Computer Vision to detect *Phishing*

Relatório Final do Projeto Integrador

André Barbosa, up202007398
Guilherme Almeida, up202008866
José Ribeiro, up202007231
Marcos Aires, up202006888

Licenciatura em Engenharia Informática e Computação

Tutor na U.Porto: Miguel Tavares Coimbra
Orientador na empresa/Proponente: Tomás Lima

30 de junho de 2023, Porto

Conteúdo

1 Introdução **2**
1.1 Enquadramento . . . 2
1.2 Objetivos e resultados esperados . . . 2
1.3 Estrutura do relatório . . . 3

2 Metodologia utilizada e principais atividades desenvolvidas **3**
2.1 Metodologia utilizada . . . 3
2.2 Intervenientes, papéis e responsabilidades . . . 3
2.3 Atividades desenvolvidas . . . 4

3 Desenvolvimento da solução **5**
3.1 Requisitos . . . 5
3.2 Arquitetura e tecnologias . . . 6
3.2.1 Pesquisa . . . 6
3.2.2 Arquitetura Definida . . . 6
3.3 Solução desenvolvida . . . 7
3.4 Validação . . . 8
3.5 Utilização . . . 8

4 Conclusões **9**
4.1 Resultados alcançados . . . 9
4.2 Lições aprendidas . . . 10
4.3 Trabalho futuro . . . 10

1 Introdução

1.1 Enquadramento

Computer Vision trata-se de um ramo da Inteligência Artificial que permite aos computadores e sistemas retirar informação valiosa de imagens, vídeos, ou qualquer outro elemento visual e tomar algum tipo de ação consoante essa informação que recebem. Exemplificando no contexto do nosso projeto, *Computer Vision* pode ser útil para fazer classificação de imagens. Imagine-se um exercício de classificação de imagens em que queremos dividir um conjunto das mesmas em duas classes: cão e gato. Resumindo o processo, começaríamos por treinar o nosso modelo de classificação, fornecendo-lhe imagens rotuladas (com os rótulos cão ou gato, consoante a imagem apresentada mostrasse um cão ou um gato, respetivamente) e, seguidamente, após treinarmos o nosso modelo com imagens suficientes, iríamos testá-lo com um conjunto de imagens nunca visto por ele e averiguar o quão preciso seria o nosso modelo consoante a classificação que ele fizesse de cada uma das imagens. A partir daí, refinaríamos o formato de treino do modelo até obter uma classificação que fosse minimamente aceitável.

Phishing trata-se de uma técnica de engenharia social que tem como objetivo principal ludibriar utilizadores da Internet de modo a fazer com que eles partilhem informações pessoais confidenciais (palavras-passe, dados de contas bancárias, etc.) de modo que o atacante possa tirar vantagem financeira ou outra. Uma das técnicas mais comuns de *phishing*, e a que pretendemos combater com este projeto, trata-se de fazer com que *websites* falsos, com aparência idêntica a legítimos, enganem o utilizador, que, sem desconfiar de que o *website* a que está a aceder não é de todo de confiança, deposite informações confidenciais neste que vão ser direcionadas para o agente malicioso que as pretende roubar. Normalmente, as motivações do atacante para efetuar este roubo de informações podem ser diversas, mas a principal razão é a obtenção de lucro monetário com o roubo das mesmas.

O número de indivíduos que se aproveitam da falta de conhecimento de muitos utilizadores da Internet tem vindo a aumentar, aplicando novas técnicas de falsificação de *websites* que são visualmente idênticos aos legítimos visando roubar-lhes informação confidencial e lucrar em cima dessa informação. Sendo um problema corrente, foi nos apresentada uma proposta de arranjar solução para ele. Assim, pretendemos, de uma forma simplificada, desenvolver uma solução que seja eficaz e precisa na deteção de *phishing*, recorrendo a técnicas de *Computer Vision*. Pretendemos que a nossa solução consiga identificar a marca afetada (diga-se, por exemplo, dum banco) através da identificação e reconhecimento do logo usando *Supervised Machine Learning* (recorrendo ao uso de uma técnica idêntica à explicada no exemplo anterior) e verificar se o website é de facto legítimo ou se se trata efetivamente de um caso de *phishing*.

Este projeto foi proposto pela empresa AbuseTotal[4], que se dedica a fornecer uma solução de colaboração no tratamento de ameaças à segurança cibernética.

1.2 Objetivos e resultados esperados

A proposta inicial de trabalho tinha como objetivo de, a partir de um *screenshot* de um *website* de uma determinada empresa, identificar se o mesmo se tratava ou não de um *website* de *phishing*. Após a reunião inicial com a empresa, ficou determinado que o *URL* também era necessário para fazer esse reconhecimento. Dessa forma, dividiu-se o problema em dois aspetos: o reconhecimento da empresa através do *screenshot*, que permite a identificação de logótipos por meio de um modelo de *Machine Learning*, o que passou a ser o objetivo mínimo proposto pela empresa, e a identificação de *phishing*, que consiste na comparação do *URL* fornecido pelo utilizador para obter o *screenshot* com o *website* da empresa obtido pelo reconhecimento anterior.

Alguns objetivos extra incluíam alternativas ao reconhecimento pelos logótipos, como OCR ou análise do código no URL, técnicas que, apesar de não terem sido exploradas, são descritas no tópico 2.3 "Atividades desenvolvidas".

1.3 Estrutura do relatório

- **Introdução:** Esta secção descreve tópicos e conceitos comuns ao desenvolvimento do nosso projeto, o objetivo do projeto e os resultados que o projeto procura alcançar.
- **Metodologia e atividades desenvolvidas:** Esta secção determina e refere o método utilizado ao longo do desenvolvimento do projeto, a identificação de todos os intervenientes no planeamento e desenvolvimento, incluindo *stakeholders*, a distribuição de responsabilidades e a descrição das atividades concretizadas e que culminaram na conclusão do projeto.
- **Desenvolvimento da solução:** Envolve todo o processo de produção da solução, desde o trabalho inicial de pesquisa, passando pelo desenvolvimento do código em si e terminando com a descrição da análise aos resultados obtidos e métricas de avaliação.
- **Conclusões:** Esta secção envolve uma descrição mais detalhada dos resultados, competências adquiridas e lições aprendidas com o trabalho. Refere ainda possíveis direções para o desenvolvimento de um trabalho futuro mais aprofundado.

2 Metodologia utilizada e principais atividades desenvolvidas

2.1 Metodologia utilizada

Utilizamos o Linear[5] como forma de dividir melhor as tarefas entre os diferentes elementos do grupo, podendo-se considerar que recorremos a um desenvolvimento iterativo com *sprints* quinzenais.

Foram realizadas reuniões semanais. Apesar de não terem sido feitas com acompanhamento do tutor da U.Porto, quase todas as semanas houve encontro entre todos os elementos do grupo para discutir o ponto de situação do projeto e a fazer nova distribuição de tarefas.

De forma a conseguirmos trabalhar todos em conjunto com a empresa e de modo que eles pudessem acompanhar o trabalho que íamos a desenvolver ao longo do tempo, utilizámos um servidor próprio criado por eles no Visual Studio Code[1], o que facilitou bastante a troca de ideias com os elementos da empresa.

Para desenvolver e treinar o nosso modelo de *Machine Learning*, utilizamos a linguagem de programação Python[2], já que julgamos ser a que fornece maior apoio, em termos de bibliotecas, para o desenvolvimento destes modelos.

Foi usado também o GitHub[3] para termos um repositório comum onde pudéssemos armazenar e compartilhar o código desenvolvido por cada elemento.

2.2 Intervenientes, papéis e responsabilidades

Além dos quatro elementos constituintes do grupo, este projeto envolveu, também, o tutor do nosso projeto na U.Porto, o professor Miguel Tavares Coimbra e o orientador na empresa/proponente do nosso projeto, Tomás Lima.

Consideramos que os *stakeholders* (todas as partes que estão interessadas no projeto/produto final a ser desenvolvido) deste projeto são, além dos já mencionados em cima, todos os envolvidos na empresa AbuseTotal (da qual o proponente do nosso projeto é um dos donos), empresa que pretende tirar proveito da solução por nós desenvolvida, podendo ser, por isso, considerados como utilizadores finais.

Como o problema para o qual nos propusemos a desenvolver uma solução é bastante complexo, decidimos dividi-lo em partes menores, de forma a ser mais fácil tratá-lo, com algumas etapas que podiam ser realizadas em paralelo com outras, e outras que exigiam o desenvolvimento das anteriores para poderem ser tratadas. Irá ser discutido em mais detalhe, na próxima secção do relatório, como foi feita a divisão das etapas por cada elemento do grupo. Será também feita uma descrição pormenorizada daquilo que se pretendia com cada uma das etapas.

2.3 Atividades desenvolvidas

Como primeira forma de contacto em que começamos efetivamente a desenvolver algum trabalho, destacamos a primeira reunião que tivemos com o proponente do nosso projeto e com mais alguns membros da AbuseTotal nas suas instalações. Nessa primeira reunião procedemos à divisão do problema em diferentes etapas, definindo as etapas 0 e 1 como objetivo mínimo, etapa 2 como objetivo do trabalho proposto e etapas 3 e 4 como extras, e à distribuição de cada uma destas pelos diferentes elementos do grupo:

- 0. Neste primeiro passo, decidimos começar por desenvolver um script que, dado o *URL* de um determinado *website*, fosse capaz de retornar um *screenshot* desse mesmo website.
- 1. Aqui, no primeiro sub-passo, procedemos à recolha de logótipos. Escolhemos fazer a recolha de 10 logótipos de empresas de 4 tipos diferentes (bancos, plataformas de pagamento online, redes sociais e lojas online), fazendo um total de 40 logótipos. Recolhemos apenas esta quantidade como forma de uma amostra. Porém, num cenário ideal, seria necessária uma base de dados com milhares de logótipos de modo a, dada uma determinada *screenshot*, haver a mais alta probabilidade da identificação e reconhecimento do logótipo ser efetuada com sucesso, dado que seria impossível o modelo reconhecer um logótipo caso ele não tivesse sido treinado com o mesmo. No segundo passo começamos a fazer uma pesquisa sobre como poderíamos desenvolver e treinar o nosso modelo, pesquisando sobre diferentes modelos de *Machine Learning* que fossem capazes de identificar logótipos e a correspondente empresa a que eles se referem. Foi nesta fase que nos apercebemos que iríamos ter que ser nós a desenvolver o nosso próprio modelo (com base em modelos já existentes), uma vez que não encontramos um modelo que fizesse exatamente o que pretendíamos.
- 2. Esta etapa refere-se à solução propriamente dita, que consistiu, na sua totalidade, resumidamente, na extração de *screenshots*, no treino do modelo para reconhecimento dos logótipos, e na deteção de *phishing* através do *URL* do *website*.
- 3. A partir deste tópico, encontram-se possíveis adições ao trabalho requerido e que poderiam permitir uma maior eficácia na deteção de *phishing*. Neste tópico em específico, era considerada uma hipótese de, a partir de uma análise da estrutura do código com *Machine Learning*, encontrar alguns padrões que fossem comuns a práticas de *phishing*. Esta análise poderia ser mais geral, relativamente à estrutura do *website* em si(gráfica e visualmente), ou mais particular, relativamente a padrões de código comuns.
- 4. A abordagem definida desde o início para a utilização de *Computer Vision* prendeu-se em focar nos logótipos presentes na imagem para facilitar o reconhecimento da marca afetada. Neste tópico, propõem-se uma alternativa, OCR (*Optical Character Recognition*), que teria como principal foco o texto presente nas imagens recolhidas. A partir do reconhecimento do texto, seria possível compará-lo com alguns erros ortográficos ou gírias que costumam ser utilizados em *phishing*, facilitando a sua deteção.

Uma vez que o desenvolvimento das etapas 0 e 1 poderiam ser realizadas em simultâneo, decidimos paralelizar a sua execução. Assim, dois membros do grupo dedicaram-se à etapa 0 e os outros dois à etapa 1.

Fizemos a segunda reunião com o proponente algum tempo depois, em que apresentámos o desenvolvimento completo das etapas 0 e 1. Nessa mesma reunião, continuamos a trabalhar no sentido de encontrar um modelo de *Machine Learning* que permitisse, dado um *screenshot* de um determinado *website*, identificar o logótipo e reconhecer a que empresa esse mesmo correspondia. No final deste encontro, ficamos com uma ideia de como atacar o problema e foram definidos os próximos passos assim como divisão de tarefas entre os elementos do grupo.

A partir daí fomos reunindo entre os elementos do grupo para o contínuo desenvolvimento do modelo.

Como reunião final relevante, destacamos a reunião que tivemos e que ditou o fim do desenvolvimento do modelo de identificação e reconhecimento de logótipos. Nesta, procuramos fazer ajustes que tornassem o modelo mais eficaz e preciso. Para isso, testamos diferentes graus de confiança. Por último, desenvolvemos a parte de deteção de *phishing*. Precisamos de recolher, para cada logo, o seu

respetivo domínio do *website*, de forma a podermos comparar com ele em caso do modelo detetar a marca.

Na Figura 1 estão descritas todas as atividades realizadas ao longo do tempo.

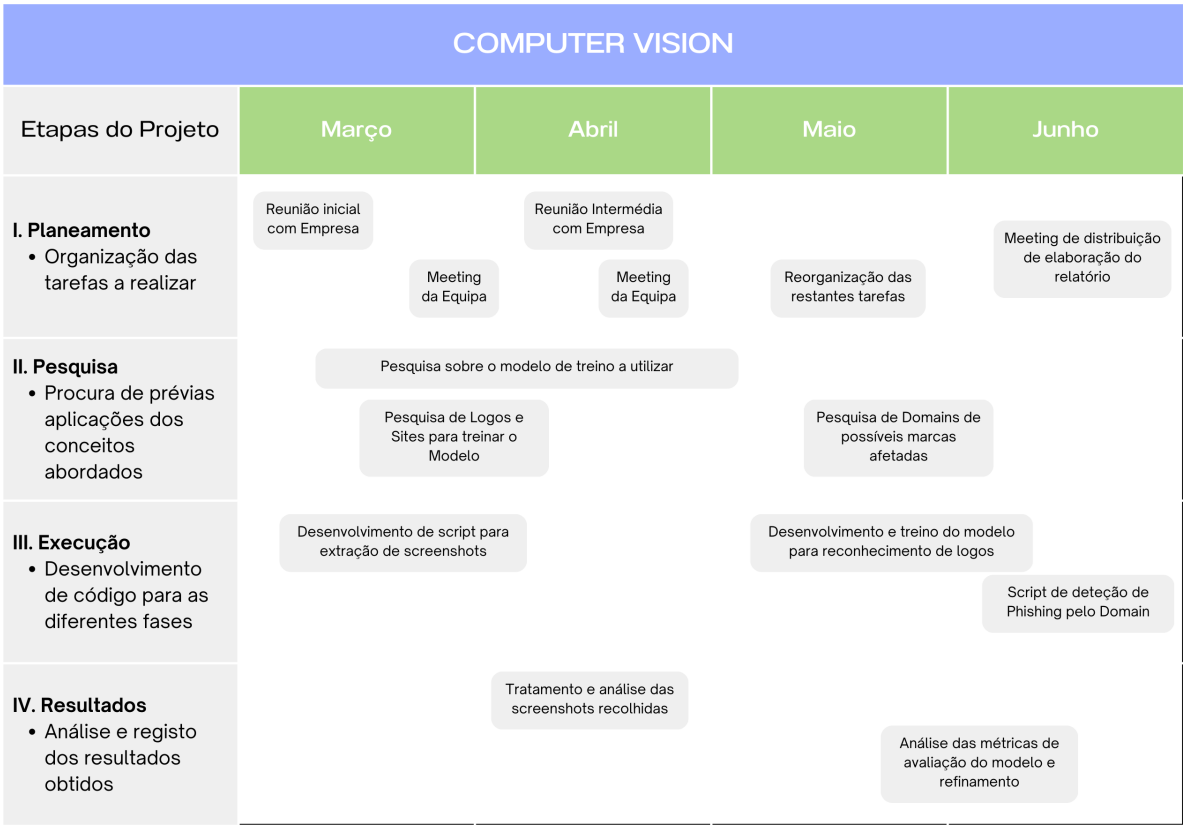


Figura 1: Diagrama de Gantt

3 Desenvolvimento da solução

3.1 Requisitos

Como requisitos funcionais (funcionalidades que o sistema deve ser apresentar, isto é, uma ação que o sistema deve ser capaz de realizar), a nossa solução deve ser capaz de identificar logótipos numa dada *screenshot* de um *website* e fazer a correspondência entre esses logótipos e a empresa a que o mesmos representam. Além disso, deve permitir, fazendo a comparação entre o *URL* fornecido pelo utilizador com o *URL* real do *website* em questão, averiguar se este último se trata de um caso de *phishing* ou não.

Como requisito não funcional (característica que o sistema deve apresentar, por exemplo, no que toca a segurança, disponibilidade, usabilidade, etc.), a empresa apresentou-nos ainda o facto de que a solução deveria ser desenvolvida em Python, de forma a facilitar a integração da mesma com o código já desenvolvido por eles em relação a este problema.

Em termos de restrições ao projeto, apenas destacamos a limitação temporal devido à data estabelecida para a entrega do projeto.

3.2 Arquitetura e tecnologias

3.2.1 Pesquisa

Inicialmente realizou-se uma pesquisa entre diferentes tipos de tecnologias de deteção de logótipos, para nos familiarizarmos com os conceitos. Depois procuramos encontrar um modelo de inteligência artificial já totalmente treinado e com uma vasta coleção de empresas na sua base de dados. Naturalmente, para utilizar qualquer um destes softwares encontrados seria necessário pagar o seu serviço. A partir deste momento percebemos que seria necessário treinarmos o nosso próprio modelo com uma tecnologia *open source* de inteligência artificial e *Machine Learning*. Para isso, surgiram alguns conceitos técnicos que poderiamos utilizar, como a tecnologia *SIFT(Scale Invariant Feature Transform)*, que deteta pontos de interesse únicos numa imagem, ou a tecnologia de deteção *Harris Corner*, que identifica os cantos de uma imagem através de variações de grande intensidade do gradiente em todas as direções, mas acabamos por seguir a estrutura do *YOLOv5*.

3.2.2 Arquitetura Definida

O *software* utilizado foi desenvolvido pela *Ultralytics* e denomina-se por *YOLOv5*. É uma uma arquitetura de rede neural desenvolvida para a deteção de objetos em imagens e videos. Esta tecnologia pertence à família YOLO (*You Only Look Once*) sendo já a quinta versão desenvolvida. A escolha por esta tecnologia foi relativamente fácil pois apresenta uma arquitetura mais rápida e o seu foco principal vai ao encontro do tema do nosso trabalho.

Este software apresenta alguns modelos pré-treinados no qual podíamos utilizar como ponto inicial no treino do nosso modelo. Acabamos por optar pelo segundo modelo mais pequeno (*YOLOv5s*) devido à necessidade de se realizar os treinos localmente nos nossos computadores pessoais e pelo facrto de ser o modelo mais rápido.

A principal dificuldade encontrada foi na procura e tratamento dos dados a utilizar para treinar o modelo. Como se sabe a obtenção de um conjunto de dados de qualidade é crucial para treinar um modelo de inteligência artificial. Idealmente teria sido utilizado um elevado número de imagens de logotipos e websites das empresas alvo juntamente com um respetivo ficheiro contendo as coordenadas do logotipo na imagem. A pesquisa envolveu explorar diferentes fontes, como repositórios públicos e plataformas online de bases de dados, avaliando a qualidade das imagens e tendo em conta a diversidade das empresas apresentadas.

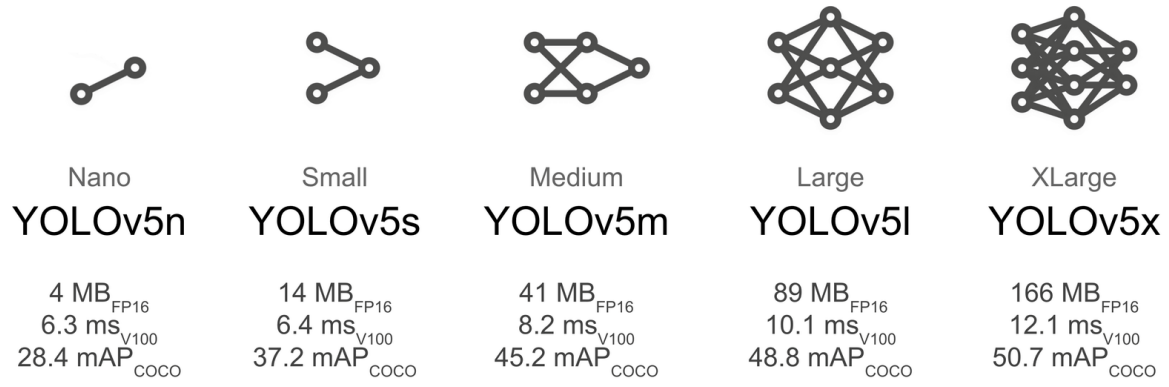


Figura 2: Modelos YOLOv5 disponíveis

3.3 Solução desenvolvida

Inicialmente definiu-se as empresas que seriam utilizadas para a base de dados do modelo, empresas estas que têm mais tendência a serem alvo de phishing (bancos, redes sociais, lojas online, etc.). Definiu-se cerca de 35 empresas e recolheu-se cerca de 2 imagens por empresa. Foi utilizado o *software open source Label Studio* como apoio para adicionar as localizações de cada logotipo em cada imagem.

Após um primeiro treino do modelo com as imagens recolhidas e sem qualquer êxito, percebemos que a disponiblidade de um conjunto de dados extenso e diversificado seria fundamental para treinar um modelo de qualidade. Deste modo, decidimos explorar bases de dados online que continham grandes quantidades de imagens diferentes de diversas empresas, descartando assim as empresas definidas previamente, tendo como principal foco o desnvolvimento de um modelo capaz de detetar com precisão os logotipos.

A base de dados utilizada (acessível aqui) fornecia também já os ficheiros adicionais com as localizações dos logos para cada imagem, algo que evitou a necessidade de realizar esta anotação manualmente. O conjunto de dados final contém então um total de 4862 imagens com 5 empresas - *Coca-Cola, Fedex, Ford, Pepsi and Shell*.

Após a recolha dos dados estar concluida foi necessário realizar o tratamento dos mesmos, separando as imagens em três conjuntos distintos - *train, validate and test*, utilizados para treinar, ajustar os hiperparâmetros e avaliar o desempnho e testar o desmpenho final, respetivamente.

Tendo os dados completamente recolhidos e tratados, avançamos para a preparação do YOLOv5 para o treino do modelo, onde foi necessário criar e configurar o ficheiro *logodetection2.yaml* que contem as informações para os diferentes *datasets* e informação sobre as classes. Nesta fase foi também necessário tomar decisões em relação aos hiperparâmetros, decisões que foram limitadas devido aos recursos computacionais disponíveis. Deste modo optou-se pelo ficheiro de aumento de baixo nível - *hyp.scratch-low.yaml*. De seguida realizou-se o treino, onde foram utilizados 16 *batches* - tamanho de cada lote, e 10 *epochs* - número de iterações pelos dados todos, o treino teve uma duração total de cerca de oito horas e meia.

O próximo passo foi criar o *frontend* da aplicação, onde se desenvolveu um pequeno menu através da *command line*, onde o utilizador consegue inserir o link e/ou o screenshot da empresa obtendo assim uma resposta relativa à legitimidade do website. O menu oferece informações sobre o estado do programa ao longo do processo todo assim como avisos e ajudas quando algo de errado acontece. O menu oferece ao utilizador duas opções, inserir apenas o url do *website* ou inserir o url juntamente com o *screenshot* do *website*. Na primeira opção o programa irá automaticamente tirar a foto do *website* inserido.

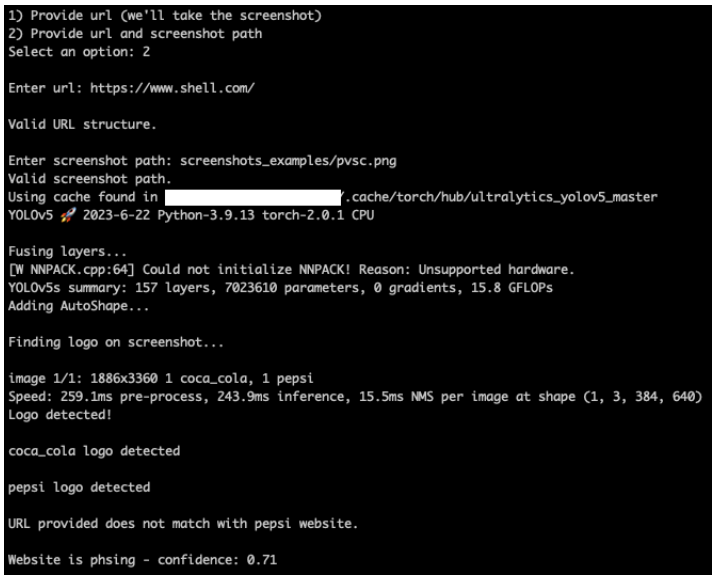


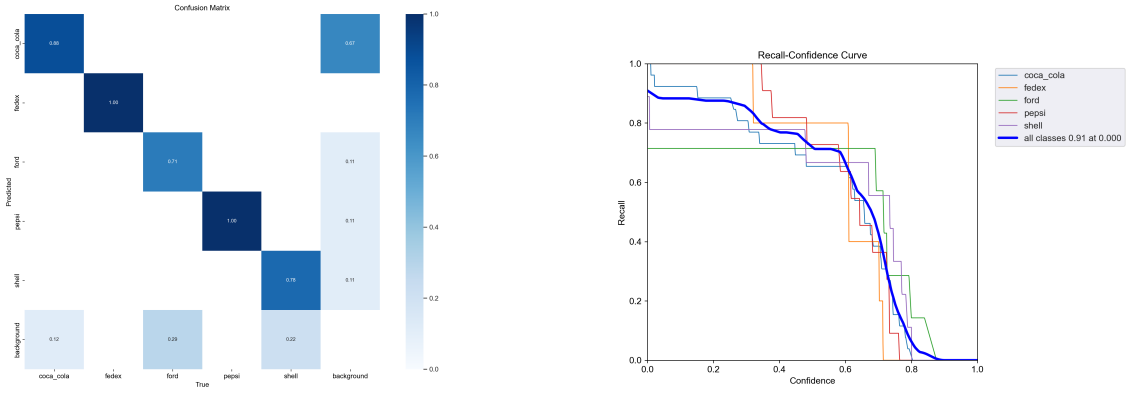
Figura 3: Exemplo da Interface da Command Line

3.4 Validação

Após a deteção do logotipo o programa vai comparar o URL fornecido pelo utilizador com o URL oficial da empresa reconhecida, a partir de um nível de confiança (*threshold*), o programa dá a sua resposta afirmando se o *website* é potencialmente *phishing* ou não.

Os resultados relativos à *performance* do modelo não foram de todo ao encontro das nossas expectativas. O desempenho de um modelo depende muito da qualidade e da quantidade de dados de treino, assim como o próprio processo de treino. Um aspeto importante a considerar é o número de iterações de treino (*epochs).* Normalmente recomenda-se um número relativamente alto de epochs para o treino, de modo a permitir o modelo a aprender padrões significtivos nos dados. No entanto, um treino com um grande número de *epochs* é computacionalmente dispendioso e demorado, limitando assim o potencial do nosso modelo. Contudo, ao testar o produto final, verificamos que na maior parte dos *screenshots* inseridos, o modelo foi capaz de detetar corretamente os logotipos das empresas, mostrando assim que num contexto real, o programa funciona particularmente bem, contrariando os resultados da sua *performance*.

Os gráficos da Figura 4 mostram uma tendência de maior eficácia para logos com maior detalhe (maior diferença em cores ou formato), como o da Fedex ou da Pepsi. Analisando a curva de *Recall-Confidence*, denota-se que a partir de uma *threshold* da confiança de 0.6, o *threshold* diminui bastante. Encontra-se também valores de *recall* maiores com *threshold* de confiança menores e valores de *recall* menores com *threshold* de confiança maiores para logos como o da Fedex comparados com logos como o da Ford, devido à tendência de, com intervalos de confiança menores, os detalhes serem mais importantes e facilmente reconhecidos.



(a) Confusion Matrix

(b) Recall-Confidence Curve

Figura 4: Análise de métricas de performance

3.5 Utilização

De modo a correr corretamente o código deve-se seguir os seguintes passos:

- instalar bibliotecas - *pip install -r requirements.txt*
- correr script - *python3 script.py*

4 Conclusões

4.1 Resultados alcançados

Como já foi referido anteriormente, tínhamos dois objetivos diferentes que pretendíamos atingir com este projeto: o objetivo mínimo, que se prendia com, dado um *screenshot* de um determinado *website*, identificar e reconhecer a que empresa correspondia o logótipo identificado; o objetivo adicional, que consistia em averiguar se o *website* se tratava ou não de *phishing*.

Conseguimos atingir o objetivo, sendo que o nosso modelo consegue identificar e reconhecer os logótipos em diferentes *screenshots* de diferentes *websites* e fomos capazes de fazer com que a nossa solução, através da comparação entre o *URL* fornecido pelo utilizador com o *URL* real do *website*, identificasse se este último se tratava de um caso de *phishing* ou não. Todavia, temos algumas noções do que seria necessário fazer para que o nosso produto final desenvolvido fosse também capaz de detetar *websites* de *phishing* de forma ainda mais eficaz, tal como demonstramos no tópico 2.3 "Atividades desenvolvidas", em que descrevemos cada um dos passos que teríamos que seguir para alcançar esse objetivo. Para atingir esse propósito, poderíamos optar, por, em detrimento de fazer uma análise do screenshot, analisar o código-fonte do website, por exemplo, abordagem que é sempre mais rápida do que fazer a análise de uma imagem. A análise deste código poderia ser útil no sentido de detetar semelhanças/padrões entre este e o código de websites não legítimos, parecenças estas que se poderiam revelar fulcrais na deteção de *phishing*. Outra abordagem que se poderia mostrar de grande utilidade no aumento da eficácia na deteção de phishing seria o uso de OCR, que canaliza o seu foco para o texto presente nas imagens recolhidas. Este texto poderia ser posteriormente confrontado com alguns erros ortográficos ou gírias comummente usadas em *phishing*, facilitando assim a sua deteção.

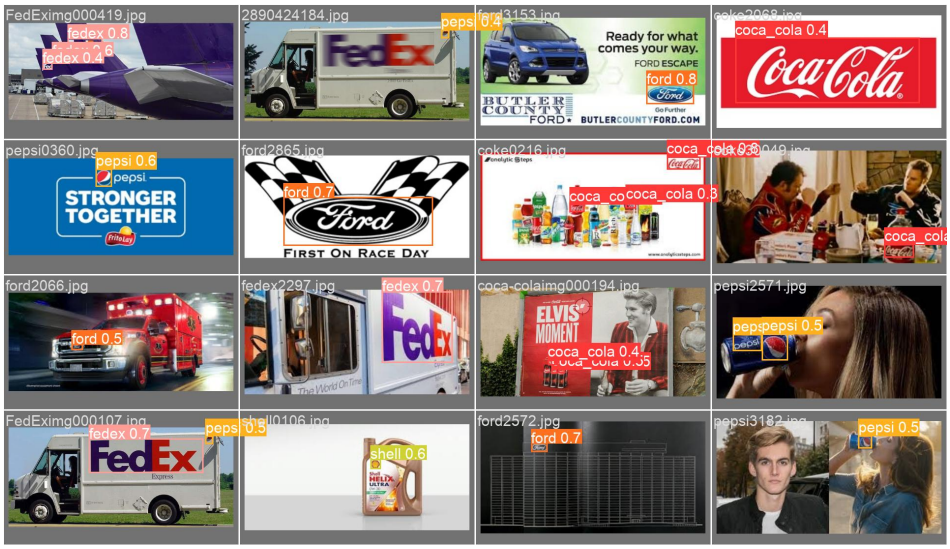


Figura 5: Exemplo de Resultados do Modelo

Consideramos que o trabalho foi igualmente distribuído entre os vários elementos do grupo e todos contribuíram igualmente para o desenvolvimento do projeto.

Contribuição de cada elemento (em termos quantitativos):

- André Barbosa, 25%
- Guilherme Almeida, 25%
- José Ribeiro, 25%
- Marcos Aires, 25%

4.2 Lições aprendidas

Num projeto desta dimensão, é comum aparecerem obstáculos de baixa e média dificuldade. A maioria deles, neste projeto, resultou da nossa falta de conhecimento na área de *Machine Learning* e *Computer Vision*, uma vez que apenas estávamos a ter o primeiro contacto com estas áreas aquando da realização deste projeto. Essa lacuna levou a que a criação do modelo fosse a etapa mais morosa na caminhada para o objetivo final. Para nos adaptarmos às dificuldades, começamos por trabalhar com modelos pré-feitos, que acabaram por não ir ao encontro da solução que pretendíamos desenvolver. Na falta de modelos que fossem ao encontro do objetivo que pretendíamos alcançar, tivemos então de, a partir duma pesquisa extensiva, criar um modelo próprio, com base num modelo já existente, o que levou a um grande atraso no nosso progresso inicial. Isto deveu-se principalmente ao facto de, inicialmente, mais uma vez devido a não estarmos familiarizados com as áreas de conhecimento que o desenvolvimento desta solução exigia, nos termos focado em demasia na procura de um modelo já treinado e que fosse imediatamente de encontro às nossas exigências. Apesar de todos estes percalços, acabamos por completar os objetivos propostos. Contudo, consideramos que poderíamos ter feito um melhor trabalho no controlo do tempo e na comunicação com o proponente, uma vez que houve algumas semanas em que houve um maior descuido, o que levou a deixar uma parte significativa do trabalho para a fase final do desenvolvimento. Caso tivéssemos feito uma melhor gestão do tempo, possivelmente teríamos conseguido alcançar os objetivos extra, acabando por tornar o nosso modelo mais eficaz na deteção de *phishing*.

Apesar de tudo, consideramos que, com a criação desta solução para o problema proposto, adquirimos conhecimentos relevantes na área de *Computer Vision*, *Machine Learning* e Inteligência Artificial, conhecimentos tais que nos podem ser bastante úteis no futuro. Além disso, adquirimos ainda alguma capacidade em compreender como devemos trabalhar em grupo em projetos de média escala em que estamos em contacto com um cliente que nos procura para desenvolvermos uma solução para o problema que ele quer resolver, desde a parte de análise do problema, passando pela divisão do mesmo em partes unitárias, distribuição e desenvolvimento dessas mesmas etapas por cada um dos elementos do grupo, após uma fase intensiva de pesquisa sobre cada uma delas, acabando na junção de cada uma dessas partes, para obter um produto final coeso e que, além de resolver o problema proposto, respeite e vá de encontro aos requisitos propostos pelo cliente.

4.3 Trabalho futuro

Nós consideramos que atingimos o objetivo inicial. O nosso produto, contudo, não é a forma mais eficaz de fazer a deteção de *phishing* nos *websites*, uma vez que tem um consumo maior de tempo e de recursos, comparado com outras técnicas. Para o tornar mais eficaz, deveríamos providenciar melhorias ao produto. A principal melhoria que poderiamos fazer no futuro seria seguir os passos extra que tinhamos delineado nas etapas 3 e 4, isto é, usar a técnica de OCR ou analisar o código-fonte com mais detalhe para encontrar indícios.

Referências

[1] Visual Studio Code. Visual studio code. `https://code.visualstudio.com/`, 2023.

[2] Python Software Foundation. Python. `https://www.python.org/`, 2023.

[3] GitHub Inc. Github. `https://github.com/`, 2023. [Online; acedido 3 de março de 2023].

[4] AbuseTotal Lda. Abusetotal. `https://abusetotal.com/`, 2023.

[5] Linear. Linear app. `https://linear.app/`, 2023.