU. CUDA é uma camada de software que dá acesso direto ao conjunto de instruções virtuais da GPU e elementos de computação paralela, para a execução de kernels. É desenhada para trabalhar com linguagens de programação como C, C++ ou Fortran, esta acessibilidade permite ser mais fácil programar em paralelo usando os recursos da GPU [17].

O uso do CUDA no projeto serviu para melhorar a performance do algoritmo de deteção dos objetos nas máquinas que contém uma GPU da NVIDIA, porém, este foi um processo complicado e até demoroso a realizar-se, conforme será explicado no Ponto 4.2: O CUDA, que faz parte do desenvolvimento do projeto.

## Microsoft Azure

O Microsoft Azure é um serviço de cloud computing, criado pela Microsoft para construir, testar, implementar e, aplicações e serviços administrativos, através de data centers fornecidos pela Microsoft. Providencia software como serviço (Os chamados SaaS – Software as a Service), plataforma como serviço (Os chamados PaaS – Platform as a Service) e infraestruturas como serviço (IaaS – Infrastucture as a Service). Suporta várias linguagens de programação, frameworks, incluindo softwares e sistemas third-party quer software e sistemas da própria Microsoft. O Azure foi apresentado pela primeira vez a 27 de outubro de 2008 e lançado oficialmente em fevereiro de 2010. O serviço funciona no Windows e no Linux, a licença é fechada para plataforma, porém é open source para SDKs (Software Development Kit) para clientes [9].

Dentro da plataforma, uma conta Azure tem acesso a mais de 200 produtos que são fornecidos pelo Azure, dependendo da categoria de trabalho. Dentro destas categorias, são fornecidas ferramentas para, referindo alguns exemplos: Ambiente de Trabalho Virtual do Windows, análise de dados, armazenamento, Blockchain, computação, programação, gestão, IA + Machine Learning, Internet das Coisas, multimédia, segurança ou web [10].

No contexto do projeto, o Azure vai servir-nos para um processamento rápido de captura de imagem ou de vídeo para o reconhecimento de objetos em tempo real e para a interface web que é desenvolvida numa máquina virtual. Ora, havendo estes requisitos para a realização do projeto, a criação de uma máquina virtual com o sistema operativo Ubuntu, que irá correr a interface web, é feita no Azure, assim como a base de dados do Azure para PostgreSQL para o armazenamento de dados e consultas posteriores.

O Azure só foi utilizado na parte final do projeto, devido a problemas com as contas do Azure fornecidas pela UAL, entre nunca se conseguir ter acesso à conta ou a licença terminar de forma abrupta. Dito isto, a construção da base de dados e criação da interface web foram inicialmente realizadas nos PCs dos elementos do grupo e depois portado para o Azure, que, infelizmente, não correu como era esperado inicialmente pelo enunciado do projeto.

## Python

Python é uma linguagem de programação de alto-nível, multi-paradigma, ou seja, suporta vários paradigmas de programação, lançada em fevereiro de 1991. É uma das linguagens de programação mais populares [13]. A sua documentação está disponível livremente em: <https://docs.python.org/3/> . A escolha desta linguagem no projeto, deve-se a vários motivos: Para começar, a versão do Python que foi utilizada no projeto foi a versão 3.9.4. É uma linguagem na qual já trabalhamos bastante ao longo do curso, que estamos familiarizados com ela para construir o programa requerido, para a utilização do Flask, que é uma framework de Python para páginas web, para fazer os testes do YOLOv3 e do Darknet para a deteção de alvos e também por ser uma linguagem que se dá bem com a linguagem SQL, focada para criação de base de dados, através da biblioteca SQLAlchemy do Python, que será expandida no ponto a seguir. Todas estas vantagens fizeram que escolhêssemos Python como a linguagem de programação a ser utilizada no projeto.

## SQL

SQL, Structured Query Language, é uma linguagem de domínio específico usada para programação e gerenciamento de dados para gerenciamento de base de dados relacional seguindo um paradigma multiparadigma, lançada em 1974 por Donald D. Chamberlin e Raymnond F. Boyce, que tinha como objetivo inicial demonstrar a viabilidade da implementação do modelo relacional. SQL diferencia-se das outras linguagens de programação de base de dados no sentido em que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele. SQL chegou a ser revisto em 1992, em 1999 e 2003, para introduzir algumas novas funções, como triggers, em 1999 e características relacionadas ao XML em 2003. A linguagem pode ser facilmente migrada de plataforma para outra plataforma sem grandes mudanças estruturais e permite ligação com outras linguagens de programação como Java, C ou Python [27].

No contexto do projeto, SQL é a linguagem que utilizamos para criar a base de dados criada no PostgreSQL e permitir a ligação com Python a partir do SQLAlchemy.

## CSS, JavaScript e HTML

HTML, CSS e JavaScript são linguagens de programação viradas para programação web, que estão bastante conectadas entre si. Começando por HTML, significa HyperText Markup Language, é a linguagem standard para documentos mostrados em um browser de Internet, criada em 1993. É assistido por tecnologias como o CSS (Cascading Style Sheet) e linguagens de script como o JavaScript. Os browsers de Internet recebem documentos HTML a partir de um servidor web ou a partir de um lugar local e faz renderização dos documentos em multimédia na página web [21].

Já CSS (Cascade Style Sheet), é uma linguagem de folha de estilos, lançada em 1996, usada para mostrar um documento escrito em uma linguagem de marcos como HTML. É uma das linguagens principais da World Wide Web ao lado de HTML e JavaScript. CSS é projetado para possibilitar a separação da apresentação e do conteúdo, incluindo layouts, fontes ou cores [22].

JavaScript é uma linguagem de programação que está em conformidade com a especificação ECMAScript, lançada inicialmente a 4 de dezembro de 1995. É uma linguagem de alto nível, normalmente compilada em tempo real e multi-paradigma. Contém orientação a objetos e funções de primeira classe. Ao lado de HTML e CSS, JavaScript é uma das linguagens mais usadas na World Wide Web, enquanto linguagem de multi-paradigma, JavaScript suporta paradigmas de programação imperativa, funcional e orientado por eventos. Contém APIs para trabalho de texto, estruturas de dados, datas ou DOM (Document Object Model, Modelo de Objeto de Documentos em português) [23].

No projeto estas três linguagens foram essenciais para a construção da Interface Web requerida no enunciado do projeto.

## PostgreSQL

PostgreSQL é um software gratuito e open source de gestão de base de dados relacional, feito para ser compatível com SQL agnóstico. Como outros softwares de base de dados, o PostgreSQL efetua transações com as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), visualizações atualizadas de forma automática, vistas materializadas, triggers, chaves estrangeiras e stored procedures. Está desenhado para lidar com uma variedade de cargas de trabalho, desde máquinas simples até Data Warehouses ou Web Services com um número forte de usuários em simultâneo [11]. Para nós, apenas nos interessou três características: o facto de já termos usado o software anteriormente na universidade, a vasta documentação disponível online e o suporte completo dado pelo Microsoft Azure a esta tecnologia.

## Flask

Flask é uma micro framework web escrita em Python, lançada a 1 de abril de 2010. A sua licença é BSD. É classificada desta maneira porque não requere ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível. Flask não possui uma camada de abstração de base de dados, nem de validação de formato, nem de outros tipos de componentes onde bibliotecas third-parties providenciam funções comuns. Sites como o Pinterest ou o Linkedin usam esta framework[12]. O Flask contém um GitHub que pode ser acedido em: <https://github.com/pallets/flask> e a sua documentação é gratuita, disponível em: <https://flask.palletsprojects.com/en/2.0.x/> .

O uso de Flask no projeto está relacionada com a construção da página web onde apresenta os resultados e a classificação da deteção de objetos da imagem ou do vídeo representado.

## SQLAlchemy

SQLAlchemy é uma toolkit open-source de SQL e mapeamento objeto-relacional (ORM – Object-relacional Mapping em inglês) para linguagem Python, lançada sobre a licença MIT a 14 de fevereiro de 2006[14]. Providencia um conjunto completo de padrões de persistência de nível empresarial bem conhecidos, desenhados para eficiência e alta performance no acesso à base de dados, adaptada numa simples linguagem e dominante como é Python [15]. Empresas como o Reddit ou a DropBox utilizam SQLAlchemy. Contém um GitHub oficial, disponível em: <https://github.com/sqlalchemy/sqlalchemy> e a sua documentação está disponível em: <https://www.sqlalchemy.org/library.html> .

A escolha da utilização do SQLAlchemy no projeto deve-se à facilidade de conectar o código Python do algoritmo de deteção e a página web com a base de dados, PostgreSQL, escrita em SQL, na mesma linguagem.

## Bootstrap

Bootstrap é uma framework front-end de CSS gratuita e open-source lançada pela Bootstrap Core Team a 19 de agosto de 2011. É responsiva, ótima para desenvolvimento de páginas web em dispositivos móveis ou diferentes tipos de janelas. Vai neste momento na versão 5.0.1, é escrita em HTML, CSS, Less, Sass e JavaScript e está sobre a licença MIT License. Bootstrap contém templates de design de CSS e de JavaScript para tipografia, formas, botões, navegação e outras componentes de interface [20]. Contém um GitHub oficial em: <https://github.com/twbs/bootstrap> , a sua instalação está disponível em: <https://getbootstrap.com/docs/5.0/getting-started/download/> e contém documentação oficial em: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>.

No projeto, o uso de Bootstrap serve para a Interface Web ser compatível com qualquer dispositivo e de qualquer tamanho (por exemplo telemóveis), sem prejudicar o conteúdo disponível no site.

## Ubuntu

Ubuntu é uma distribuição Linux baseada em Debian e consiste na sua maioria em software open-source e gratuito, criada pela Canonical Ltd. onde o lançamento inicial ocorreu a 20 de outubro de 2004. O Ubuntu está disponível em três edições: Ubuntu Desktop, a edição normal do Sistema Operativo, Ubuntu Server, especializada para servidores web e Ubuntu Core para dispositivos do Interent of Things e robôs. Qualquer uma das edições pode correr em um computador ou numa máquina virtual. Ubuntu é um sistema operativo popular para computação por nuvem, com suporte para OpenStack. Desde a versão 17.10 que o desktop padrão do Sistema Operativo tem sido o GNOME. O Sistema Operativo contém atualizações constantes, com novas versões a cada seis meses. À data de escrita deste relatório, a última versão do Ubuntu lançada foi a 21.04 a 22 de abril de 2021. Conforme referido antes, Ubuntu é desenvolvido principalmente pela Canonical e por uma comunidade de outros desenvolvedores seguindo um modelo de governação sob a forma de Meritocracia. A Canoncial disponibiliza atualizações de segurança e suporte por cada lançamento do Ubuntu, começando desde o seu lançamento até chegar à data planeada para o fim de vida de uma versão do Ubuntu [24]. O Ubuntu está disponível gratuitamente em: <https://ubuntu.com/download> .

No contexto do projeto, o Ubuntu é usado como uma máquina virtual, na edição Desktop, fornecida pela parceria da UAL na realização deste projeto, a LEITEK INNOVATIVE SOLUTIONS que contém 2GB de RAM, não contém GPU e como CPU um AMD Ryzen 9 3900X com 12 cores. Esta VM servirá para correr o nosso programa, de forma a termos acesso a esta máquina foi utilizado um software chamado TeamViewer de forma a podermos mexer na máquina fornecida de forma remota.

## TeamViewer

TeamViewer é um software de computador que permite acesso e controlo remoto, lançado originalmente em 2005 pela TeamViewer AG, que permite manutenção de computadores e outros dispositivos. Este software, a sua licença é proprietária, porém não requer qualquer tipo de registo e é de uso livre para uso não-comercial. O TeamViewer está disponível no Windows, macOS, Linux, Android, Chrome OS, Raspberry Pi e iOS [25][26].

Em contexto de projeto, o TeamViewer foi-nos fornecido pela LEITEK INNOVATIVE SOLUTIONS de forma a acedermos à máquina virtual Ubuntu da empresa de forma a corrermos o programa nessa máquina.

# Fases de Desenvolvimento do Projeto

## Primeira Fase: Protótipo e Planeamento

A primeira fase do projeto foi marcada por imensa discussão e *brainstorming* à volta de como iríamos fazer o que nos é pedido. Tínhamos todos ideias razoavelmente diferentes de como implementar os componentes necessários para o projeto, mas após alguma pesquisa, rapidamente, entendemos como as coisas iam ser implementadas.

Numa fase inicial do desenvolvimento do projeto, decidimos começar por experimentar, através de um pequeno programa, o algoritmo de Deep Learning do YOLOv3/Darknet, disponível no GitHub de pjreddie, com suporte ao YOLO normal e ao YOLOtiny, para dispositivos mais fracos, como telemóveis ou portáteis com gama U de processador. Este pequeno programa serviu para testar a capacidade do algoritmo em detetar objetos antes de começarmos com o projeto em si. Os resultados que obtivemos nestes testes foram, de forma geral, satisfatórios. Detetamos que o algoritmo deteta com bastante precisão, pessoas, ao ponto de só precisar de uma mão aparecer na câmara para o algoritmo dizer que é uma pessoa. Porém, nestes testes verificou-se que o algoritmo não é tão eficaz em detetar alguns objetos diferentes, ainda que os consegue reconhecer alguns. De seguida, apresenta-se algumas imagens (figuras 1 a 5) retiradas pelos elementos do grupo de forma a experimentar as capacidades do algoritmo e os seus resultados. Os resultados das figuras mostramos que o YOLO normal nas nossas máquinas locais apresenta bons resultados de precisão, porém a framerate é bastante baixa, enquanto o YOLOTiny tem uma framerate superior, a eficácia do algoritmo em detetar coisas pode falhar ou a eficácia cair para metade.

Houve também a primeira reunião com a Leitek em que conversámos sobre o YOLO e o DarkNet. Sabíamos agora que o projeto seria indiscutivelmente uma plataforma online com FrontEnd para um sistema de câmaras, e qualquer fonte de streaming. Portanto o primeiro desafio seria implementar o OpenCV, juntamente com o YOLO e o Darknet a funcionar.

A nossa primeira escolha, além de vermos como funcionava o OpenCV e o YOLO, seria a plataforma de Python em que o iríamos usar. Tínhamos duas escolhas: O Django e o Flask. Um dos membros do nosso grupo já tinha experiência profissional com Django, e sugeriu o Flask, por ser mais leve e focado em websites com menos operações. Tendo em conta que o Django é mais indicado para websites grandes com muitos componentes diferentes, tais como o YouTube e o Spotify. Para nós, o Flask parecia uma alternativa mais simples e focada numa operação apenas, que era o nosso objetivo. Após pesquisa, conseguimos implementar um esboço de um protótipo.

Usando os componentes Python do Flask e do OpenCV, entendemos rapidamente como o OpenCV, funcionava. Embora parecesse algo complexo, na verdade, era um processo simples. A cada frame do vídeo ou da stream, a IA analisava a deteção de objetos na imagem, sabíamos também que a IA do OpenCV tinha maior facilidade a detetar objetos caso não existissem cores no frame do vídeo a ser analisado, portanto tínhamos de transformar todos os frames em *greyscale.* Depois, conseguindo implementar um dataset de rostos de pessoas, desenhávamos quadrados à volta deles no frame, e apresentávamos de volta a stream ao utilizador. Todo este esboço não tinha ainda front-end nem website, mas era totalmente servido pelo Flask. Sabíamos que estávamos a ir no bom caminho.

Mais tarde, começámos a dividir o projeto em dois, um com o YOLOv3 e o outro com o Tiny Yolo. O Tiny era uma versão menos intensiva nos recursos do algoritmo, feita especialmente para telemóveis. O Tiny foi adotado por dois membros do grupo com CPUs de portátil mais fracos, de forma mais específica, da gama U da Intel, e o YOLOv3 foi adotado pelos outros dois membros, que tinham ao seu dispor computadores mais potentes. O teste e a pesquisa ao longo desta semana levou à correção de alguns bugs menores, mas, mais importante, levou ao desenvolvimento do nosso primeiro, completo protótipo, agora a funcionar com todos os objetos do dataset no YOLOv3 e no Tiny Yolo.

As figuras que se seguem, figuras 1 a 5, mostram os testes realizados nesta fase utilizando as capacidades do algoritmo do YOLO e do Tiny YOLO. Os resultados mostram testes com alguns objetos, quando usado o YOLO reconhece-os bem, enquanto o Tiny YOLO apresenta alguma dificuldade em reconhecer com clareza o objeto correto.

Uma imagem com texto, homem

Descrição gerada automaticamente

*Figura 1 - Rosto de Ricardo Cardoso a ser reconhecido pelo algoritmo com uma eficácia de 52% usando o YOLOTiny.*

Uma imagem com texto, interior, pessoa

Descrição gerada automaticamente

Figura 2 – Rosto de Diogo Mendes reconhecido pelo algoritmo com uma eficácia de 100% usando o YOLO.

Uma imagem com texto, pessoa, parede, interior

Descrição gerada automaticamente

Figura 3 – Rosto de Reinaldo Coelho reconhecido pelo algoritmo usando o YoloTiny. Aqui verifica-se que o algoritmo ainda não consegue reconhecer uma caixa de um videojogo.

Uma imagem com texto, captura de ecrã, monitor, interior

Descrição gerada automaticamente

Figura 4 – Uma nova experiência de Reianldo Coelho com outra caixa de videojogo mostra alguma dificuldade do algoritmo em detetar corretamente pessoas em uma caixa. Pode-se ver que o algoritmo consegue reconhecer como uma pessoa, o jogador Cristiano Ronaldo, já o jogador Marcus Rashford, o algoritmo reconhece-o como uma garrafa.

Uma imagem com texto, parede, monitor, interior

Descrição gerada automaticamente

Figura 5 – Quarto de Bruno Silba, onde o algoritmo reconhece uma televisão, cadeira e uma garrafa com uma eficácia acima de 90%, mostrando que o algoritmo consegue identificar mais objetos além de pessoas utilizando o YOLO.

## Segunda Fase: O CUDA

CUDA é uma API da NVIDIA para computação gráfica paralela e no que toca a melhorar performance no algoritmo, tínhamos pouca escolha senão utilizá-la. Foi aqui que sentimos as nossas primeiras dificuldades, mas a documentação da NVIDIA deu a assistência necessária para o conseguir fazer facilmente [28]. No entanto, compilar o CUDA e implementá-lo juntamente com o OpenCV tornou-se trabalhoso e demorado, e foram precisos alguns dias de pesquisa para o efeito desejado ser conseguido.

No Windows, a NVIDIA disponibiliza instalações do CUDA e do CuDNN, e apenas precisávamos de uma conta no NVIDIA Developer Program. Após instalar ambos, compilámos e instalámos o OpenCV juntamente com o CUDA utilizando o CMake [29]. Após testar que o nosso sistema e o OpenCV estava a usar o GPU, foi tempo de metermos mãos à obra no código.

Utilizando agora o CUDA, tínhamos melhor performance e criámos um zip do OpenCV pré-compilado com CUDA para as gráficas GTX série 10XX da NVIDIA, que dois membros do grupo tinham, e assim, com isto, tínhamos o YoloTestes com o CUDA a funcionar.

Tentámos mais tarde instalar o CUDA no Azure. Usámos um GPU com drivers CUDA da NVIDIA já instalados, o nvcc, e precisávamos agora de instalar o OpenCV em conjunto com o CUDA [30]. Ao tentar instalar tivemos um erro que não conseguimos encontrar solução [31], e encontrámos bastante falta de disponibilidade no Azure para sistemas de GPU. Após múltiplas tentativas, o nosso saldo no Azure tinha terminado, e ficámos sem capacidade de continuar a implementar o nosso projeto nele.

Foi também nesta altura do projeto, que estávamos a experimentar ideias diferentes. Uma das ideias falhadas foi utilizar um módulo de tradução automática da Google para objetos detetados, parecia uma excelente ideia, mas o Google traduzia automaticamente *sink* para *afundar* em vez de lava-louças, e outras falhas que dariam demasiado tempo a corrigir, então traduzimos tudo à mão. Depois disso, iniciámos a otimização para multi-utilizador e multi-thread nas câmaras. Houve algumas dificuldades nessa parte, incluindo um bug do OpenCV, quando se usava duas câmaras [32], que supostamente tinha sido corrigido na versão 3, mas continuava a existir nas nossas versões. Após muita martelada, descobrimos que o problema era o nome das janelas, e finalmente tínhamos multi-processamento e multi-utilizador no nosso site, com várias sessões.

Por último, criámos um mecanismo de deteção do CUDA no sistema, e ele corria de acordo com uma flag *true* ou *false* devolvida por esse mecanismo. Caso fosse *false*, a aplicação web funcionava normalmente com o OpenCV sem CUDA, e se fosse true, usaria o CUDA e obtínhamos performance superior. Tínhamos chegado a outra versão beta do nosso projeto, com multi-processo, uma UI experimental, sessões, login, painel de controlo do utilizador e acesso direto a streaming de câmaras ligadas.

Nas figuras que se seguem, figuras 6 a 10, serão mostrados os passos ocorridos nesta fase, começando pela máquina virtual fornecida pelo Azure com os GPUs da NVIDIA que usamos para poder compilar o CUDA no serviço de cloud da Microsoft na figura 6. Na figura 7 vemos as especificações da máquina virtual que tivemos acesso durante a compilação, figura 8 mostra o CUDA a ser compilado com sucesso na máquina virtual do Azure, figura 9 demonstra o OpenCV a ser compilado no CUDA e a figura 10 demonstra um teste realizado com sucesso com deteção de duas câmaras em simultâneo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6 – Máquina Virtual do Azure com GPUs que usámos para o CUDA.

Uma imagem com texto

Descrição gerada automaticamente

Figura 7 – Especificações da VM utilizada no Azure, com SO Ubuntu 18.04, CPU Intel Xeon Platinum 8171M com 4 núcleos de velocidade 2,095GHz, 16GB de RAM.

Uma imagem com texto

Descrição gerada automaticamente

Figura 8 – CUDA compilado com sucesso no Azure, através do nvcc.

Uma imagem com texto, eletrónica, captura de ecrã

Descrição gerada automaticamente

Figura 9 – Compilação do OpenCV com o CUDA na máquina do Azure.

Uma imagem com texto, monitor, captura de ecrã, ecrã

Descrição gerada automaticamente

Figura 10 - Teste de Bruno Silva realizado com sucesso sobre deteção de duas câmaras em simultâneo, apresenta em ambos uma eficácia de 94% usando o YOLO com CUDA na máquina local.

## Terceira Fase: A Base de Dados

Na terceira fase do nosso projeto, tivemos de desenhar a base de dados, e aprender o módulo do Python que comunicava com ela. Devido a experiências anteriores, escolhemos PostgreSQL para a nossa base de dados, e SQLAlchemy, para as nossas comunicações com o backend. A forma como o SQLAlchemy funciona, é através de um módulo que a documentação chama de Engine. Esta, é responsável por ser o ponto inicial de todas as comunicações com a base de dados, que depois converte para um dialeto de SQL, em que o PostgreSQL está incluído [34].

engine = create\_engine(config.database, echo=False, future=True)

Nesta linha de código, estamos a criar o Engine mencionado anteriormente, que comunica com a base de dados através de *statements* SQL, que são instruções para operações com a base de dados em SQL.

Essencialmente tivemos de satisfazer quatro tipos de operação distintas, chamadas de CRUD (acrónimo para Create, Read, Update and Delete) [35]. Além das operações, tivemos de desenhar a base de dados para satisfazer todos os requisitos do nosso programa.

Ao fim de bastantes revisões, o desenho nosso final da base de dados pode ser visto na figura 11. Segue-se também uma explicação dos conteúdos de cada uma das tabelas.

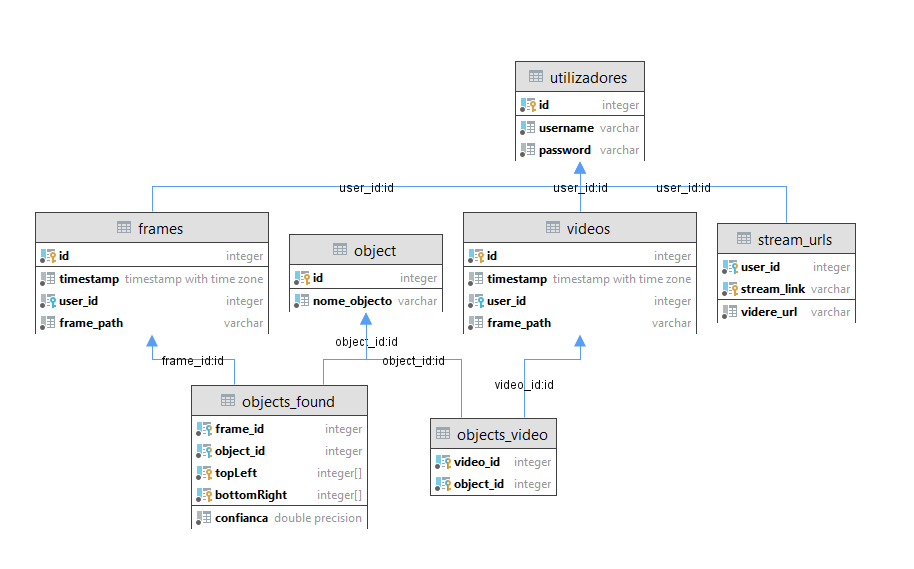


Figura 4.11 - Desenho da Base de Dados realizada, contém as classes frames, object, objects\_found, objects\_video, vídeos, stream\_urls e utilizadores

Algumas destas tabelas são básicas de compreensão. A tabela utilizadores, obviamente, guarda cada utilizador e password. No entanto, é de realçar que estas passwords são guardadas encriptadas com o módulo bcrypt do Python. Usando o bcrypt, durante o registo do utilizador, o nosso programa faz encode, aplicando a função de hash a uma password. Durante a verificação do utilizador, ele verifica se na base de dados existe uma password encriptada igual à codificação do input do utilizador, com a função *checkpw* [36]*.* A tabela frames, outra tabela importante para o nosso objeto, guarda cada frame em que foi detetado objetos, com a sua respetiva timestamp. Relaciona com *objects\_found,* onde cada objeto detetado será relacionado com uma tabela *object*, que contém todos os objetos do nosso dataset. A mesma lógica é aplicada aos vídeos, que também tem *timestamp* e uma relação de objetos encontrados na tabela *objects\_video.* O motivo pelo qual estas tabelas são distintas trata-se que, na *objects\_found¸* dedicada a frames individuais, existem campos de coordenadas para as caixas se encontrarem, enquanto nos vídeos, o vídeo é renderizado já com as caixas a serem processadas pelo OpenCV, e não é necessário guardar as coordenadas na base de dados.

Por último, e possivelmente, mais importante, a tabela *stream\_urls*, contém todos os links com câmaras e streams disponíveis, organizadas por utilizador. É de realçar que os frames e os vídeos também têm uma relação com os utilizadores, para existir um sistema de conteúdo, separado por cada utilizador. Isto para impedir que os utilizadores tenham acesso a conteúdo de outros utilizadores.

Esta fase além de ter sido a mais demorosa e com mais tentativas e erros, foi também onde avançamos bastante na UI do nosso projeto. Outra opção importante desta fase, foi a implementação de um ficheiro de configuração (*config.py*) onde seria guardado todas as variáveis configuráveis de sistema do nosso projeto. O código foi otimizado, os bugs do Flask das sessões foram resolvidos, e estávamos prontos agora para mostrar todas as imagens e frames guardados na galeria do utilizador.

De seguida são apresentadas duas figuras, 12 e 13, onde mostram um protótipo a Interface da Página Web realizada. Na 12 vê-se um protótipo da página das câmaras e na 13 um vídeo realizado para testes, que foi dado upload para o site para o algoritmo do YOLO detetar pessoas, que foi um sucesso.

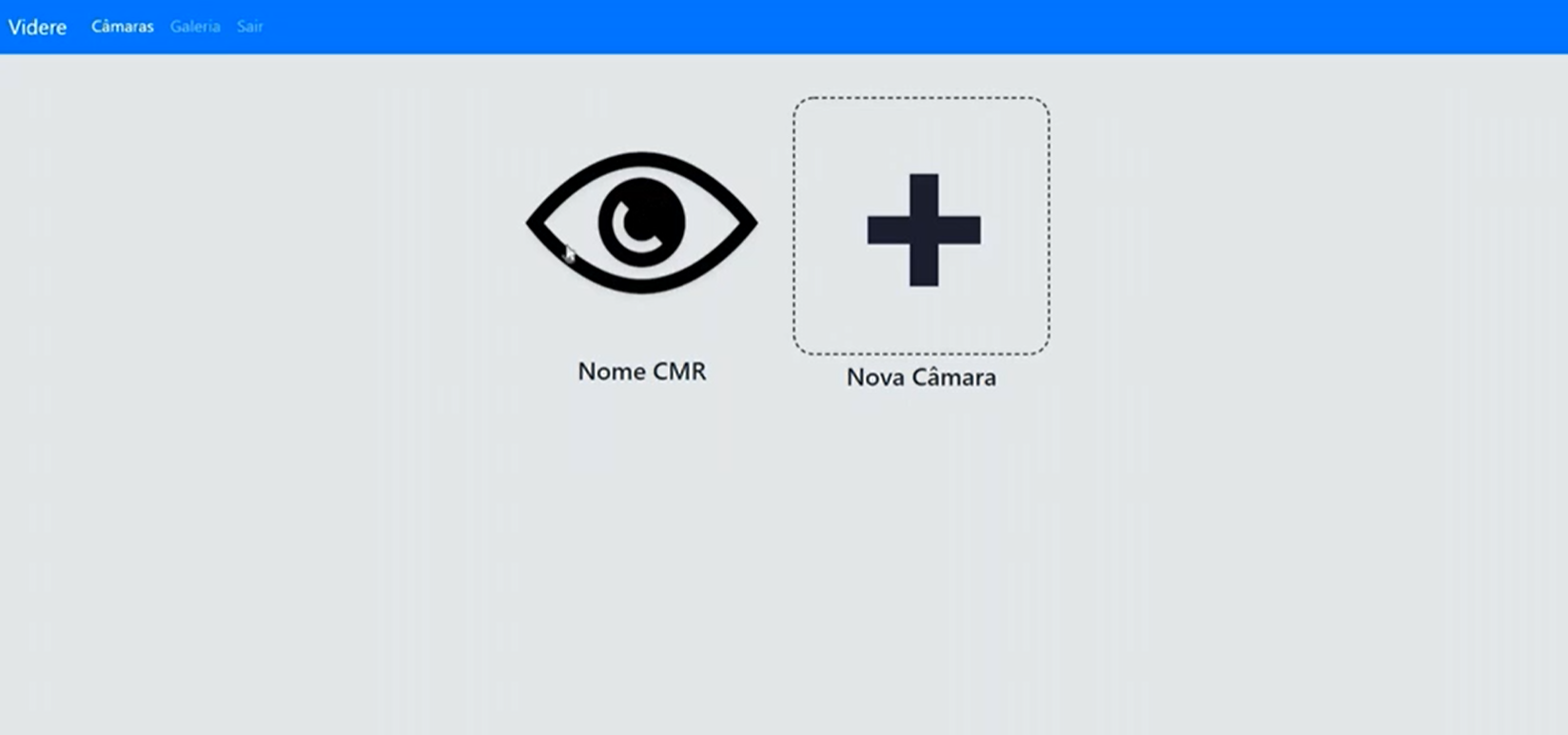


Figura 12 - Protótipo da Interface Web, página das câmaras

Uma imagem com texto, eletrónica

Descrição gerada automaticamente

Figura 13 - Protótipo da Interface Web, deteção de pessoas pelo YOLO através de upload de vídeo. A página também apresenta uma barra para o brilho e para o contraste.

## Quarta Fase – A Galeria

Toda a estrutura da base de dados foi formada para um único motivo, a galeria. A galeria seria discutivelmente a parte mais importante e complexa do projeto, que tinha de satisfazer os seguintes requisitos:

* Apresentar todos os frames e vídeos guardados.
* Apresentar os frames e os vídeos separadamente.
* Apresentar apenas o conteúdo do utilizador em sessão.
* Apresentar os filtros para filtrar a seleção de imagens apresentadas.

Sendo assim, a UI da galeria foi uma guerra longa de ficar bem estruturada. Ao fim de algumas tentativas, conseguimos centrar as imagens de forma correta, mas ainda demorou algum tempo.

Na figura a seguir, a figura 14, mostra-se um protótipo da interface da Galeria, com os frames guardados e as pessoas a serem detetadas.

Uma imagem com texto, diferente

Descrição gerada automaticamente

Figura 14 – Protótipo da Interface da Galeria.

Outros bugs desta fase foram relacionados com a *navbar* não ficar no tamanho certo quando a janela diminuía. E os quadrados dos objetos detetados passaram a ter cores diferentes.

A lógica da população da galeria é simples, quando processamos uma stream, em cada 5 segundos, se que um objeto novo é detetado, ele envia o frame em que o detetou para a base de dados registado com a autoria do utilizador em sessão. A galeria depois busca todos os frames guardados desse utilizador para apresentar de acordo com os filtros.

Os vídeos são organizados de forma mais simples, sendo que são enviados para a base de dados cada vídeo processado que encontra objetos, a galeria simplesmente vai buscar todos os que encontra.

## Quinta Fase – A reta final

Durante a nossa reta final do projeto, depois da galeria estar terminada, dedicámo-nos a resolver bugs e problemas do nosso projeto. E após testar, eram bugs de design e de funcionamento.

Foi implementado o sistema de carregamento de ficheiros e vídeos individuais, em vez de apenas streaming ou câmaras. Neste sistema, depois mais tarde, implementámos uma barra de progresso durante o processo dos vídeos.

Na figura a seguir, figura 15, podemos ver uma barra de progresso enquanto se carrega um vídeo para o site.

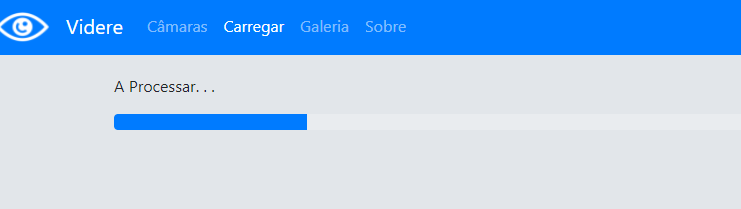


Figura 15 – Barre de progresso do processamento de um vídeo no site.

Foi descoberto nesta altura um problema no Firefox que fazia um memory leak no aplicativo, que impossibilitava o uso do computador. Esse problema já existia desde o início, mas pensávamos que era um problema do Linux, o que fez ocorrer que quando estivemos a utilizar a máquina fornecida pela Leitek para o correr o programa na sua máquina, crashasse algumas vezes. No entanto, estava associado ao Firefox.

Entretanto, na galeria, era necessário dividir os vídeos dos frames.

Na figura que se segue, a figura 16, podemos ver mais sobre a galeria onde faz a seleção entre vídeos e imagens guardadas na Galeria, em ambos com os resultados da deteção de objetos.

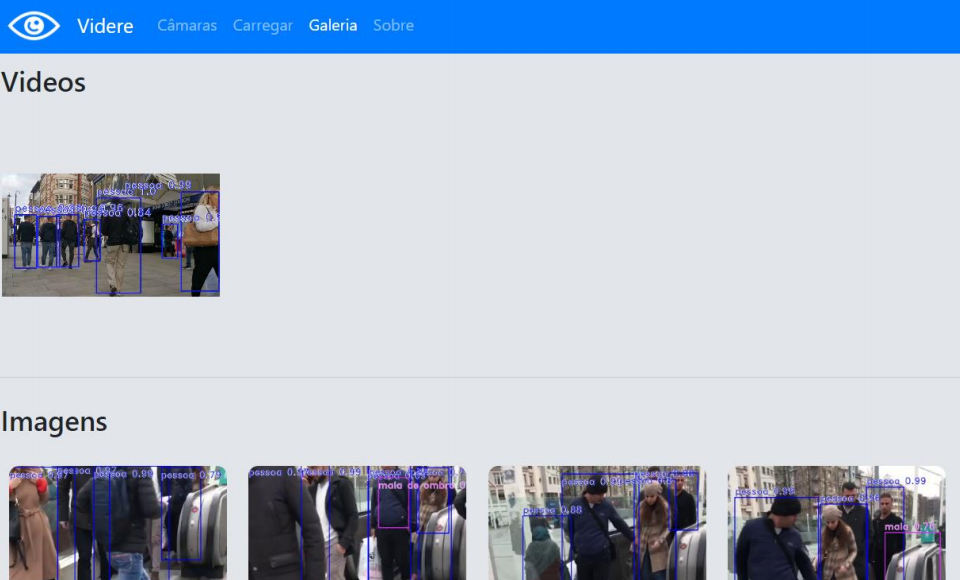


Figura 16 – Página da Galeria contendo Vídeos e Imagens em separado. Em ambos, podemos ver os resultados da deteção com bastante eficácia.

Algumas outras implementações de HTML durante esta reta final, basearam-se em UX como por exemplo mostrar a imagem processada ao utilizador depois de o processamento terminar, verificação se o ficheiro é um vídeo ou imagem para organizar corretamente, e um sistema para terminar as threads corretamente após o final do processamento.

Depois destes bugs serem arranjados, assim como apagar informação extra na base de dados, demos o projeto como terminado.

# Funcionamento do Código da VidereApp

A VidereApp contém vários ficheiros de código para o seu funcionamento, e na parte de Python temos os ficheiros: app.py, camara.py, carregarFicheiros.py, config.py, dataset.py, galeria.py, novaCamara.py, run.py, utilizador.py e videredb.py.

O run.py é o ficheiro mais pequeno destes todos porque o seu objetivo pretende-se principalmente em apenas inicializar a execução do programa, com a configuração da porta, da flag de debug e das pastas. O objetivo da criação deste ficheiro é por motivos de organização, evitando que haja milhares de linhas no mesmo ficheiro, dificultando a sua compreensão. Ao estarem em ficheiros separados, torna-se mais fácil de se entender os processos envolvidos.

Segue-se o config.py, onde o objetivo deste é guardar todas as variáveis configuradas no projeto. De forma muito simples, ela guarda parâmetros da porta utilizada, da flag de debug, das pastas, frames e vídeos, parâmetros do YOLO, que estão de forma pré-definida, o algoritmo do YOLO normal, caso se pretenda alterar para o Tiny YOLO, será necessário alterar este parâmetro no código. Guarda também os parâmetros da Base de Dados e da Sessão. Existe uma função chamada pastas() que verifica se existe a diretória de vídeos e de frames, caso elas não existam, serão criadas.

Já a novaCamara.py, serve para configurar uma nova câmara na página web do projeto. Aplica-se o processo do Blueprint para a página web das câmaras introduzidas. O Blueprint é um conceito que o Flask usa para a realização de componentes de aplicação e suportar padrões comuns dentro de uma aplicação ou entre aplicações [37]. Existe um processo de routing que utiliza os métodos POST e GET para obter a galeria. A função novaCamaraStream() tem como propósito descobrir se a nova câmara a ser introduzida está a respeitar os requisitos necessários. Irá ser feito um requisito para descobrir o nome e o link da câmara, e uma string chamada filtro Objetos para verificar se há objetos para serem detetados pela nova câmara. Caso a câmara não conter um nome, um link ou algum objeto para ser detetado, será envidada uma mensagem de erro que redireciona o URL de novo para a página da nova câmara. Também ocorrerá erro se existir uma câmara guardada com o mesmo nome com a câmara a ser introduzida. Caso o processo realizado seja um sucesso, a página vai redirecionar para a página de login.

O dataset.py liga-se, de certa forma, com o config.py. O objetivo deste é chamar o dataset que o YOLO traz, que contém 80 objetos chamado coco. O que se pretende aqui é que a cada 10 objetos diferentes detetados pelo algoritmo, haja uma mudança na cor do quadrado da deteção, para ser mais simples de se identificar como resultados diferentes. Ora, como são 80 objetos, haverá no total 8 cores para os resultados da deteção. As cores escolhidas estão escritas sob a forma de RGB no código.

O camara.py tem como função permitir o utilizador, que seja dono de uma certa stream, tenha acesso à mesma. Volta-se a aplicar o processo do Blueprint para a página da stream, guardada e processos de routing que usa os métodos POST e GET, para obter o feed da string. A função janelaCamara() permite verificar se o utilizador é dono da stream, para assim poder acedê-la. Ao permitir o acesso à stream, o utilizador pode fazer um requisito para apagar a stream do site, assim como pode fazer um requisito para alterar o brilho, o contraste ou os filtros da stream, caso seja um requisito do método POST. Através do método GET, o utilizador vai obter os dados da sua stream, como ID, contraste, nome da stream, brilho ou filtros. Ao terminar as suas funções, volta para a página de login.

O carregarFicheiros.py, conforme o nome indica, serve para carregar um ficheiro que o utilizador pretenda, seja vídeo ou imagem, para o website para depois ocorrer a deteção de objetos da imagem/vídeo. Volta-se a aplicar o Blueprint para a página e processos de routing com os métodos POST e GET para carregar os ficheiros. A função carregaFicheiros() vai verificar se o utilizador está a carregar o ficheiro para o site de forma correta. Vai aparecer uma mensagem de erro se não houver nenhum ficheiro selecionado ou se o pedido que foi feito, não ter nenhum ficheiro. Haverá também erros se o utilizador não escolher nenhum filtro para inicializar a deteção ou o formato do ficheiro escolhido não seja PNG, JPG, JPEG ou MP4. O processo termina com um if onde segue um processo para imagem e outro processo para o vídeo. No processo da imagem, a frame usa dois processos do OpenCV para leitura e escrita de imagens e o cv2.cvtColor() que é usado para converter uma imagem de um espaço de cor para outra. Após isto, será criado uma REST *call* com a imagem que irá ser o link desta que é acedido pelo /verimagem/. Caso seja um vídeo, vai gerar um nome para o ficheiro que vai servir como ID e irá ser guardado no site. Ao terminar o processo do vídeo, é criado um link para aceder ao vídeo no /vídeo. Caso o utilizador não estiver na sessão, é redirecionado para a página do login.

O galeria.py tem como objetivo, o utilizador tenha acesso a dados guardados que estejam no site. Aplica-se, mais uma vez, o processo do Blueprint para a página da galeria e os processos de routing, com os métodos POST e GET para obter a galeria. A função janelaGaleria() pretende que o utilizador, ao conectar-se, que vá buscar as imagens e os vídeos que estejam guardados, seguindo-se da renderização desses mesmos dados no ficheiro galeria.html. Não estando ligado à sessão e a página redireciona para a página de login. A função obremFrameGaleria() vai buscar os frames guardados e manda o utilizador para a diretoria da mesma. A função VideoGaleriaVer(feed) vai mostrar os vídeos da galeria do utilizador onde estará o seu feed e o seu dono. Contém a opção de apagar o vídeo, caso seja o seu desejo. Contém os métodos POST e GET no processo de routing onde o GET serve para mostrar o vídeo e o POST para apagar o vídeo. A última função deste ficheiro, o paginaVerImagem() que segue exatamente os mesmos processos do VideoGaleriaVer(), apenas num separador diferente. Se o utilizador proceder ao método GET, receberá as imagens guardadas com a data da mesma e o seu ID do dono, caso faça um requisito ao método POST, dá-se a possibilidade de apagar as imagens do site.

A app.py, começa pela app ser feita em Flask, uma framework para Python que contém registos de Blueprint para a câmara, galeria, nova câmara e carregamento de ficheiros. Volta-se a usar os processos de routing com os métodos POST e GET. A função login(), conforme o nome indica, verifica se as credenciais que o utilizador meteu, nome e palavra-passe estejam corretas e se o utilizador está registado no site, caso o requisito seja POST. Se estiveram alguma credencial errada, aparecerá uma mensagem de erro e é redirecionado para a página de login. A função registo() serve para criar um utilizador novo, que tem de cumprir certos requisitos como a senha e o nome não podem conter espaços em branco, o nome de utilizador tem de ser diferente dos registados e a senha tem de ter pelo menos 5 caracteres. E não cumprir estes requisitos haverá uma mensagem de erro e o registo terá de ser realizado novamente. A função painel() serve como painel de controlo para o utilizador, onde pode iniciar uma nova câmara ou obter a ligação da câmaras na sua sessão. O progressoProcessoVideo(), como diz o seu nome, demonstra o progresso do processo do vídeo escolhido. VideoPaginaProcesso() é uma função que mostra a página do resultado do vídeo em um ficheiro processado. A função videoProcessadoResultado() mostra o resultado do vídeo carregado pelo utilizador no formato MP4, processado. A função transmitirImagem() obtém um vídeo de uma câmara de um utilizador. Já o transmitirthumbnail() tem como objetivo obter uma thumbnail para o vídeo escolhido. Se não houver frames disponíveis para serem a thumbnail do vídeo, apenas será escolhida uma imagem comum para carregar o vídeo.

A videredb.py é o ficheiro que faz ligar a interface web com a base de dados feita, neste ficheiro encontra-se alguns comandos de SQL no código Python, graças ao SQLAlchemy que permite fazer a ponte entre SQL e Python. O ficheiro começa por criar uma engine que configura a base de dados. A primeira função que aparece, selectTabela() inicializa as quatro operações básicas do CRUD, criar, ler, atualizar e apagar. As funções seguintes acabam por ser processos simples e de fácil compreensão, inserirUtilizador() fará que o utilizador registado tenha a sua conta guardada na base de dados e o verfiicaUtilizador() vai verificar se este utilizador está registado ou não na base de dados, já o verificaDisponibilidadeUser() vai verificar se este nome de utilizador está disponível para ser registado. O inserirStream(), como o próprio nome indica, fará o carregamento da stream, seja imagem ou vídeo, para a base de dados, guardando-a. O verificaCriador(), vai verificador o dono da stream, deleteStreamURL() fará que se o utilizador desejar apagar alguma das suas streams, esta seja apagada na base de dados e o buscaURLs(), serve para quando o utilizador pretende ter acesso para visualizar alguns das suas streams guardadas na base de dados. O guardaFrame() é um processo que utiliza alguns métodos de OpenCV no sua função para guardar frames, sob a forma de imagens da stream desejada pelo utilizador na base de dados. O obtemFrames() serve para mostrar todas os frames guardados na base de dados de um determinado utilizador. Já o obtemDadaFrame() acaba por ser um processo um pouco mais específico, já que vai buscar uma frame específica que esteja guardada na base de dados que o utilizador possa pedir, onde o resultado mostrado no site vai aparecer com as horas a que o frame foi tirado no formato UTC. O removeFrame() é como o próprio nome diz serve para apagar as frames da base de dados que o utilizador ache que deva apagar e o seUserDonoFrame() verifica se o utilizador em questão é o proprietário destas frames que pretende visualizar, processar ou apagar. Para o processo dos vídeos, a estrutura é praticamente idêntica, o guardaVideo() vai guardar o vídeo processado na base de dados, o obtemVideo() para mostrar o vídeo no site para o utilizador, removeVideo() para apaga-lo da base de dados e o seUserDonoVideo() vai verificar se o utilizador em questão é o dono do vídeo para assim poder fazer as alterações que pretende.

Por fim, na parte de Python da aplicação, chega-nos a videreapp.py. Este é o ficheiro principal do todo o projeto, já que é o que se conecta com praticamente todos os outros ficheiros para o funcionamento do código. Ele inicializa-se com uma string que mantém todos os utilizadores em processo no servidor, ou seja, vai mostrar os utilizadores ativos no momento que estejam no site. Entrando na sessão, a primeira coisa que se faz será obter as câmaras, os vídeos e observar a existência da câmara dos utilizadores que estejam ativos. Estes três processos entram numa lógica para obter os dados do utilizador, que é usado para obter o URL da câmara e a thumbnail, para obter o objeto da câmara e o vídeo. Também aqui se encontra a função chamada ApagaCamara() que serve para apagar uma câmara se o utilizador não a quiser mais usar. De seguida, temos as classes usadas, Utilizador, Camara, Imagem e Video. A classe Utilizador é a mais simples, tendo apenas como atributo o ID do mesmo. Sendo o utilizador, ele terá a capacidade de inserir câmaras, vídeos ou imagens no site e o seu processamento e a verificação da ligação da câmara, se ela está disponível ou não. Já a Camara tem nos seus atributos o link URL da mesma, o vídeo que a câmara está a transmitir, o ID do utilizador, o nome da câmara e os filtros que estejam a ser utilizados. A classe verifica primeiro se o utilizador em questão tenha CUDA instalado ou não, seguindo-se de obter frames, criação de thumbnails ou ser apagada, a parte de processamento é a mais longa, já que precisa de verificar vários aspetos durante a sua captura, entre ajustes da imagem, o grau de confiança sobre a imagem, verificar se o objeto que está a ser capturado está presente nos filtros de deteção ou não, impedir que o texto do quadrado de deteção fique fora do ecrã e conversão para uma imagem em formato JPG, para no fim guardar os resultados na base de dados. A classe Imagem segue exatamente o mesmo processo para a captura de imagem que a Camara, porém sem a possibilidade da retirada da câmara ou detetar se a câmara esteja ativa, já que não será preciso. Apresenta menos atributos, tendo apenas o ID, os filtros e a imagem em si. Por fim, a classe do Video, segue a mesma lógica das outras duas classes que foram anteriormente referidas para o processamento, com os mesmos atributos da classe Imagem (ID, filtros e vídeos invés de imagem), com a novidade de haver um bloco if para quando não houver mais frames no vídeo, para terminar o processo. Nesta classe, acaba por haver uma função que mostra o progresso do processamento do vídeo no site.

# Conclusões

Com o prazo de entrega do projeto a chegar ao fim, eis as funcionalidades desenvolvidas pelo grupo ao longo do trabalho realizado durante o semestre.

Os testes realizados por nós, demonstram que o código que foi construído, é bastante positivo. A aplicação consegue registar e guardar utilizadores, inicializar câmaras, inserir imagens, vídeos e apagá-los com sucesso através da conectividade entre o site e a base de dados. A implementação do algoritmo de Deep Learning YOLOv3/Darknet apresenta resultados muito positivos no site, onde o algoritmo do YOLO deteta com boa precisão a deteção de objetos que consegue reconhecer. A aplicação também permite filtrar que objetos é que o utilizador pretende detetar, assim como alterar o seu brilho e o seu contraste. Os resultados com o Tiny YOLO são bastante medíocres nos testes que realizamos. Apesar de ser uma solução para máquinas mais fracas a precisão do mesmo é sempre relativamente baixa, podendo até reconhecer de forma errada o objeto e “atrofia-se” um pouco quando há vários objetos no mesmo frame. Por isso, recomenda-se utilizar o YOLO normal se possível para correr o site.

Os objetivos principais enunciados pelo enunciado do projeto foram alcançados na sua maioria. O programa consegue detetar objetos em tempo real, independemente da sua fonte de origem, apresenta os resultados da classificação de objetos no próprio feed do vídeo ou da imagem e está a ser guardado os resultados numa base de dados para consulta posterior. Porém, um objetivo pedido, infelizmente, não foi alcançado. A utilização do Azure. Conforme já referido antes no relatório, tivemos o problema da situação do plafond oferecido pela UAL que acabou por ser consumida muito rapidamente, chegando ao ponto de ser impossível continuar o projeto na plataforma, que mesmo tendo sido discutido algumas vezes com a Leitek e com o professor Mário Marques da Silva este problema, e o suporte que nos deram para encontrar uma solução para este problema, não se conseguiu resolver de todo e tivemos que seguir o projeto na utilização das nossa máquinas locais. Infelizmente, é uma situação alheia a nós, que gostávamos de ter continuado a trabalhar no Azure mas não foi possível e que teria sido um benefício para nós, pois chega às vezes ao ponto que as nossas máquinas começavam a piorar a performance por não conseguir aguentar tanto processamento a ocorrer.

Ao longo do projeto foram encontradas algumas dificuldades que tivemos pelo caminho que se conseguiram ser resolvidas. Sem dúvida dois dos mais marcantes foram a compilação do CUDA com OpenCV, foi um processo longo que demorou vários dias e bastante pesquisa para conseguir ultrapassar este obstáculo. Outra dificuldade foi o tal bug que referimos anteriormente das duas câmaras em simultâneo do OpenCV que supostamente tinha sido corrigido na versão 3, mas lá descobrimos que era devido ao nome das câmaras. Foi preciso bastante martelada para corrigirmos este problema.

Estes meses em que realizamos o projeto, acabamos por ganhar bastante aprendizagem sobre conceitos de IA, principalmente focados em Deep Learning. Ao realizarmos este projeto, permitiu-nos verificar de uma forma mais perto como é que esta área da IA funciona e aprender a usá-la para o que nos foi pedido. Foi uma experiência muito interessante ver um algoritmo andar a detetar objetos, em especial de pessoas, independemente da sua resolução ou área que tivesse a ser capturada, fosse um lugar muito movimentado ou uma divisão das nossas casas com uma eficácia bastante interessante. Ao desenvolvermos este projeto para a Leitek, deu para fazer um programa que pode acabar por ser comercializado, seguindo uma perspetiva de Software as a Service que permite salvar pessoas de um incêndio florestal, por exemplo. Sem dúvida existe bastante potencial nesta tecnologia que está agora a começar a dar os seus primeiros passos no mundo, onde pode ser usada para uso positivo ou negativo, como iremos referir no ponto a seguir sobre o Trabalho Futuro. O alcance de criar aplicações com uso de Deep Learning está ao alcance de todos, para isso, basta alguns conhecimentos de programação e alguma pesquisa

Num ponto de vista mais pessoal, para nós, além de ter permitido mais conhecimento sobre Deep Learning, permitiu trabalhar com algumas ferramentas que não tínhamos usado ainda, melhorando assim os nossos conhecimentos sobre os mesmos, em especial o Azure, que enquanto foi possível utilizá-lo foi uma experiência bastante divertida descobrir tudo o que o Azure tem e o que pode fazer. Foi um projeto demoroso, no qual gastamos bastante tempo para realizar todos os objetivos e corrigir os bugs que apareceram ao longo do desenvolvimento, mas apesar de toda esta luta, valeu a pena produzir este trabalho e sentimos, no fundo, contentes com os resultados obtidos.

O capítulo a seguir, Trabalho Futuro, serve para falar daquilo que achamos que terá de ser feito no futuro sobre este projeto que desenvolvemos, existe ainda alguns passos que terão de ser resolvidos para chegar à perspetiva futura referida no enunciado, para o programa virar um Software as a Service.

# Trabalho futuro

Existe algum trabalho futuro que poderá ter de ser realizado. O primeiro ponto, é como já referido anteriormente no relatório algumas vezes, seria transitar o programa de máquina local para o Microsoft Azure. Tivemos bastantes dificuldades em trabalhar com o Azure, não tanto por causa de correr o programa em si lá, mas sim por causa das licenças que as nossas contas da UAL que fizeram que o plafond oferecido tivesse sido esgotado muito rapidamente, impedindo a continuação da realização do projeto na plataforma.

Este problema das contas no Azure foi registado em mais grupos para além do nosso, por isso, caso se decida realizar projetos com Azure nos próximos anos, a UAL e os parceiros, terão de investigar mais a fundo este problema que ocorreu, para evitar que haja complicações de novo por causa do Azure.

No nosso caso, por exemplo, o Azure beneficiaria o nosso programa com o suporte do mesmo e de GPUs dedicadas, onde o desempenho do funcionamento teria um aumento significativo, onde o passo a seguir seria tecnologia de reconhecimento de objetos e facial por GPUs.

Entre as nossas reuniões com a Leitek, o plano da empresa com o nosso programa, não será para comercialização para já e que pretende usar para áreas de Proteção Civil, como, por exemplo, detetar pessoas que estejam perdidas num incêndio. Caso a Leitek decida continuar a parceria com a UAL nos próximos anos e tenha planos para comercializar o produto, terá de haver algumas mudanças para estar pronto para o mercado.

Para começar, o site terá de obedecer às normas da União Europeia sobre o Regulamento Geral de Proteção de Dados, que após algumas questões que tivemos na reunião, não foi implementado. Esse será, de imediato, o primeiro ponto a ser feito caso venha a ser comercializado. O segundo ponto será, finalmente, o programa correr no Azure sem as restrições que ocorreram. Após este passo, sugere-se a ideia de expandir para mais plataformas o uso do programa da deteção de objetos, como uma aplicação para telemóveis iOS e Android onde o funcionamento todo no programa para estes dispositivos seja fornecido totalmente pelo Azure. E quem sabe até mesmo Smart TVs não possam vir a ter um dia o programa a correr no futuro, ainda que longínquo.

A ponto de curiosidade, os conceitos de Deep Learning e programas com foco em deteção de objetos, apresentam um potencial tremendo, que pode ser usado para o bem, como são os planos da Leitek, mas também podem ser usados para métodos mais duvidosos. Recentemente, a empresa chinesa Tencent, decidiu criar um programa de deteção facial de caras, para detetar se a pessoa é menor ou não. O objetivo do programa consiste em controlar o tempo que as crianças estão a jogar videojogos durante a noite, de forma mais precisa entre as 22 horas da noite e as 8 horas da manhã. Segundo a Tencent, o sistema de verificação facial vai ajudar a monitorizar jogadores e a detetar indivíduos que gastem “um espaço significativo de tempo” online à noite. O sistema fará que os menores de idade estejam limitados a 90 minutos de jogo à noite durante a semana e três horas aos fins de semana e feriados. O sistema já está a ser aplicado em, pelo menos, 60 jogos na China, com intenções de expandir para mais [1].

Isto demonstra que existe ainda várias maneiras de implementar algoritmos de deteção de objetos, que, provavelmente, não fazemos ainda sequer ideia de como inseri-la, que pode ser um benefício para todos, como pode não ser tanto assim e meter em perigo algumas liberdades do Homem. Sem dúvida, que será um assunto para discutir dentro de algumas décadas.

# Bibliografia

[1] <https://autonoma.pt/departamento/>, acedido a 10 de julho de 2021.

[2] “A Autónoma”, acedido por <https://autonoma.pt/a_autonoma/a-autonoma/> em 10 de julho de 2021.

[3] <https://grupoceu.pt/> , acedido a 10 de julho de 2021.

[4] “LEITEK INNOVATIVE SOLUTIONS”, acedido por <https://www.leitek.co/>, em 10 de julho de 2021.

[5] “QUEM SOMOS”, acedido por <https://www.leitek.co/leitek> , em 10 de julho de 2021.

[6] “Deep Learning. O que é e qual sua importância?”, acedido por <https://www.sas.com/pt_br/insights/analytics/deep-learning.html> , em 18 de maio de 2021.

[7] “OpenCV”, acedido por <https://en.wikipedia.org/wiki/OpenCV> , em 18 de maio de 2021.

[8] <https://opencv.org/> , acedido em 18 de maio de 2021.

[9] “Microsoft Azure”, acedido por <https://en.wikipedia.org/wiki/Microsoft_Azure> , em 18 de maio de 2021.

[10] “Produtos do Azure”, acedido por <https://azure.microsoft.com/pt-pt/services/> , em 18 de maio de 2021.

[11] “PostgreSQL”, acedido por <https://en.wikipedia.org/wiki/PostgreSQL> , em 18 de maio de 2021.

[12] “Flask (web framework)”, acedido por <https://en.wikipedia.org/wiki/Flask_(web_framework)> , em 19 de maio de 2021.

[13] “Python (programming language)”, acedido por <https://en.wikipedia.org/wiki/Python_(programming_language)> , em 19 de maio de 2021.

[14] “SQLAlchemy”, acedido por <https://en.wikipedia.org/wiki/SQLAlchemy> , em 19 de maio de 2021.

[15] “The Python SQL Toolkit and Object Relational Mapper”, acedido por <https://www.sqlalchemy.org/> , em 19 de maio de 2021.

[16] “Machine leraning: o que é e para que serve?”, CCG – Centro de Computação Gráfica, 13 de dezembro de 2017, acedido por <https://www.ccg.pt/machine-learning-o-que-e/> , em 25 de maio de 2021.

[17] “CUDA”, acedido por <https://en.wikipedia.org/wiki/CUDA> , em 25 de maio de 2021.

[18] “YOLO: Real-Timr Object Detection”, acedido por <https://pjreddie.com/darknet/yolo/> , em 25 de maio de 2021.

[19] Adriane Rosebrock, “YOLO and Tiny-YOLO object detection ont the Raspberry Pi and Movidius NCS”, pyimagesearch, 27 de Janeiro de 2020, acedido por <https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/>, em 25 de maio de 2021.

[20] “Bootstrap (front-end framework), acedido por <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>, em 31 de maio de 2021.

[21] “HTML”, acedido por <https://en.wikipedia.org/wiki/HTML> em 1 de junho de 2021.

[22] “CSS”, acedido por <https://en.wikipedia.org/wiki/CSS>, em 1 de junho de 2021.

[23] “JavaScript”, acedido por <https://en.wikipedia.org/wiki/JavaScript> , em 1 de junho de 2021.

[24] “Ubuntu”, acedido por <https://en.wikipedia.org/wiki/Ubuntu> , em 9 de julho de 2021.

[25] “TeamViewer”, acedido por <https://en.wikipedia.org/wiki/TeamViewer> , em 9 de julho de 2021.

[26] “TeamViewer for Windows”, acedido por <https://www.teamviewer.com/en/download/windows/>, em 9 de julho de 2021.

[27] “SQL”, acedido por <https://pt.wikipedia.org/wiki/SQL>, em 9 de julho de 2021.

[28] “Using NGC with Azure Setup Guide”, 20 de novembro de 2019, acedido por <https://docs.nvidia.com/ngc/ngc-azure-setup-guide/launching-gpu-cloud-instance-from-azure-portal.html> em 10 de julho de 2021.

[29] TheCodingBug, “Build and Install OpenCV With CUDA GPU Support on Windows 10 | OpenCV 4.5.1 | 2021”, 21 de janeiro de 2021, acedido por <https://www.youtube.com/watch?v=YsmhKar8oOc> em 11 de julho de 2021.

[30] raulqf, “Install\_OpenCV4\_CUDA11\_CUDNN8.md” , acedido por <https://gist.github.com/raulqf/f42c718a658cddc16f9df07ecc627be7> em 11 de julho de 2021.

[31] vivekpatani, “Makefile 160: recipe for target ‘all’ failed make: \*\*\* [all] Error 2”, 11 de novembro de 2016, GitHub, acedido por <https://github.com/opencv/opencv/issues/7652> em 11 de julho de 2021.

[32] nerdfever, “[WARN:0] terminating async callback”, 4 de maio de 2019, GitHub, acedido por <https://github.com/opencv/opencv-python/issues/198> em 11 de julho de 2021.

[33] Jerome Heath, “Midnight patrol: Tencent activates facial recognition system to prevent children gaming at night in China”, DOT ESPORTS, 6 de julho de 2021 às 9:17, Acedido por <https://dotesports.com/general/news/tencent-activates-facial-recognition-system> em 12 de julho de 2021.

[34] “Engine Configuration”, 28 de junho de 2021, SQLAlchemy, acedido por <https://docs.sqlalchemy.org/en/14/core/engines.html> em 12 de julho de 2021.

[35] “Create, read, update and delete”, acedido por <https://en.wikipedia.org/wiki/Create,_read,_update_and_delete> em 12 de julho de 2021.

[36]”Class BCrypt”, acedido por <https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCrypt.html> em 12 de julho de 2021.

[37] “Modular Applications with Blueprints”, acedido por <https://flask.palletsprojects.com/en/2.0.x/blueprints/> em 9 de julho de 2021.

# Créditos

Elementos do desenvolvimento do Projeto e do Relatório:

- Bruno Silva, nº30003696;

- Diogo Mendes, nº30003865;

- Reinaldo Coelho, nº30003670;

- Ricardo Cardoso, nº30002299;

Alunos do curso de Engenharia Informática da Universidade Autónoma de Lisboa, também conhecido por “UALunos”.

P.S- O termo “UALunos” foi criado pelo nosso colega Ricardo Cardoso, que insistiu que colocássemos o termo no relatório e não se calou até ser introduzido. O termo é um trocadilho entre a sigla “UAL” com a palavra “alunos”. Caso algum professor queira utilizar este termo e achou piada, estão à vontade para o usar no futuro, têm a nossa autorização para o uso.

Por parte da Universidade Autónoma de Lisboa:

Os professores da Unidade Curricular de Laboratório de Projeto:

- Adriana Fernandes;

- Gonçalo Valadão;

- Mário Marques da Silva;

- Valéria Pequeno;

O professor orientador deste projeto:

- Mário Marques da Silva.

Por parte da LEITEK INNOVATIVE SOLUTIONS:

Os elementos da empresa que estiveram presentes para dar suporte ao desenvolvimento do projeto:

- Artur Pedro Morais, Engenheiro;

- Cleo Dinis Leite, CEO da empresa;

- Rita Capela;

- Tiago Mendes, programador;

**©**2021 Universidade Autónoma de Lisboa / LEITEK INNOVATIVE SOLUTIONS. As duas entidades são proprietárias deste projeto e deste relatório. A comercialização ou distribuição deste projeto é proibida sem o consentimento das duas entidades envolvidas. Todos os direitos reservados. Este projeto foi realizado no âmbito da Unidade Curricular de Laboratório de Projeto no curso de Engenharia Informática da Universidade Autónoma de Lisboa no ano letivo 2020/2021.

Muito obrigado por terem lido o relatório até ao fim. Espero que tenham gostado de ler sobre esta incrível jornada sobre Reconhecimento/Classificação de Alvos por Algoritmos de Deep Learning em OpenCV através de Interface Web. De seguida, apresenta-se em anexo, o código realizado no projeto na parte de Python.

# Anexo 01 – Código do projeto

app.py

import json

from flask import Flask, render\_template, Response, request, redirect, url\_for, session, send\_file, flash, \

send\_from\_directory

import config

import utilizador

import videredb

from camara import camara\_pagina

from galeria import galeria\_pagina

from carregarFicheiros import carregarFicheiros\_pagina

from novaCamara import novaCamara\_pagina

app = Flask(\_\_name\_\_)

app.register\_blueprint(camara\_pagina)

app.register\_blueprint(galeria\_pagina)

app.register\_blueprint(novaCamara\_pagina)

app.register\_blueprint(carregarFicheiros\_pagina)

app.static\_folder = 'static'

app.secret\_key = config.chaveSession

@app.route('/', methods=["POST", "GET"])

def login():

if "user\_id" in session:

return redirect(url\_for("painel"))

if request.method == "POST":

user\_nome = request.form["userNome"].strip()

passworduser = request.form["userSenha"]

if " " in passworduser or " " in user\_nome:

flash("Credenciais inválidas")

return redirect(url\_for("login"))

user\_id = videredb.verificaUtilizador(user\_nome, passworduser)

if user\_id is not None:

if user\_id not in utilizador.UTILIZADORES\_ATIVOS:

u = utilizador.Utilizador(user\_id)

utilizador.UTILIZADORES\_ATIVOS[user\_id] = u

session["user\_nome"] = user\_nome

session["user\_id"] = user\_id

return redirect(url\_for("painel"))

else:

flash("Credenciais inválidas")

return redirect(url\_for("login"))

return render\_template("login.html") # Se for GET

@app.route('/registo', methods=["POST", "GET"])

def registo():

if "user\_id" not in session:

if request.method == "POST":

user\_nome = request.form["nomeUser"].strip()

senha = request.form["senhaUser"]

if not len(user\_nome) > 0:

flash("Nome de utilizador está vazio.")

return redirect(url\_for("registo"))

if " " in senha:

flash("Uma senha não pode conter espaços em branco")

return redirect(url\_for("registo"))

if " " in user\_nome:

flash("Um nome não pode conter espaços em branco")

return redirect(url\_for("registo"))

if not videredb.verificaDisponibilidadeUser(user\_nome):

flash("Este utilizador já existe")

return redirect(url\_for("registo"))

if not len(senha) > 5:

flash("A senha precisa de ter mais que 5 caracteres.")

return redirect(url\_for("registo"))

videredb.inserirUtilizador(user\_nome, senha)

return redirect(url\_for("login"))

else:

return render\_template("registo.html")

else:

return redirect(url\_for("painel"))

@app.route('/painel', methods=["POST", "GET"]) # Painel de controlo do utilizador

def painel():

if "user\_id" not in session: # Se utilizador não tiver ligado

return redirect(url\_for("login"))

vds\_id = []

if request.method == "POST":

if "novacmr" in request.form: # Inicia um nova camara

return redirect(url\_for("novaCamara.novaCamaraStream"))

elif request.method == "GET":

if session["user\_id"] in utilizador.UTILIZADORES\_ATIVOS:

vds\_id = utilizador.UTILIZADORES\_ATIVOS.get(session["user\_id"]).obtemCamarasLigacao()

return render\_template("painel.html", vds\_id=vds\_id)

@app.route('/pg<string:feed>')

def progressoProcessoVideo(feed):

if "user\_id" in session:

try:

dado = utilizador.obtemVideo(session["user\_id"], feed).progresso()

return json.dumps({'progresso': dado}), 200, {'ContentType': 'application/json'}

except:

return json.dumps({'progresso': -1}), 200, {'ContentType': 'application/json'}

else:

return redirect(url\_for("login"))

# ==============================================================

@app.route('/video<string:feed>') # Pagina de resultado de video em ficheiro procesado

def VideoPaginaProcesso(feed):

if "user\_id" in session:

return render\_template("videoResultado.html", feed=feed)

else:

return redirect(url\_for("login"))

@app.route('/vdc<string:feed>') # Video mp4 do ficheiro carregado processado

def videoProcessadoResultado(feed):

if "user\_id" in session:

return send\_from\_directory(config.pastaVideos, feed + '.webm')

else:

return redirect(url\_for("login"))

# ==============================================================

@app.route('/vd<string:feed>') # Video para transmissão das camaras

def transmitirImagem(feed):

# Obtem video de uma camara de um utilizador, de momento o video é privado

if "user\_id" in session:

return Response(utilizador.obtemCrm(session["user\_id"], feed).obtemFrame(),

mimetype='multipart/x-mixed-replace; boundary=frame')

else:

return redirect(url\_for("login"))

@app.route('/tb<string:feed>') # Obtem thumbnail

def transmitirthumbnail(feed):

if "user\_id" in session:

tb = utilizador.obtemCrm(session["user\_id"], feed).obtemThumbnail()

if tb is None: # Se não houver frames disponiveis, retorna uma imagem comum de loading

return redirect(url\_for('static', filename='img/eyetumb.gif'))

else:

return Response(tb, mimetype='multipart/x-mixed-replace; boundary=frame')

else:

return redirect(url\_for("login"))

camara.py

import json

from flask import request, Response, Blueprint, render\_template, url\_for, session

from werkzeug.utils import redirect

import app

import dataset

import utilizador

from videredb import deleteStreamURL, verificaCriador

camara\_pagina = Blueprint('camara', \_\_name\_\_, template\_folder='templates')

@camara\_pagina.route('/cm<string:feed>', methods=["POST", "GET"])

def janelaCamara(feed):

if "user\_id" in app.session and verificaCriador(session["user\_id"], feed): # Verifica se o Uitilizador é dono da stream e pode a ver

if request.method == "POST":

if "apagarCmr" in request.form:

utilizador.ApagaCamara(session["user\_id"], feed)

deleteStreamURL(feed, session["user\_id"])

return redirect(url\_for("painel"))

r = request.json

if r['tipo'] == "b":

per = r['valor']

brilho = round(int(per) \* 255 / 100)

utilizador.obtemCrm(session["user\_id"], feed).brilho = brilho

elif r['tipo'] == "c":

per = r['valor']

contraste = round(int(per) \* 255 / 100)

utilizador.obtemCrm(session["user\_id"], feed).contraste = contraste

elif r['tipo'] == "f":

filtros\_selecionados = r['valor']

utilizador.obtemCrm(session["user\_id"], feed).filtros = list(map(int, filtros\_selecionados))

return Response(status=200)

elif request.method == "GET":

# Obtem Dados

cmr = utilizador.obtemCrm(session["user\_id"], feed)

return render\_template("camera.html", vd\_id=feed, camara\_nome=cmr.nome, brilho=cmr.brilho,

contraste=cmr.contraste, classes=dataset.classes,

selecionados=json.dumps(cmr.filtros))

return redirect(url\_for("login"))

carregarFicheiros.py

import os

import uuid

import cv2

import numpy

from flask import Blueprint, request, render\_template, url\_for, flash, session, Response

from werkzeug.utils import redirect

import dataset

import utilizador

carregarFicheiros\_pagina = Blueprint('carregaFicheiro', \_\_name\_\_, template\_folder='templates')

@carregarFicheiros\_pagina.route('/carregar', methods=["POST", "GET"])

def carregaFicheiros():

if "user\_id" in session:

filtroObjetos = []

if request.method == "POST":

if 'file' not in request.files:

flash('Pedido feito não possui ficheiro.')

return redirect(url\_for("carregaFicheiro.carregaFicheiros"))

file = request.files['file']

if file.filename == '':

flash('Nenhum ficheiro selecionado')

return redirect(url\_for("carregaFicheiro.carregaFicheiros"))

if file and not file.filename.lower().endswith(('.png', '.jpg', '.jpeg', '.mp4')):

flash('Formato do ficheiro não é suportado. Use apenas PNG, JPG, JPEG ou MP4')

return redirect(url\_for("carregaFicheiro.carregaFicheiros"))

for i in dataset.classes.keys():

if request.form.get(str(i)) == "on":

filtroObjetos.append(i)

if len(filtroObjetos) == 0:

flash("Não há nenhum filtro selecionado para iniciar a detectação.")

return redirect(url\_for("carregaFicheiro.carregaFicheiros"))

if not file.filename.lower().endswith('mp4'): # É uma imagem

frame = cv2.imdecode(numpy.frombuffer(request.files['file'].read(), numpy.uint8), cv2.IMREAD\_UNCHANGED)

frame = cv2.cvtColor(frame, cv2.COLOR\_RGBA2RGB)

link = utilizador.UTILIZADORES\_ATIVOS[session["user\_id"]].CriaProcessoImagem(frame, filtroObjetos)

return redirect("/verimagem/" + link)

else: # É um video

vid = str(uuid.uuid1()).replace("-", "") # Gera um nome para o ficheiro que vai servir como ID

file.save(os.path.join(vid + ".mp4"))

utilizador.UTILIZADORES\_ATIVOS[session["user\_id"]].CriaProcessoVideo(vid, filtroObjetos)

return redirect("/video" + vid)

else: # GET

return render\_template("carregarFicheiro.html", classes=dataset.classes)

else:

return redirect(url\_for("login"))

config.py

import os

# Port e flag de debug da aplicação

port = 5000

debugFlag = True

# Pastas

pastaFrames = "frames"

pastaVideos = "videos"

# Parametros do yolo

yoloPath = "yolo/yolov3.cfg"

yoloPathWeights = "yolo/yolov3.weights"

yoloDataset = "yolo/coco.names"

# basi di dados

database = 'postgresql+psycopg2://postgres:admin@localhost:5432/Videre'

# Sessão

chaveSession = os.urandom(24)

def pastas():

if not os.path.exists(os.path.join("videos")):

os.makedirs(os.path.join("videos"))

if not os.path.exists(os.path.join("frames")):

os.makedirs(os.path.join("frames"))

dataset.py

import config

classes = {} # Classes de nomes dos objetos (id, nome)

classes\_cores = {}

cores = [(255, 0, 0), (255, 163, 0), (255, 39, 220), (0, 39, 220), (0, 248, 85), (109, 51, 64), (253, 250, 247), (1, 142, 41)]

with open(config.yoloDataset, "r", encoding="utf-8") as f: # Coco dataset é o dataset do yolo que contem os 80 objetos

apontador\_cor = -1

for c, i in enumerate(f):

classes[c] = i.strip()

if c % 10 == 0: # A cada 10 objetos mete uma cor diferente (8 cores por 80 objetos)

apontador\_cor += 1

classes\_cores[c] = cores[apontador\_cor]

classes = dict(sorted(classes.items(), key=lambda item: item[1]))

galeria.py

from flask import request, Blueprint, render\_template, url\_for, send\_from\_directory

from werkzeug.utils import redirect

import app as main

import config

import dataset

import videredb

galeria\_pagina = Blueprint('galeria', \_\_name\_\_, template\_folder='templates')

@galeria\_pagina.route('/galeria', methods=["POST", "GET"]) # obtem a galeria

def janelaGaleria():

if "user\_id" in main.session:

if request.method == "POST":

pass

fotos = videredb.obtemFrames(main.session["user\_id"])

videos = videredb.obtemVideo(main.session["user\_id"])

return render\_template("galeria.html", classes=list(dataset.classes.values()), fotos=fotos, videos=videos)

return redirect(url\_for("login"))

@galeria\_pagina.route("/foto/<img>")

def obtemFrameGaleria(img):

if "user\_id" in main.session:

return send\_from\_directory(config.pastaFrames, img + ".png")

return redirect(url\_for("login"))

@galeria\_pagina.route('/vervideo/<string:feed>', methods=["POST", "GET"]) # Mostra Video da galaria

def VideoGaleriaVer(feed):

if "user\_id" in main.session:

if request.method == "GET":

dono = videredb.seUserDonoVideo(feed, main.session["user\_id"])

return render\_template("verVideo.html", feed=feed, dono=dono)

elif request.method == "POST":

if "apagavid" in request.form: # Apaga Video

videredb.removeVideo(feed, main.session["user\_id"])

return redirect(url\_for("galeria.janelaGaleria"))

else:

return redirect(url\_for("login"))

@galeria\_pagina.route('/verimagem/<img>', methods=["POST", "GET"])

def paginaVerImagem(img):

if "user\_id" in main.session:

if request.method == "GET":

dono = videredb.seUserDonoFrame(img, main.session["user\_id"])

data = videredb.obtemDadaFrame(main.session["user\_id"], img)

return render\_template("verImagem.html", img=img, data=data, dono=dono)

elif request.method == "POST":

if "apagaimg" in request.form: # Apaga Imagem

if not videredb.removeFrame(img, main.session["user\_id"]):

return redirect(url\_for("login"))

return redirect(url\_for("galeria.janelaGaleria"))

return redirect(url\_for("login"))

novaCamara.py

from flask import Blueprint, request, render\_template, url\_for, flash

from werkzeug.utils import redirect

import app as main

import dataset

import utilizador

import videredb

novaCamara\_pagina = Blueprint('novaCamara', \_\_name\_\_, template\_folder='templates')

@novaCamara\_pagina.route('/novacamara', methods=["POST", "GET"]) # obtem a galeria

def novaCamaraStream():

if "user\_id" in main.session:

if request.method == "POST":

if "novacmr" in request.form:

if main.session["user\_id"] in utilizador.UTILIZADORES\_ATIVOS:

linkCamara = request.form["linkCamara"].strip()

nomeCamara = request.form["nomeCamara"].strip()

# if linkCamara == "TESTE": linkCamara = "video.mp4" # USADO PARA TESTES

filtroObjetos = []

for i in dataset.classes.keys():

if request.form.get(str(i)) == "on":

filtroObjetos.append(i)

erros = 0

if not linkCamara:

flash("Link da câmera vazio. É necessário inserir um.")

erros += 1

if not nomeCamara:

flash("Nome da câmera vazio. É necessário inserir um.")

erros += 1

if len(filtroObjetos) == 0:

flash("Não há nenhum objeto selecionado para instruir a detectação.")

erros += 1

if utilizador.ObtemExistenciaCmr(main.session["user\_id"], nomeCamara):

flash("Já existe uma câmera com este nome.")

erros += 1

if erros > 0: return redirect(url\_for("novaCamara.novaCamaraStream"))

utilizador.UTILIZADORES\_ATIVOS[main.session["user\_id"]].CriaCamara(linkCamara, nomeCamara, filtroObjetos)

return redirect(url\_for("painel"))

else:

return render\_template("novaCamara.html", classes=dataset.classes)

return redirect(url\_for("login"))

run.py

import videredb

from app import app as VidereApp

import config

if \_\_name\_\_ == "\_\_main\_\_":

config.pastas()

videredb.apagaTabelaStreams() # Remove todos os dados em relações ás streams sempre que o servidor inicia

VidereApp.run(port=config.port, debug=config.debugFlag)

utilizador.py

import os

import threading

import time

import uuid

import cv2

import numpy as np

import dataset

import config

import videredb

UTILIZADORES\_ATIVOS = {} # Lista de mantem todos os utilizadores em processo no servidor

# Obtem dados do utilizador, usado para obter url da camara e thumbnail, obter objeto da camara e video

def obtemCrm(user, vid):

if user in UTILIZADORES\_ATIVOS and vid in UTILIZADORES\_ATIVOS.get(user).camaras:

return UTILIZADORES\_ATIVOS.get(user).camaras.get(vid)

def obtemVideo(user, vid):

if user in UTILIZADORES\_ATIVOS and vid in UTILIZADORES\_ATIVOS.get(user).videos:

return UTILIZADORES\_ATIVOS.get(user).videos.get(vid)

def ObtemExistenciaCmr(nome, nomeCmr):

v = UTILIZADORES\_ATIVOS.get(nome).camaras.values()

for i in v:

if nomeCmr == i.nome:

return True

return False

def ApagaCamara(user, vid):

if user in UTILIZADORES\_ATIVOS and vid in UTILIZADORES\_ATIVOS.get(user).camaras:

UTILIZADORES\_ATIVOS.get(user).camaras.get(vid).imagem.release()

UTILIZADORES\_ATIVOS.get(user).camaras.get(vid).thread.join()

d = UTILIZADORES\_ATIVOS.get(user)

del d.camaras[vid]

class Utilizador:

def \_\_init\_\_(self, id\_u):

self.id = id\_u

self.camaras = {} # vid: objeto

self.videos = {} # vid: objeto

def CriaCamara(self, lnk, nome, filtros):

vid = str(uuid.uuid1()).replace("-", "") # Gera o id do video que também é usado para url para aceder via web

cmr = Camara(lnk, vid, self.id, nome, filtros)

videredb.inserirStream(self.id, vid)

t = threading.Thread(target=cmr.processa)

cmr.thread = t

self.camaras[vid] = cmr

t.start()

def CriaProcessoImagem(self, imagem, filtros):

img = Imagem(self.id, filtros, imagem)

nome = img.processar()

return nome

# threading.Thread(target=img.processar).start()

def CriaProcessoVideo(self, video\_id, filtros):

video = Video(self.id, filtros, video\_id)

t = threading.Thread(target=video.processa)

video.thread = t

self.videos[video\_id] = video

t.start()

def obtemCamarasLigacao(self):

cmrs = {}

for i in self.camaras:

cmrs[self.camaras[i].nome] = i

return cmrs

class Camara:

def \_\_init\_\_(self, lnk, vid, id\_user, c\_nome, filtros):

self.filtros = filtros

self.nome = c\_nome

self.id = vid

self.id\_user = id\_user

self.imagem = cv2.VideoCapture(lnk, 0)

self.net = cv2.dnn.readNet(config.yoloPath, config.yoloPathWeights)

self.framecurrente = None

self.tempoInicial = time.time() # Tempo inicial da contagem para guardar a proxima frame da BD

self.tempoPassado = 0

self.thread = None

self.retoques = True

self.brilho = 0

self.contraste = 0

# Inicia CUDA, se utilizador não suportar, estas linhas são ignoradas

if cv2.cuda.getCudaEnabledDeviceCount() > 0:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

else:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_DEFAULT)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)

def obtemFrame(self):

while True:

if self.framecurrente is None:

img = cv2.imread(

"static\img\espera.png") # Caso o frame não esteja disponivel então mete uma imagem de espera

\_, buffer = cv2.imencode('.jpg', img)

frame = buffer.tobytes()

yield b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n'

else:

\_, buffer = cv2.imencode('.jpg', self.framecurrente)

frame = buffer.tobytes()

yield b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n'

def obtemThumbnail(self):

if self.framecurrente is None:

return None

else:

\_, buffer = cv2.imencode('.jpg', self.framecurrente)

frame = buffer.tobytes()

return b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n'

def \_\_del\_\_(self):

self.imagem.release()

def processa(self):

while True:

ativo, frame = self.imagem.read()

if not ativo: break

# Ajusta imagem

img = np.int16(frame)

img = img \* (self.contraste / 127 + 1) - self.contraste + self.brilho

img = np.clip(img, 0, 255)

frame = np.uint8(img)

tamanho = frame.shape

layer\_names = self.net.getLayerNames()

outputlayers = [layer\_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]

altura = frame.shape[0]

comprimento = frame.shape[1]

blob = cv2.dnn.blobFromImage(frame, 1 / 255, (128, 128), swapRB=True, crop=False)

self.net.setInput(blob)

processados = self.net.forward(outputlayers)

class\_ids = []

confidences = [] # Grau de confiança sobre a imagem

caixas = []

for p in processados:

for ObjetoApanhado in p:

pontuacoes = ObjetoApanhado[5:]

class\_id = np.argmax(pontuacoes)

if class\_id not in self.filtros: # Este objeto detetado não está presente nos filtros para detetar

continue

certeza = pontuacoes[class\_id]

if certeza > 0.5:

# Obtem posição (eu não percebo a magia negra que o net.forward faz, mas as posições das coisas estão ai)

centroX = int(ObjetoApanhado[0] \* comprimento)

centroY = int(ObjetoApanhado[1] \* altura)

c = int(ObjetoApanhado[2] \* comprimento)

a = int(ObjetoApanhado[3] \* altura)

x = int(centroX - c / 2)

y = int(centroY - a / 2)

caixas.append([x, y, c, a])

confidences.append(float(certeza))

class\_ids.append(class\_id)

indexes = cv2.dnn.NMSBoxes(caixas, confidences, 0.5, 0.4)

objetos\_captuados\_frame = []

for i in indexes:

objeto\_no\_frame = {}

i = i[0]

caixa = caixas[i]

x = caixa[0]

y = caixa[1]

w = caixa[2]

h = caixa[3]

objeto\_no\_frame["object\_id"] = class\_ids[i]

objeto\_no\_frame["confianca"] = confidences[i]

objeto\_no\_frame["topLeft"] = [x, y]

objeto\_no\_frame["bottomRight"] = [w, h]

objetos\_captuados\_frame.append(objeto\_no\_frame)

label = str(dataset.classes[class\_ids[i]])

cor = dataset.classes\_cores[class\_ids[i]]

cv2.rectangle(frame, (x, y), (x + w, y + h), cor, 2)

# Impede que texto fique fora do ecrã

if y < 40: # se o X estiver 40 pixeis perto do topo da imagem

y += 45 # Baixa 45 pixeis

if x < 5: x += 5

if x + (len(label) + 5) \* 15 > tamanho[1]: # Impede que o texto saia para o lado da imagem

x -= (len(label) + 5) \* 15

# Faz um texto em baixo que serve como contorno

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x - 1, y - 10),

cv2.FONT\_HERSHEY\_DUPLEX, 1, (255, 255, 255), 2, lineType=cv2.LINE\_AA)

# Nome do objeto, texto de cima

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x, y - 10), cv2.FONT\_HERSHEY\_DUPLEX, 1,

cor, 1, lineType=cv2.LINE\_AA)

# Converte para jpg

self.framecurrente = frame # Esta var precisa de estar antes do codigo debaixo para ser funcional no firefox

\_, buffer = cv2.imencode('.jpg', frame)

frame = buffer.tobytes()

# self.framecurrente = frame

# Guarda a imagem na Base Dados

if self.tempoPassado > 5:

if len(indexes) > 0:

self.tempoInicial = time.time()

videredb.guardaFrame(frame, self.id\_user, time.time(), objetos\_captuados\_frame)

self.tempoPassado = 0

else:

self.tempoPassado = time.time() - self.tempoInicial

class Imagem:

def \_\_init\_\_(self, id\_user, filtros, imagem):

self.filtros = filtros

self.id\_user = id\_user

self.imagem = imagem

self.net = cv2.dnn.readNet(config.yoloPath, config.yoloPathWeights)

# Inicia CUDA, se utilizador não suportar, estas linhas são ignoradas

if cv2.cuda.getCudaEnabledDeviceCount() > 0:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

else:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_DEFAULT)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)

def processar(self):

frame = self.imagem

tamanho = frame.shape

layer\_names = self.net.getLayerNames()

outputlayers = [layer\_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]

altura = frame.shape[0]

comprimento = frame.shape[1]

blob = cv2.dnn.blobFromImage(frame, 1 / 255, (512, 512), swapRB=True, crop=False)

self.net.setInput(blob)

processados = self.net.forward(outputlayers)

class\_ids = []

confidences = [] # Grau de confiança sobre a imagem

caixas = []

for p in processados:

for ObjetoApanhado in p:

pontuacoes = ObjetoApanhado[5:]

class\_id = np.argmax(pontuacoes)

if class\_id not in self.filtros: # Este objeto detetado não está presente nos filtros para detetar

continue

certeza = pontuacoes[class\_id]

if certeza > 0.5:

# Obtem posição (eu não percebo a magia negra que o net.forward faz, mas as posições das coisas estão ai)

centroX = int(ObjetoApanhado[0] \* comprimento)

centroY = int(ObjetoApanhado[1] \* altura)

c = int(ObjetoApanhado[2] \* comprimento)

a = int(ObjetoApanhado[3] \* altura)

x = int(centroX - c / 2)

y = int(centroY - a / 2)

caixas.append([x, y, c, a])

confidences.append(float(certeza))

class\_ids.append(class\_id)

indexes = cv2.dnn.NMSBoxes(caixas, confidences, 0.5, 0.4)

objetos\_captuados\_frame = []

for i in indexes:

objeto\_no\_frame = {}

i = i[0]

caixa = caixas[i]

x = caixa[0]

y = caixa[1]

w = caixa[2]

h = caixa[3]

objeto\_no\_frame["object\_id"] = class\_ids[i]

objeto\_no\_frame["confianca"] = confidences[i]

objeto\_no\_frame["topLeft"] = [x, y]

objeto\_no\_frame["bottomRight"] = [w, h]

objetos\_captuados\_frame.append(objeto\_no\_frame)

label = str(dataset.classes[class\_ids[i]])

cor = dataset.classes\_cores[class\_ids[i]]

cv2.rectangle(frame, (x, y), (x + w, y + h), cor, 2)

# Impede que texto fique fora do ecrã

if y < 40: # se o X estiver 40 pixeis perto do topo da imagem

y += 45 # Baixa 45 pixeis

if x < 5: x += 5

if x + (len(label) + 5) \* 15 > tamanho[1]: # Impede que o texto saia para o lado da imagem

x -= (len(label) + 5) \* 15

# Faz um texto em baixo que serve como contorno

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x - 1, y - 10),

cv2.FONT\_HERSHEY\_DUPLEX, 1, (255, 255, 255), 2, lineType=cv2.LINE\_AA)

# Nome do objeto, texto de cima

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x, y - 10), cv2.FONT\_HERSHEY\_DUPLEX, 1,

cor, 1, lineType=cv2.LINE\_AA)

# Converte para jpg

\_, buffer = cv2.imencode('.jpg', frame)

frame = buffer.tobytes()

tempo = time.time()

nomeFrame = f"{self.id\_user}-{time.strftime('%Y%m%d\_%H%M%S', time.gmtime(tempo))}"

videredb.guardaFrame(frame, self.id\_user, tempo, objetos\_captuados\_frame)

return nomeFrame

class Video:

def \_\_init\_\_(self, id\_user, filtros, video):

self.filtros = filtros

self.objetos = []

self.id\_user = id\_user

self.video\_id = video

self.frameNumero = 0

self.imagem = cv2.VideoCapture(video + ".mp4", 0)

self.frameTotais = int(self.imagem.get(cv2.CAP\_PROP\_FRAME\_COUNT))

self.net = cv2.dnn.readNet(config.yoloPath, config.yoloPathWeights)

comprimento = int(self.imagem.get(3))

altura = int(self.imagem.get(4))

self.out = cv2.VideoWriter(config.pastaVideos + "/" + self.video\_id + '.webm',

cv2.VideoWriter\_fourcc(\*'vp80'),

self.imagem.get(cv2.CAP\_PROP\_FPS),

frameSize=(comprimento, altura))

self.thread = None

# Inicia CUDA, se utilizador não suportar, estas linhas são ignoradas

if cv2.cuda.getCudaEnabledDeviceCount() > 0:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

else:

self.net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_DEFAULT)

self.net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)

def progresso(self):

p = round((self.frameNumero / self.frameTotais) \* 100)

return str(p)

def processa(self):

while True:

ativo, frame = self.imagem.read()

self.frameNumero += 1

if not ativo: # Termina Processo quando não existe mais frames

self.imagem.release()

d = UTILIZADORES\_ATIVOS.get(self.id\_user)

del d.videos[self.video\_id]

os.remove(os.path.join(self.video\_id + ".mp4"))

videredb.guardaVideo(self.video\_id, self.id\_user, self.objetos)

break

tamanho = frame.shape

layer\_names = self.net.getLayerNames()

outputlayers = [layer\_names[i[0] - 1] for i in self.net.getUnconnectedOutLayers()]

altura = frame.shape[0]

comprimento = frame.shape[1]

blob = cv2.dnn.blobFromImage(frame, 1 / 255, (128, 128), swapRB=True, crop=False)

self.net.setInput(blob)

processados = self.net.forward(outputlayers)

class\_ids = []

confidences = [] # Grau de confiança sobre a imagem

caixas = []

for p in processados:

for ObjetoApanhado in p:

pontuacoes = ObjetoApanhado[5:]

class\_id = np.argmax(pontuacoes)

if class\_id not in self.filtros: # Este objeto detetado não está presente nos filtros para detetar

continue

certeza = pontuacoes[class\_id]

if certeza > 0.7:

# Obtem posição (eu não percebo a magia negra que o net.forward faz, mas as posições das coisas estão ai)

centroX = int(ObjetoApanhado[0] \* comprimento)

centroY = int(ObjetoApanhado[1] \* altura)

c = int(ObjetoApanhado[2] \* comprimento)

a = int(ObjetoApanhado[3] \* altura)

x = int(centroX - c / 2)

y = int(centroY - a / 2)

caixas.append([x, y, c, a])

confidences.append(float(certeza))

class\_ids.append(class\_id)

indexes = cv2.dnn.NMSBoxes(caixas, confidences, 0.7, 0.4)

objetos\_captuados\_frame = []

for i in indexes:

objeto\_no\_frame = {}

i = i[0]

caixa = caixas[i]

x = caixa[0]

y = caixa[1]

w = caixa[2]

h = caixa[3]

objeto\_no\_frame["object\_id"] = class\_ids[i]

objeto\_no\_frame["confianca"] = confidences[i]

objeto\_no\_frame["topLeft"] = [x, y]

objeto\_no\_frame["bottomRight"] = [w, h]

objetos\_captuados\_frame.append(objeto\_no\_frame)

label = str(dataset.classes[class\_ids[i]])

cor = dataset.classes\_cores[class\_ids[i]]

cv2.rectangle(frame, (x, y), (x + w, y + h), cor, 2)

# Impede que texto fique fora do ecrã

if y < 40: # se o X estiver 40 pixeis perto do topo da imagem

y += 45 # Baixa 45 pixeis

if x < 5: x += 5

if x + (len(label) + 5) \* 15 > tamanho[1]: # Impede que o texto saia para o lado da imagem

x -= (len(label) + 5) \* 15

# Faz um texto em baixo que serve como contorno

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x - 1, y - 10),

cv2.FONT\_HERSHEY\_DUPLEX, 1, (255, 255, 255), 2, lineType=cv2.LINE\_AA)

# Nome do objeto, texto de cima

cv2.putText(frame, label + " " + str(round(confidences[i], 2)), (x, y - 10), cv2.FONT\_HERSHEY\_DUPLEX, 1,

cor, 1, lineType=cv2.LINE\_AA)

if class\_ids[i] not in self.objetos:

self.objetos.append(class\_ids[i])

self.out.write(frame)

videredb.py

import cv2

import numpy

from sqlalchemy import create\_engine, text, MetaData

import flask

import config

import bcrypt

import time

import os

# TODO: Inserir frames com execute many

import dataset

engine = create\_engine(config.database, echo=False, future=True)

videreMETA = MetaData()

videreMETA.reflect(bind=engine)

# CRUD

def selectTabela(tabela):

with engine.connect() as con:

return con.execute(text(f"SELECT \* FROM {tabela}"))

# UTILIZADOR

def inserirUtilizador(username, password):

hashedPW = bcrypt.hashpw(password.encode(), bcrypt.gensalt())

statement = text(f"INSERT INTO utilizadores (username, password) VALUES ('{username}', '{hashedPW.decode()}')")

with engine.connect() as con:

con.execute(statement)

con.commit()

def verificaUtilizador(username, password):

with engine.connect() as con:

result = con.execute(text(f"SELECT id, username, password "

f"FROM utilizadores WHERE username = '{username}'")).fetchone()

if result and bcrypt.checkpw(password.encode(), result[2].encode()): # tb verifica se existe username

return result[0]

else:

return None

def verificaDisponibilidadeUser(user):

with engine.connect() as con:

result = con.execute(text(f"SELECT username "

f"FROM utilizadores WHERE username = '{user}'")).fetchone()

if not result:

return True

else:

return False

# STREAM URLS

def inserirStream(user\_id, videre\_url):

with engine.connect() as con:

con.execute(text(f"INSERT INTO stream\_urls (user\_id, videre\_url) "

f"VALUES ({user\_id}, '{videre\_url}')"))

con.commit()

def verificaCriador(user\_id, videre\_url):

with engine.connect() as con:

result = con.execute(text(f"SELECT user\_id, videre\_url FROM stream\_urls "

f"WHERE videre\_url = '{videre\_url}' AND user\_id = '{user\_id}'")).fetchone()

if result:

return True

else:

return False

def deleteStreamURL(videre\_url, user\_id):

with engine.connect() as con:

con.execute(text(

f"DELETE FROM stream\_urls WHERE videre\_url = '{videre\_url}' and user\_id = {user\_id} RETURNING \*")).fetchone()

con.commit()

def apagaTabelaStreams():

with engine.connect() as con:

con.execute(text(f"DELETE FROM stream\_urls"))

con.commit()

# FRAMES

def guardaFrame(frame, userid, timestamp, objects\_found):

"""

:type userid: int # id do utilizador na base de dados

:type objects\_found: [{"object\_id": int, "confianca": double, "topLeft":[x,y], "bottomRight":[w,z]}, ... ]

"""

nomeFrame = f"{userid}-{time.strftime('%Y%m%d\_%H%M%S', time.gmtime(timestamp))}"

caminhoFrame = f"{config.pastaFrames}/{nomeFrame}.png"

img = cv2.imdecode(numpy.fromstring(frame, numpy.uint8), cv2.IMREAD\_UNCHANGED)

cv2.imwrite(caminhoFrame, img)

with engine.connect() as con:

frameID = con.execute(text(f"INSERT INTO frames (timestamp, user\_id, frame\_path) "

f"VALUES ('{time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(timestamp))}', {userid}, '{nomeFrame}') RETURNING id")).fetchone()

con.commit()

"""

videreMETA.tables["objects\_found"].insert()

con.execute(videreMETA.tables["objects\_found"].insert(),

objects\_found)

con.commit()

"""

for object in objects\_found:

topLeft = object.get("topLeft")

bottomRight = object.get("bottomRight")

stmt = "INSERT INTO objects\_found VALUES (" + str(frameID[0]) + ", " + str(

object.get("object\_id")) + ", " + str(object.get("confianca")) + ", '{" + str(topLeft[0]) + ", " + str(

topLeft[1]) + "}', '{" + str(bottomRight[0]) + ", " + str(bottomRight[1]) + "}')"

# statement = text("INSERT INTO objects\_found VALUES (%d, %d, %f, '{{%d, %d}}', '{{%d, %d}}')".format(frameID[0], object.get("object\_id"), object.get("confianca"), topLeft[0],topLeft[1], bottomRight[0], bottomRight[1]))

statement = text(stmt)

con.execute(statement)

con.commit()

def obtemFrames(user\_id):

with engine.connect() as con:

objetos = con.execute(text(

f"SELECT object\_id, frame\_path "

f"FROM frames inner join objects\_found on frames.id = objects\_found.frame\_id "

f"where user\_id = {user\_id}"))

if objetos:

fotos = {}

for row in objetos:

if row[1] in fotos:

if dataset.classes[row[0]] in fotos[row[1]]: continue

fotos[row[1]].append(dataset.classes[row[0]])

else:

fotos[row[1]] = []

fotos[row[1]].append(dataset.classes[row[0]])

for i in fotos:

fotos[i] = " ".join(str(v) for v in fotos[i])

return fotos

else:

return None

def obtemDadaFrame(user\_id, frame):

with engine.connect() as con:

data = con.execute(text(

f"SELECT timestamp "

f"FROM frames "

f"where user\_id = {user\_id} and frame\_path= '{frame}'"))

if data:

dataframe = ""

for i in data:

dataframe = i[0]

return str(dataframe) + " UTC"

else:

return None

def removeFrame(frame, user\_id):

with engine.connect() as con:

result = con.execute(

text(f"DELETE FROM frames WHERE frame\_path = '{frame}' and user\_id = {user\_id} RETURNING \*")).fetchone()

if not result:

return False

path = f"{config.pastaFrames}/{frame}.png"

if os.path.exists(path):

os.remove(path)

else:

return False

con.commit()

return True

def seUserDonoFrame(frame, user\_id):

with engine.connect() as con:

result = con.execute(

text(f"SELECT \* FROM frames WHERE frame\_path = '{frame}' and user\_id = {user\_id}")).fetchone()

if not result:

return False

else:

return True

# VIDEOS

def guardaVideo(nomeVideo, userid, objects\_found):

"""

:type userid: int # id do utilizador na base de dados

:type objects\_found: lista com os números dos objectos detetados

"""

with engine.connect() as con:

videoID = con.execute(text(f"INSERT INTO videos (timestamp, user\_id, frame\_path) "

f"VALUES ('{time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(time.time()))}', {userid}, "

f"'{nomeVideo}') RETURNING id")).fetchone()

con.commit()

for object in objects\_found:

statement = text(f"INSERT INTO objects\_video VALUES ({videoID[0]}, {object})")

con.execute(statement)

con.commit()

def obtemVideo(user\_id):

with engine.connect() as con:

objetos = con.execute(text(

f"SELECT frame\_path, object\_id, timestamp "

f"FROM videos inner join objects\_video on videos.id = objects\_video.video\_id "

f"where user\_id = {user\_id}"))

videos\_user = {}

for row in objetos:

if row[0] in videos\_user:

if dataset.classes[row[1]] in videos\_user[row[0]]: continue

videos\_user[row[0]].append(dataset.classes[row[1]])

else:

videos\_user[row[0]] = []

videos\_user[row[0]].append(dataset.classes[row[1]])

for i in videos\_user:

videos\_user[i] = " ".join(str(v) for v in videos\_user[i])

return videos\_user

def removeVideo(video, user\_id):

with engine.connect() as con:

result = con.execute(

text(f"DELETE FROM videos WHERE frame\_path = '{video}' and user\_id = {user\_id} RETURNING \*")).fetchone()

if not result:

return False

path = f"{config.pastaVideos}/{video}.webm"

if os.path.exists(path):

os.remove(path)

else:

return False

con.commit()

return True

def seUserDonoVideo(frame, user\_id):

with engine.connect() as con:

result = con.execute(

text(f"SELECT \* FROM videos WHERE frame\_path = '{frame}' and user\_id = {user\_id}")).fetchone()

if not result:

return False

else:

return True