UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Sistema Integrado de Análise Forense de Dispositivos de Armazenamento para Reconhecimento Visual Automático

Tânia Vanessa Simões Troncão

Mestrado em Segurança Informática

Dissertação orientada por:
Prof. Doutor Fernando Manuel Valente Ramos

2020

Agradecimentos

Agradeço de coração a todos aqueles que me apoiaram, de uma forma ou de outra, na concretização deste trabalho. Em primeiro lugar, à minha família, à minha avó e ao meu namorado, especialmente, por todo o apoio psicológico, motivação e muita paciência para a minha pessoa.

Agradeço ao meu orientador, Prof. Doutor Fernando Ramos, por toda a disponibilidade para me ajudar, por todas as revisões, pelos (infinitos) emails trocados, pela sua paciência comigo, por tudo, obrigada !

Ao meu co-orientador Francisco Loureiro, que me acompanhou ao longo do trabalho e ao Nelson Blanco por me ter proposto esta aventura.

Sou grata por ter tido todas estas pessoas comigo na realização deste trabalho. Graças a Deus !

Resumo

Um dos requisitos fundamentais da análise forense é a necessidade de recolha de evidências digitais para investigação de crimes. Em particular, é muito comum a procura de indícios de crimes em dispositivos pertencentes aos suspeitos, de forma a encontrar pistas que ajudem na investigação. Esta procura de pistas no computador é um problema complexo e moroso, estando dependente, muitas vezes, de intervenção humana. O objetivo desta tese é ajudar este processo, automatizando a procura, em particular no contexto de fotografias que possam indiciar algum crime, para auxiliar os especialistas na área.

O processo de análise do computador de um suspeito levanta, no entanto, dois problemas. O primeiro consiste na análise das imagens de disco rígido, necessária para aceder ao conteúdo a analisar. O segundo problema é a classificação de fotografias de forma automática. Atualmente, é possível encontrar no mercado soluções para cada um dos problemas individualmente, mas não existe nenhuma solução integrada. O nosso objetivo é desenvolver um sistema integrado que ajude os especialistas forenses a procurar indícios de crime numa imagem de disco do computador de um suspeito.

Para nos ajudar a definir as especificações da solução a desenvolver, realizámos um estudo de mercado relativamente às soluções já existentes para cada um dos problemas. Além disso, realizámos testes práticos para conhecer de forma aprofundada as funcionalidades e o desempenho destas soluções, de forma a servir de base para a definição dos requisitos da solução a desenvolver. A solução integrada que desenvolvemos foi avaliada com fotografias retiradas da internet, gerando uma classificação maioritariamente positiva e obtendo poucos falsos positivos. Os resultados da avaliação demonstram a utilidade da solução, mas apontam também melhorias a fazer no futuro.

Palavras-chave: Análise forense, imagem de disco, classificação de fotografia.

Abstract

One of the fundamental requirements of forensic analysis is to collect digital evidence for crime investigations. It is very common to look for crime evidences on devices which belong to suspects in order to find clues for the investigation. This search for clues on the computer is a complex time-consuming problem and is often dependent on human intervention. The purpose of this thesis is to help this process by automating the search, particularly in the context of photographs that may indicate a crime, to assist experts in the field. The process of scanning a suspect's computer raises however two problems. The first is the analysis of hard disk images to access the content to be analyzed. The second problem is the automatic classification of photographs. Currently, solutions to each problem can be found on the market individually, but there is no integrated solution. Our goal is to develop an integrated system that will help forensic specialists to look for evidence of a crime on a suspect's computer disk image. To help us define the specifications of this system, we have conducted a market survey of existing solutions to each problem. In addition, we have conducted practical tests to gain a thorough understanding of the features and the performance of the solutions as a basis for defining the requirements of the solution we intended to develop. The integrated solution developed was evaluated with photographs taken from the internet, generating accurate classifications, with few false positives. The evaluation results demonstrate the usefulness of the solution but also point to improvements to be made in the future.

Keywords: Forensic analysis, disk image, photo classification

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

É bastante comum, na área da análise forense, a análise de dispositivos como computadores ou discos externos, com o intuito de encontrar provas de crime ou alguma evidência que auxilie a investigação de determinados casos policiais. Nos casos de crimes relacionados com crianças, em particular, é feita uma tentativa da sua identificação no dispositivo recolhido como prova, através da análise ao conteúdo do computador do suspeito. Este procedimento é comum a outros crimes, tal como aqueles relacionados com o uso de armas ou falsificação de cartões de crédito: os especialistas forenses analisam os dispositivos pertencentes aos suspeitos com o objetivo de identificar fotografias que possam servir como prova do crime que estão a investigar.

O procedimento é, em teoria, relativamente simples. Em primeiro lugar, é necessário recolher os equipamentos a analisar. Após a sua recolha e identificação como evidências, é realizada uma análise aos mesmos através de software especializado, de forma a extrair todas as fotografias que existam no computador recolhido. Depois de todas as fotografias serem localizadas é feita a classificação de cada uma delas, através de outro software especializado.

Como estes dois processos estão separados, a obtenção de resultados é dificultada e o processo é moroso e ineficiente. Idealmente, todo o processo seria automatizado, para simplificar a tarefa dos especialistas forenses e agilizar o processo.

1.1 Motivação

A importância deste problema pode ser demonstrada através do exemplo de um caso policial relativo ao crime de pedofilia. Os especialistas têm de ir a casa de um suspeito recolher evidências fotográficas para posterior análise. Para tal, é necessário aceder ao computador do mesmo, de onde é extraída uma imagem de disco, de forma a ser possível realizar uma análise detalhada em laboratório. Essa imagem de disco poderá conter conteúdo extremamente relevante para encontrar indícios de crime, tal como a presença de fotografias de menores. Essa procura é normalmente complexa, envolvendo várias soluções especializadas cuja integração exige trabalho manual, tornando o processo moroso.

Mais especificamente, são necessárias soluções direcionadas para ler a imagem de disco e extrair as fotografias, e outros programas distintos direcionados para fazer a classificação das fotografias extraídas. Existem atualmente várias aplicações que solucionam cada um dos subproblemas, no entanto não existe uma solução integrada. O ideal seria uma solução que automatizasse e permitisse ao especialista realizar todo o processo num só local.

O desafio passa então por estudar detalhadamente as soluções que já existem para cada um dos problemas, incluindo a concretização de testes de funcionalidade e desempenho. Além disso, é preciso desenvolver novos mecanismos que permitam integrar soluções que foram pensadas individualmente. Este é o alvo de estudo deste trabalho.

1.2 Objetivo

O objetivo do trabalho é o desenvolvimento e avaliação de um sistema integrado que ajude os especialistas forenses a encontrar indícios de crime em computadores de suspeitos, agilizando e simplificando todo o processo. Especificamente, pretende-se estudar e desenvolver um sistema que integre tecnologias da área de análise de imagem de disco e da área de classificação de fotografias.

Portanto, o objetivo é desenvolver uma solução que consiga analisar e extrair fotografias de uma imagem de disco e fazer, de forma automática, uma análise e posterior classificação das fotografias extraídas para o reconhecimento de três tipos de fotografias-alvo: crianças, armas e cartões de crédito. Como resultado, pretende-se facilitar o processo e reduzir o tempo de obtenção de resultados.

Resumindo, os objetivos práticos deste trabalho traduzem-se nos seguintes pontos:

1. Analisar uma imagem de disco.
2. Extrair fotografias da imagem de disco.
3. Classificar as fotografias extraídas, consideradas conjunto de fotografias alvo, comparando-as com um conjunto de fotografias externas, consideradas de referência, previamente rotuladas.
4. Apresentar resultados relativo à semelhança das fotografias, isto é, para cada fotografia analisada indicar a categoria em que foi classificada e o resultante nível de precisão.

1.3 Contribuições

A contribuição deste trabalho é o desenho, implementação e avaliação de um sistema integrado de procura e classificação de fotografias para ajudar na busca de indícios de crime para investigações forenses.

A solução integra a análise da imagem de um disco onde se localizam as fotografias alvo, a classificação automática e quantificação de semelhança entre as fotografias alvo e as de referência, de modo a reduzir o tempo e a morosidade deste procedimento e ajudar na tomada de decisão.

1.4 Estrutura do Documento

O trabalho desenvolvido está dividido nos seguintes capítulos:

- Capítulo 2 - Estudo do Estado da Arte

  Neste capítulo está descrito o estudo realizado relativamente às tecnologias existentes no mercado para as duas áreas de interesse deste trabalho, análise de imagem de disco e classificação de fotografias.

  As tecnologias foram analisadas e detalhadas a nível teórico, sendo que algumas das tecnologias foram analisadas também a nível prático. Examinaram-se tecnologias pagas e open-source de forma a ser feita uma comparação quanto às funcionalidades de ambas, tomando a ferramenta de código fechado como base e exemplo de comparação, dado ser normalmente a tecnologia mais madura.

  Na área de classificação de fotografias, além de serem detalhadas tecnologias da área, foram também abordadas algumas técnicas e/ou métodos relacionados com o mesmo, como machine learning, deep learning e template matching. Fez sentido abordar estes métodos uma vez que são a base conceptual de parte das tecnologias apresentadas.

- Capítulo 3 - Requisitos Funcionais e Não Funcionais

  Neste capítulo são identificados os requisitos funcionais e não funcionais e respetivas especificações para a solução a desenvolver. Especificamente, detalhamos fatores como tempos de resposta, ambientes onde a solução pode ser executada, funcionalidades e tipos de imagem suportados, entre outros.

- Capítulo 4 - Desenho e Implementação da solução

  Neste capítulo descrevemos o desenho da solução, que tem o objetivo de dar uma ideia geral da solução final, isto é, dar uma perceção intuitiva das várias soluções desenvolvidas e a forma como se interligam entre si. Enquanto o desenho dá uma noção abrangente de tudo o que foi desenvolvido, indicando os pontos chaves de cada solução, tal como os argumentos de entrada e saída, neste capítulo são também descritos todos os passos realizados para a implementação da solução.

- Capítulo 5 - Avaliação da Solução

Este capítulo contém a avaliação da solução, incluindo a metodologia de testes, as decisões tomadas para a criação do modelo de treino, assim como uma discussão dos resultados.

Os testes experimentais tiveram como base uma imagem de disco de 16GB que continha 178 fotografias localizadas aleatoriamente por pastas distintas, sendo o modelo de aprendizagem treinado e validado com base em 4800 fotografias distintas destas.

- Capítulo 6 - Conclusão e Trabalho Futuro

  Neste capítulo apresentamos as conclusões relativamente à solução desenvolvida, bem como determinadas melhorias que poderão vir a ser implementadas.

Capítulo 2

Estudo do estado da arte

Neste capítulo apresentamos o estudo realizado relativamente às ferramentas já existentes nas áreas de interesse para o desenvolvimento da solução. Nomeadamente, tecnologias na área de análise de imagens de disco e na área de análise e classificação de fotografias.

Como método optámos por analisar uma tecnologia closed-source já madura e comparar as suas características e especificidades com tecnologias open-source. Esta estratégia tem como intuito perceber, em primeiro lugar, quais as funcionalidades principais e o desempenho das melhores ferramentas (ao estudar uma tecnologia paga), para depois comparar essas características com as tecnologias open-source. A tecnologia paga é vista como um bom exemplo de tecnologia a seguir e um bom modelo de comparação com as tecnologias open-source, dado que tem toda uma equipa que investe não só a nível profissional, mas também a nível de capital, na mesma.

2.1 Tecnologias para análise de imagem de disco

Para a área de análise de imagens de disco, a tecnologia paga escolhida para estudo foi a EnCase, uma vez que é uma poderosa ferramenta de investigação forense e dado o seu historial evolutivo ao longo dos tempos. Relativamente às tecnologias open-source, foram escolhidas a SANS SIFT e o The Sleuth Kit por serem as ferramentas mais populares para a investigação forense.

Além de um estudo teórico sobre as tecnologias escolhidas, também foram realizados testes práticos com algumas destas ferramentas. Não foi possível realizar ao EnCase, uma vez que é um produto pago ao qual não tivemos acesso. A ferramenta SANS SIFT não foi testada globalmente mas apenas parcialmente, uma vez que se trata de uma apliance que contém ferramentas que foram testadas à parte, individualmente, como o Autopsy.

2.1.1 EnCase

A ferramenta EnCase [2] [3], da opentext (antiga Guidance Software) é reconhecida como o "gold standard" da forense digital, sendo uma solução que permite obter

informação a partir de uma grande diversidade de equipamentos, incluindo telemóveis/smartphones e tablets. Esta ferramenta oferece um entendimento profundo do ciclo de vida da investigação digital e da importância de manter a integridade das evidências obtidas. Além de ter uma poderosa capacidade de processamento, apresenta também facilidade de utilização na ótica do utilizador.

Esta ferramenta faz uma investigação forense completa, incluindo as seis fases do ciclo de vida de uma investigação digital, permitindo a criação de relatórios com diferentes características dependendo do público alvo em questão. O ciclo de vida de investigação digital, divide-se, como se lista, em seis fases, sendo:

- Triagem

  Neste primeiro estágio, a ferramenta permite procurar, identificar e priorizar evidências, de forma a percebermos se existe a necessidade de uma investigação adicional ou não, fazendo com que os investigadores se foquem apenas no que realmente importa e não seja desperdiçado tempo em evidências que não sejam necessárias.

- Recolha

  O EnCase oferece uma ampla compatibilidade, quer com sistemas operativos, quer ao nível dos dispositivos, permitindo analisar aplicações móveis de iOS, Android e Blackberry.

- Decifragem

  Inclui várias plataformas de suporte à criptografia, como Dell Data Protection, Symantec e McAfee, podendo ser adicionados outros componentes, sendo um o Tableau Password Recovery, uma solução de hardware que identifica e desbloqueia ficheiros protegidos com palavra-passe.

- Processamento

  O processador de evidências do EnCase, considerado o processador de liderança do mercado, consegue automatizar a preparação de evidências de forma a facilitar a investigação. É alimentado através de um mecanismo de indexação com elevada escalabilidade e desempenho, que automatiza procuras complexas sobre várias evidências num único passo, aumentando a eficiência.

- Investigação e análise

  Esta fase foi desenvolvida especialmente para os investigadores, que têm ao seu dispor um amplo conjunto de funcionalidades que lhes permite realizar rápidas triagens sobre as evidências, além da sua análise forense profunda.

- Relatório

  O EnCase disponibiliza uma framework de reporting bastante flexível. Podem ser feitos relatórios mais gerais ou mais específicos consoante o objetivo em questão. Os relatórios podem também ser personalizados de acordo com as especificidades da audiência alvo.

2.1.2 SANS SIFT

Esta solução, open source, consiste numa appliance preparada/configurada com as ferramentas necessárias para realizar uma análise forense profunda. Isto é, é uma imagem que contém uma panóplia de software desenvolvido para forense digital, de forma a permitir uma investigação forense adequada [4]. Alguns dos pacotes incluidos, são [5]:

- DFF – Digital Forensic Framework

  Uma Framework open source, que visa a recolha e análise de evidências sem comprometer a integridade das mesmas.

- EVTX – Event Log Viewer

  Permite auditar ficheiros de logs com extensão EVTX, logs criados pelo sistema operativo Windows. É uma ferramenta que faculta análise aos logs.

- Md5deep

  Ferramenta ao nível da linha de comandos que permite calcular e comparar sínteses MD5, para verificação da integridade de um ficheiro relacionado com uma evidência digital antes de se trabalhar na mesma.

- Volatility

  Descrito na seção 2.1.4.

- Autopsy

  Descrito na secção 2.1.5.

Esta ferramenta tem suporte para evidências de tipo E01 - Expert Witness Format, AFF - Advanced Forensic Format e dd - RAW, e tem capacidade para criar uma linha no tempo através dos syslogs, dado que suporta a ferramenta log2timeline. Tem também a capacidade para examinar o recycle bin através da ferramenta Rifiuti.

Para conseguirmos aceder e analisar uma evidência através desta máquina virtual, é necessária a criação de uma pasta partilhada entre o host e a máquina virtual em questão. Para instalar esta ferramenta, pode usar-se o software Virtual Box ou semelhante software de virtualização.

2.1.3 Autopsy

Este programa [11] é uma interface gráfica para o The Sleuth Kit, que consiste numa coleção de comandos de consola (linux) para analisar imagens de disco e recuperar ficheiros da mesma. Depois de a imagem de disco ser carregada e analisada, a ferramenta permite aceder a todas as fotografias e vídeos contidas no mesmo, com a vantagem de que é possível exportar todos os ficheiros encontrados pelo Autopsy para um local a definir pelo utilizador.

Na prática

Depois de carregar a imagem de disco no programa, escolhemos o tipo de pesquisa que deve ser feita na mesma, isto é, o que faz sentido analisar e descartar. Existe uma ampla variedade de módulos de pesquisa básicos, sendo que ainda podem ser criados adicionalmente pelo examinador novos módulos para auxiliar e/ou melhorar as pesquisas. Também existe a opção de escolha do local onde a pesquisa deve ser feita, incluindo em espaço não alocado ou então em todos os ficheiros e pastas, excluindo o espaço não alocado. Os módulos existentes por omissão, na versão analisada, são mostrados na Figura 2.1.

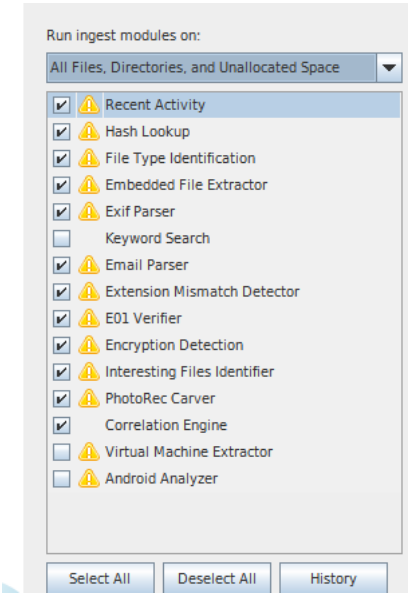

Figura 2.1: Importação de módulos no Autopsy

Para uma imagem com 4Gb, o tempo de espera para a obtenção dos resultados da pesquisa foi de aproximadamente um minuto. Para uma imagem de 16 Gb o tempo de espera para obtenção de resultados foi cerca de 4 minutos. Estes tempo são considerados razoáveis para este problema, assim como o crescimento linear do tempo de análise em função do tamanho do ficheiro.

Como resultado, o programa mostra cinco separadores raiz (Figura 2.2):

- Data Sources: onde encontramos as imagens dos discos que foram carregados.
- Views: onde nos são apresentados de forma organizada todos os ficheiros presentes na imagem identificada no separador anterior,dando-nos uma noção de quantos ficheiros de cada tipo existem no disco (Figura 2.3). Disponibiliza, além de uma contagem dos mesmos, os próprios ficheiros em si, permitindo-nos

uma análise rápida. Também nos indica ficheiros que foram apagados. Assim sendo, conseguimos selecionar facilmente todas as fotografias e exportá-las para um local específico, de forma a podermos usar este conjunto de fotografias como input de um software de análise e classificação de fotografias.

- Results: onde nos são apresentados os resultados dos módulos que correram.
- Reports: Relatórios que podem ser gerados.

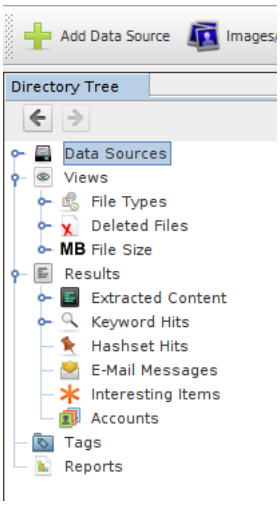

Figura 2.2: Árvore de pastas do Autopsy

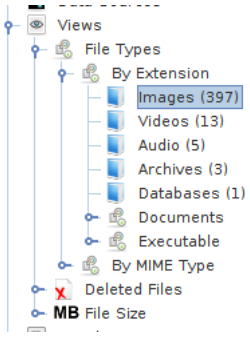

Figura 2.3: Separador "Por extensão"do Autopsy

2.2 Tecnologias para análise e classificação de fotografias

O reconhecimento facial e de objetos é o segundo requisito da solução a desenvolver, por isso nesta secção apresentamos alguns dos métodos e das tecnologias existentes para este problema. Começamos, na secção 2.2.1, por apresentar três métodos - Template Matching, Machine Learning e Deep Learning - que são centrais às tecnologias de análise e manipulação de fotografias digitais, que fazem reconhecimento de padrões faciais ou de objetos, que apresentamos nas secções seguintes.

2.2.1 Técnicas e Métodos

Nesta secção apresentamos três dos métodos que formam a base de algumas ferramentas existentes, atualmente, no mercado, direcionadas para a análise de fotografias.

Template Matching

Template Matching é uma técnica utilizada para identificar áreas numa fotografia que sejam semelhantes a uma fotografia de referência. Este método é fácil e intuitivo, consistindo em mover a fotografia de referência por todas as posições possíveis da fotografia alvo, exigindo que a fotografia de referência seja inferior à alvo, de forma a ser possível verificar se existe algum tipo de semelhança entre ambas. Esta comparação é feita pixel a pixel. Destas comparações resulta um índice de precisão, que se traduz no nível de semelhança entre a fotografia alvo e a de referência.

Ainda que exista uma ligeira diferença entre as fotografias que estão a ser comparadas, desde que essa diferença seja ligeiramente pequena a técnica template matching é efetiva. O template matching é usado normalmente para encontrar caracteres, números e objetos pequenos [16] [17].

Machine Learning

As técnicas de Machine Learning podem ser genericamente traduzidas na capacidade de uma máquina aprender sem ter de ser explicitamente programada. Para tal, são utilizados algoritmos para analisar uma grande quantidade de dados de entrada, aprender padrões presentes nos dados, e por fim fazer determinações e previsões relativas à nova informação analisada. Os nossos métodos de interesse são de aprendizagem supervisionada, uma vez que a informação fornecida ao modelo contém a resposta ao problema. a qual vai sendo cada vez mais precisa conforme recebe mais informação. Isto é, a sua precisão aumenta com o aumento dos dados de entrada que vai recebendo [18].

Assim, as técnicas de Machine Learning implicam um conjunto de algoritmos que analisam informação que lhes é passada, aprendem com essa informação, e por conseguinte baseiam-se no conhecimento que desenvolveram para tomar decisões.

Isto é, uma vez que aprenderam uma forma de tomar determinada decisão, eles podem aplicar a mesma decisão em problemas semelhantes. Quanto mais dados têm, mais precisa poderá tornar-se a decisão [19].

Deep Learning

O Deep Learning é um ramo de Machine Learning. A sua principal característica consiste em analisar continuamente a informação que lhe é fornecida, usando uma estrutura lógica semelhante à humana, de forma a conseguir tomar decisões. Para alcançar este propósito, o Deep Learning foi inspirado na função e estrutura do cérebro humano, nomeadamente nas interligações dos neurónios. Esta técnica tenta imitar a estrutura biológica do cérebro através de uma estrutura de redes neurais artificiais. Uma rede neural consiste em milhões de nós de processamento relativamente simples interligados entre si. O modo como é feita esta interligação é o que leva à aprendizagem e à capacidade de tomada de decisões de forma autónoma.

As técnicas de Deep Learning têm a capacidade de determinar, autonomamente, a precisão das suas decisões. Desta forma, este tipo de sistema consegue aprender através da sua própria computação, tornando-se mais abrangentes e não necessitando constantemente de guias para criar o seu conhecimento constante [19].

2.2.2 Photo DNA

Esta ferramenta, desenvolvida pela Microsoft, é um produto comercial pago. É direcionada para as entidades governamentais que lidam com o combate à exploração infantil, com principal foco na prevenção de pornografia infantil. Neste sentido, não é um software de reconhecimento facial e não pode ser utilizado para identificar pessoas ou objetos em fotografias, mas pode ser utilizado para encontrar fotografias semelhantes. O PhotoDNA procura identificar cópias de uma determinada fotografia alvo num conjunto de fotografias, através dos seus valores de hash. Isto é, primeiro é calculado o hash da fotografia alvo e de seguida o hash gerado é comparado com uma série de hashes de outras fotografias, de forma a identificar fotografias similares [20] [21].

O "ADN" de cada fotografia é obtido fazendo algumas alterações à fotografia. Especificamente a fotografia é convertida para uma escala de preto e branco, é redimensionada e dividida numa espécie de grelha. Para cada posição dessa grelha resultante é encontrado um histograma relativo à intensidade dos gradientes das margens, sendo o ADN da fotografia criado a partir desta informação relativa aos gradientes [22].

A Microsoft doou esta ferramenta ao Centro Nacional para o Desaparecimento e Exploração de Crianças, NCMEC, nos Estados Unidos. Este centro lida com questões relacionados com prevenção de vitimização de crianças e recuperação de crianças vítimas, incluindo rapto, abuso e outros tipos de exploração. Esta tecnologia, por não ser open-source, foi excluída da seleção das tecnologias para integrar na solução a ser desenvolvida.

2.2.3 OpenCV

A ferramenta OpenCV, desenvolvida originalmente pela Intel, em 2000, é um conjunto de bibliotecas open source de algoritmos e funcionalidades direcionados para a visão computacional, visando dar suporte ao tratamento de fotografias e reconhecimento de padrões. Esta biblioteca é constituída por um considerável número de algoritmos, otimizados incluindo algoritmos dedicados à visão computacional, mas também machine learning mais genericamente. Podemos encontrar, entre muitos outros, algoritmos para:

- Deteção e reconhecimento facial;
- Identificação de objetos;
- Classificação de ações de humanos em vídeos;
- Rastreamento de movimentos em câmaras;
- Extração de modelos 3D de objetos;
- Busca de fotografias semelhantes de uma base de dados de fotografias;
- Remoção dos olhos vermelhos de fotografias tiradas com flash.

A ferramenta tem uma compatibilidade considerável, com suporte para Windows, Linux, MacOS e Android, com interfaces C++, Python, Java, e ainda integra o MATLAB através da API do C++. Assim, o OpenCV e o MATLAB podem ser usados de forma complementar para o desenvolvimento de algoritmos de análise de fotografias e vídeos [24]. Com esta integração é possível explorar algoritmos implementados em OpenCV e MATLAB, utilizar os algoritmos OpenCV aproveitando as capacidades do MATLAB (como, por exemplo, acesso à informação, aquisição de fotografias e visualização), e também utilizar o MATLAB para explorar e analisar soluções que incorporem algoritmos do tipo OpenCV [25].

O OpenCV pode ser utilizado em diversos problemas de reconhecimento de padrões. No entanto, não existe uma solução que se possa aplicar ao nosso problema. Além disto, utilizar esta ferramenta exigiria a instalação de uma série de bibliotecas, tornando a solução complexa e dependente de atualizações de bibliotecas externas. Por estas razões, esta ferramenta não será considerada para a solução final.

2.2.4 Amazon Rekognition

A Amazon Rekognition é uma API não open-source baseada em deep learning desenvolvida por cientistas da Amazon. É uma ferramenta que auxilia outras aplicações a executar análise de fotografia e vídeo, sendo para tal apenas necessário fornecer à API da Rekognition a fotografia a analisar, e esta faz todo o trabalho de análise, retornando o resultado final.

Esta ferramenta permite identificar objetos, textos, pessoas, atividades e até conteúdo impróprio. Com o Rekognition é possível fazer análise facial com elevada precisão, dando a possibilidade de detetar e comparar faces para uma ampla variedade de utilizadores.

A API, além de analisar rapidamente qualquer fotografia ou vídeo, é também amiga do utilizador, sendo simples e fácil de usar. Além disso, o ritmo de atualizações é elevado, não só a nível de recursos para o reconhecimento facial, mas também estando em constante aprendizagem com os novos dados que vão surgindo, fazendo com que os seus resultados sejam cada vez mais precisos [28]. Apesar doutras vantagens, como é uma solução de código fechado e ser paga, foi excluída da seleção das tecnologias passíveis de serem parte integrante da solução a desenvolver.

2.2.5 YOLO, You Only Look Once

O YOLO é um detetor de objetos em fotografias e vídeos. Ao receber uma fotografia como dado de entrada, o YOLO retorna caixas delimitadoras à volta de cada objeto que reconheça, e classifica cada objeto encontrado consoante o tipo de classe ao qual pertence. O YOLO trata da deteção de objetos como um problema de regressão: ao receber uma fotografia aprende simultaneamente as coordenadas das caixas delimitadoras para os objetos e as probabilidades das classes a definir.

Um exemplo do resultado do uso desta ferramenta é apresentado na Figura 2.4 (retirada de [29]). Nela observamos a fotografia passada como entrada ao programa, mas com caixas de várias cores. Diferentes cores identificam diferentes classes. Como se pode ver, foram encontrados todos os objetos à exceção de um dos copos de vinho, o que mostra algumas limitações do YOLO.

Figura 2.4: Fotografia devolvida pelo YOLO [29]

Esta limitação está relacionada com a dificuldade de lidar com objetos pequenos, e com a incapacidade de manipular objetos agrupados. Na figura 2.4 a ferramenta não foi capaz de identificar os dois copos de vinho, uma vez que estavam demasiados

próximos, quase agrupados. Portanto, podemos concluir se a base de dados de fotografias incluir objetos pequenos ou muito próximos, o YOLO não será a ferramenta mais indicada.

Para reconhecer objetos esta ferramenta utiliza uma base de dados de fotografias anotadas que, através de deep learning, aprendeu a identificar uma determinada gama de objetos e animais. Esta base de dados, denominada por COCO, Common Object in Context, tem capacidade para identificar cerca de oitenta categorias, de entre elas: pessoas, animais, carros, bicicletas e aviões [29][30]. Dado o seu potencial, resolvemos testá-la.

Na Prática

Ao correr o programa, é necessário passar alguns argumentos, nomeadamente o caminho para a fotografia a analisar e a base de dados de fotografias que o mesmo deve utilizar para classificar os objetos que encontrar [29]. Na Figura 2.5 podemos verificar que a fotografia a analisar se chama *baggage_ claim.jpg* e se encontra na pasta *images*, e que a base de dados que o programa vai utilizar é a *yolo-coco*.

```
python3 yolo.py --image images/baggage_claim.jpg --yolo yolo-coco
```

Figura 2.5: Comando para realizar análise de fotografia pelo YOLO.

O resultado da execução da instrução da Figura 2.5 é apresentado na Figura 2.6.

Figura 2.6: Resultado apresentado pelo YOLO na execução da instrução da Figura 2.5

Como podemos analisar na Figura 2.6, o YOLO identificou oito pessoas presentes na fotografia passada como input, além de ter também identificado três malas de viagem e uma mala de senhora. Cada pessoa ou objeto identificado está dentro de uma caixa delimitadora com cores diferentes e essas cores, como podemos observar, variam consoante a categoria à qual o objeto pertence. Neste caso, amarelo se for pessoa, verde se for mala de viagem e cinzento para a mala de senhora. Juntamente com cada caixa podemos verificar que existe a identificação da categoria que o programa atribuiu, assim como um número que corresponde à precisão entre o objeto detetado e a categoria atribuída. Quanto mais próximo de 1, mais precisa foi a classificação.

Além da sua limitação de lidar com objetos pequenos e próximos, esta ferramenta seria uma boa aposta para incluir na solução final, se a base de dados de fotografias anotadas utilizada tivesse uma maior abrangência e incluísse as categorias armas, crianças e cartões de crédito. Desta forma, a ferramenta não será considerada para a solução final.

2.2.6 Keras

O Keras é uma API de redes neurais de alto-nível, escrita em python e capaz de correr sobre o Tensor Flow, uma API open-source de machine learning. Estas duas APIs dão a possibilidade de criar um modelo de aprendizagem que permita treinar uma solução para reconhecer objetos. Através desta API, foi possível criar um modelo de treino com três categorias cruciais, armas, cartões de crédito e crianças. Este processo divide-se normalmente em duas partes. Em primeiro lugar, cria-se um modelo de treino fazendo a aprendizagem sobre um conjunto de fotografias de referência. Depois, classificam-se as fotografias alvo baseado no modelo de treino criado no passo anterior [33].

Esta API, embora não seja especializada para solucionar o nosso problema, tem vantagens, nomeadamente, ser uma ferramenta open-source e ter múltiplas técnicas que permitem resolver o problema, tornando-a assim, em algo a considerar para a solução final.

2.3 Comparação das Tecnologias

Dado o estudo teórico e prático das tecnologias escolhidas, em ambas as vertentes, nomeadamente, análise de disco e análise de fotografias, por forma a simplificar e facilitar a análise das propriedades de cada uma, foi feita uma matriz de comparação, com propriedades consideradas relevantes para ambas. Desta forma será possível realizar uma leitura breve e sucinta, assim como ter uma base de comparação entre as tecnologias estudadas, uma vez que é fornecida uma visão geral de cada uma.

As propriedades que se podem encontrar na matriz são as seguintes:

- Fabricante: É importante reconhecer o fabricante, para se conseguir fazer uma análise sobre a sua visibilidade e imposição nos dias de hoje face ao mercado.

- Tipo de Licença: Esta é uma característica relevante, uma vez que no âmbito deste trabalho os programas escolhidos para efetuar as avaliações práticas terão de ser open-source, devido à possibilidade de acesso ao código fonte e ao custo monetário deste acesso ser nulo. Já para a parte teórica, isto é, para a análise e base de comparação, resolvemos incluir o software pago, uma vez que, ao terem investimento de capital e toda uma equipa dedicada, terão as funções mais requeridas e importantes para este tipo de programas.

- Formatos de imagem de disco que suporta: Uma ferramenta que suporte mais formatos será a que tem maior compatibilidade e abrangência para conseguir concluir o processo de análise, uma vez que pode receber e analisar a maioria das imagens de disco disponíveis. Uma ferramenta que tenha apenas suporte para um número reduzido de formatos, vai condicionar a amplitude do processo a esses mesmos formatos.

- Tipo de Interface: Característica que se refere à forma de utilização da ferramenta, se é executada via aplicação com interface gráfica, via terminal ou ambas.

- Tempo de análise de um disco de 16 GB.

- Tempo de análise de um disco de 180 GB.

- Base de dados: Para as tecnologias de análise de fotografias é relevante se têm ou não ligação/acesso a uma base de dados de fotografias de referência anotadas.

- Reconhecimento de padrões: Ainda na vertente de análise de fotografias, é necessário que a tecnologia a integrar a solução final contenha algoritmos especializados em reconhecimento de padrões.

Com base no estudo realizado e na tabela de comparação elaborada (Figura 2.7), será feita uma seleção quanto à tecnologia mais adequada para integrar na solução final.

	Tipo Ferramenta	Formato de imagens de disco	Tipo interface	Tempo análise de disco 16 GB	Tempo análise de disco 180 GB	Base de dados	Reconhecimento padrões	Fabricante	Licença
SANS SIFT	Análise de imagem de disco	E01, dd, AFF	Ambos	N/A	N/A	-	-	SANS	Open-source
The Sleuth Kit	Análise de imagem de disco	E01, dd	Gráfica	~30 segundos	~45 segundos	-	-	Basis Technology	Open-source
EnCase	Análise de imagem de disco	E01	Linha de comandos	N/A	N/A	-	-	Opentext	Pago
OpenCV	Reconhecimento de fotografias	-	-	-	-	Não	Sim	Intel	Open-source
YOLO	Reconhecimento de fotografias	-	-	-	-	Sim	Sim	YOLO	Open-source
Keras	Reconhecimento de fotografias	-	-	-	-	Possibilidade de criar	Sim	Keras	Open-Source
Photo DNA	Reconhecimento de fotografias	-	-	-	-	Sim	Sim	Microsoft	Closed-source
Amazon Rekognition	Reconhecimento de fotografias	-	-	-	-	Sim	Sim	Amazon	Closed-source
Recognizer	Ambas	E01, dd, img	Gráfica	~8 segundos	~90 segundos	Sim	Sim	-	N/A

Figura 2.7: Tabela das tecnologias analisadas.

Feita a análise e realizados os testes práticos às ferramentas, foi possível retirar conclusões e excluir algumas tecnologias que não se adaptavam à solução a desenvolver. Como podemos confirmar na tabela apresentada na Figura 2.7, todas as tecnologias analisadas, à excepção do **En Case, Photo DNA e Amazon Rekognition**, são open-source. Este é à partida um motivo de exclusão para estas tecnologias. A tecnologia **SANS SIFT** é uma workstation com várias tecnologias já instaladas o que a torna um pouco lenta e pesada. Uma vez que obter os resultados no menor tempo possível é um dos requisitos da solução, esta tecnologia foi também excluída.

A tecnologia **The Sleuth Kit** tem vários pontos positivos, nomeadamente a capacidade de analisar imagens de disco, aceder aos ficheiros da mesma e extrair fotografias, requisitos essenciais da solução a desenvolver. Tem, no entanto, problemas, em particular a dificuldade de integração com as tecnologias de classificação de fotografias. Por isso, optámos por desenvolver a nossa própria solução de extração de fotografias de uma imagem de disco, integrando-a com um mecanismo de classificação usando a framework Keras. Esta nossa solução, o Recognizer, é descrita no capítulo 4.

2.4 Conclusão

Neste capítulo introduzimos alguns tópicos e tecnologias que servem como contexto onde a tese se insere. Foram abordadas as vertentes de análise de disco e de classificação de fotografias, e foi apresentado um estudo relativo ao estado da arte de cada uma. O objetivo foi obter uma visão abrangente do que existe a nível de tecnologias e que pode ser um bom ponto de partida para o desenvolvimento da nossa solução. Além das descrições das tecnologias foram também feitos testes práticos a algumas das tecnologias estudadas.

Na vertente de análise de disco, analisámos a facilidade de utilização, os tempos de resposta, e as limitações das ferramentas estudadas. Para a vertente de análise e classificação de fotografias foram estudadas algumas tecnologias, open-source e pagas. A ferramenta que fará parte da solução a desenvolver será a API do Keras, devido às múltiplas funcionalidades que oferece, por ser open-source e pela sua facilidade de integração com outras tecnologias.

No próximo capítulo apresentamos a identificação e análise dos requisitos funcionais e não funcionais e respetivas especificações, que vão servir de base ao desenho e implementação da solução a desenvolver.

Capítulo 3

Requisitos

De modo a desenhar de forma efetiva a solução a desenvolver, é relevante identificar quais são os requisitos que a solução deve contemplar, incluindo as funcionalidades que devem ser implementadas e as suas características. Para isso, neste capítulo vamos identificar os requisitos funcionais, isto é, as funcionalidades da solução, e os requisitos não funcionais, nomeadamente o tipo de suporte tecnológico esperado e o desempenho.

3.1 Requisitos Funcionais

Estão identificados e detalhados na Tabela 3.1 os requisitos funcionais da solução a desenvolver. Estes requisitos incluem a montagem de uma imagem de disco, a extração de fotografias nela contidas e a sua classificação de acordo com quatro categorias: armas, crianças, cartões de crédito e outros. Faz também parte dos requisitos, a apresentação dos resultados ao utilizador numa interface amigável, de modo a facilitar a análise dos mesmos.

Tabela 3.1: Requisitos Funcionais - ID | Descrição

RF 1	Montar uma imagem de disco que esteja numa pasta local.
RF 2	Montar uma imagem de disco que esteja num dispositivo externo.
RF 3	Aceder às pastas e ficheiros da imagem de disco através de uma ferramenta de análise de disco.
RF 4	Realizar pesquisas sobre a imagem montada.
RF 5	Encontrar todas as fotografias contidas na imagem montada.
RF 6	Extrair todas as fotografias encontradas, contidas na imagem de disco.
RF 7	Armazenar o conjunto de fotografias alvo numa pasta específica.
RF 8	Lidar com dois conjuntos de fotografias diferentes, conjunto de fotografias alvo e conjunto de fotografias de referência.
RF 9	Analisar as fotografias do conjunto de fotografias alvo.
RF 10	Comparar o conjunto de fotografias alvo com o conjunto de fotografias de referência.
RF 11	Evidenciar semelhanças entre os dois conjuntos de fotografias.
RF 12	Reconhecer crianças em fotografias.
RF 13	Reconhecer cartões de crédito.
RF 14	Reconhecer armas.
RF 15	Reconhecer tudo o que não seja crianças, cartões de crédito e armas como outros.
RF 16	Receber uma imagem de disco e devolver os resultados gerados relativamente à precisão das fotografias numa interface amigável para o utilizador.

3.2 Requisitos não Funcionais

Os requisitos não funcionais recaem sobre especificações que a aplicação deverá suportar, relativamente ao tipo de imagem de disco e sistema operativo.

Todos os requisitos não funcionais estão identificados na Tabela 3.2.

Tabela 3.2: Requisitos Não Funcionais - ID | Descrição

RNF 1	Tem por base tecnologias open-source.
RNF 2	Executável em Linux.
RNF 3	Suportar imagem de disco do tipo dd, img e E01.
RNF 4	Suportar fotografias do tipo png, jpeg e jpg.
RNF 5	Realizar todo o processo em 20/25 segundos, para imagens de disco de aproximadamente 16 GB e $>$ 100 fotografias.

3.3 Especificações

Baseado nos requisitos funcionais e não-funcionais, na Tabela 3.3 apresentamos as especificações da solução, de modo a identificar a tecnologia de suporte usada para cumprir cada requisito, ou seja, o mapeamento entre os requisitos e a sua concretização. Por exemplo, para as tarefas de classificação de fotografias utilizámos a API do Keras.

Tabela 3.3: Especificações - ID | Descrição | RF ID que satisfaz

ESP 1	Obter uma imagem de disco através do desenvolvimento de um script em bash.	RF 1 - RF 7
ESP 2	Obter um conjunto de fotografias alvo localmente, através do desenvolvimento de um script em bash, e outro conjunto de fotografias provenientes de uma fonte externa.	RF 8
ESP 3	Utilizar a API do KERAS e tirar partido das suas funcionalidades.	RF 9 ao RF 11
ESP 4	Obter uma base de dados treinada especificamente para reconhecer armas, crianças e cartões de crédito.	RF 12 a RF 15.
ESP 5	Integração da ferramenta de análise de disco e da ferramenta de classificação de fotografias numa solução única: o Recognizer.	RF 10

3.4 Sumário

Neste capítulo identificamos os requisitos funcionais e não funcionais da solução desenvolvida, assim como as respetivas especificações. Na próxima secção iremos, além de apresentar o desenho da solução, descrever a implementação, identificando todos os passos do desenvolvimento da solução final.

Capítulo 4

Desenho e Implementação

Neste capítulo é apresentado o desenho da solução e os detalhes da sua implementação, incluindo discussão das principais decisões tomadas. O desenho define o modo como os vários módulos se interligam, e fornece uma visão geral da solução e do fluxo do programa.

4.1 Desenho da solução

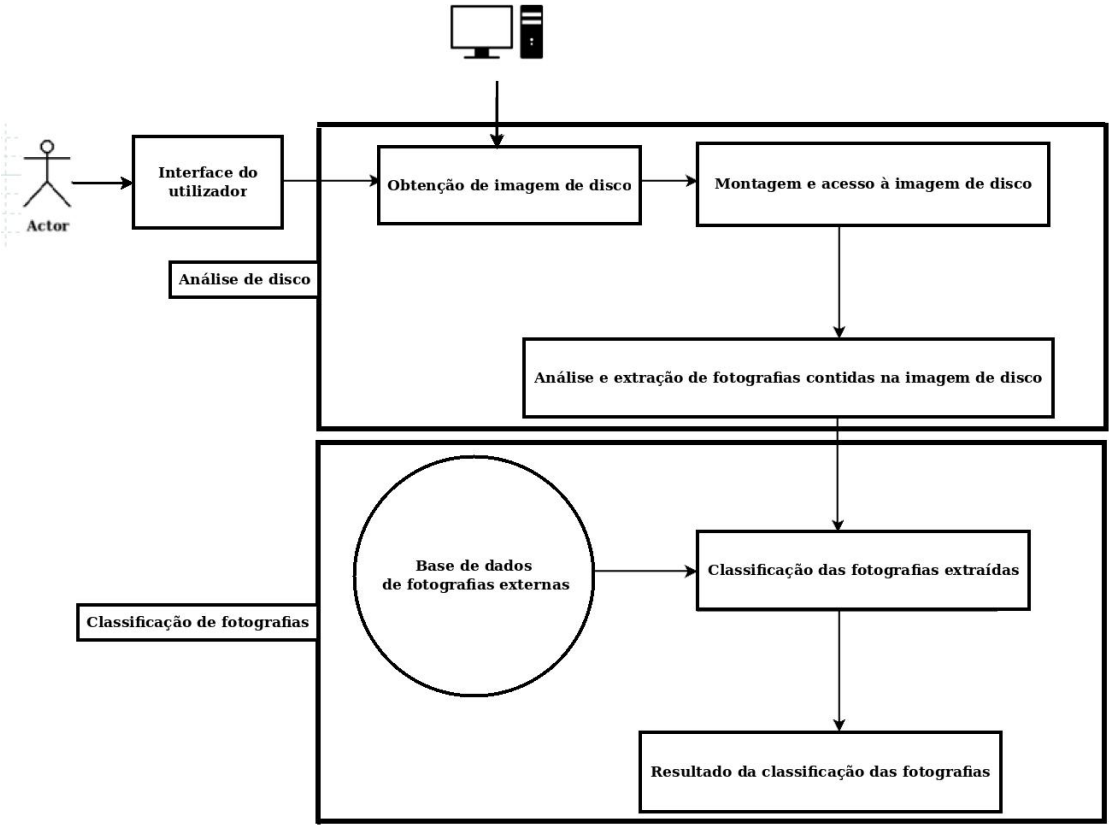

Figura 4.1: Desenho

Na Figura 4.1 apresentamos o desenho da solução, que é composto pelos módulos:

- Interface de utilizador
- Módulo de análise de disco e de recolha de fotografias
- Módulo para classificação de fotografias

A solução consiste na integração dos diversos módulos, como se pode observar na figura. O módulo de análise de disco é necessário para se aceder ao disco a analisar e recolher as fotografias nele contidas, que serão objeto de entrada para o módulo seguinte, identificado como módulo para classificação de fotografias. Neste, será feita uma classificação relativa ao nível de semelhança entre as fotografias alvo recolhidas e as categorias identificadas, nomeadamente: armas, crianças e cartões de crédito.

Para tornar esta solução amiga do utilizador, foi também pensado num módulo para a interface para o utilizador, simplificando a forma como a interação com a mesma é feita.

Resumindo, a ferramenta deve apresentar uma interface gráfica amigável para o utilizador que interliga as vertentes de análise de disco e classificação de fotografias. Com essa solução é possível obter todas as fotografias contidas na imagem de disco e classificá-las, de forma automática, facilitando a tarefa do técnico forense.

4.2 Implementação da solução

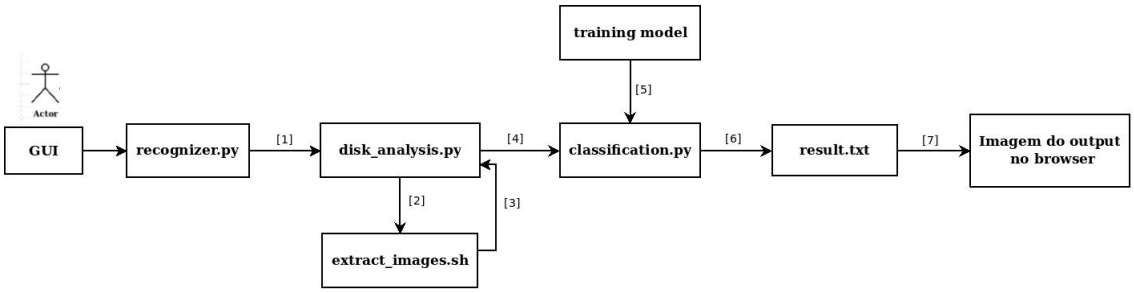

Figura 4.2: Implementação da solução

A interface para o utilizador é via browser. Depois de o programa ser iniciado, o utilizador é redirecionado para a página web da aplicação (Figura 4.2). Aí, é apresentado um campo de preenchimento que solicita a inserção do caminho para a imagem de disco que o utilizador pretende analisar (passo [1] na Figura 4.2). Depois de o utilizador ter submetido o caminho para o dispositivo, será feita uma análise e procura de fotografias no disco, e as mesmas serão extraídas para uma pasta temporária (passos [2] e [3] da Figura 4.2).

Após a recolha, segue-se o processo de classificação das fotografias recolhidas (passos [4] e [5] da Figura 4.2). Como resultado, é apresentada a precisão da classificação do objeto identificado, relativamente à categoria que lhe foi atribuída (passos [6] e [7] da Figura 4.2).

Apresentada a visão geral do programa, nas secções seguintes apresentamos os detalhes da implementação de cada módulo.

4.2.1 Interface gráfica

O ponto de entrada do nosso programa, escrito em python, dá acesso a uma página web onde será solicitado o caminho para o disco a analisar (Figura 4.3). Esta página foi projetada com simplicidade, para que o utilizador identifique rapidamente o que lhe é solicitado. A página tem apenas um texto informativo e um campo para o caminho do disco alvo de análise, que o utilizador deverá preencher e submeter (Figura 4.2, passos [1] e [2]). O programa irá guardar o caminho recebido, para ser utilizado pelo módulo responsável pela análise de dispositivos e recolha de fotografias, descrito a seguir.

Figura 4.3: Interface gráfica do Recognizer.

4.2.2 Solução de análise de disco e recolha de fotografias

A nossa solução para este módulo é um programa, escrito em bash, que executa as seguintes tarefas:

- Montar a imagem de um disco
- Aceder ao seu conteúdo
- Encontrar ficheiros do tipo fotografia (png, jpeg, jpg)
- Extrair as fotografias encontradas para um local específico.

O programa criado tirou partido dos comandos linux *dd, find, file, grep, awk* e *cp*. O objetivo é criar uma pasta com todos os ficheiros do tipo fotografia que existam na imagem de disco passada ao programa.

Na prática, o primeiro passo é montar a imagem de disco que é passada nos argumentos. Depois, é necessário encontrar todos os ficheiros do tipo fotografia, contidos na imagem que foi montada. Depois de os encontrar, copiar todos esses ficheiros para uma pasta específica, onde serão tratados.

Para conseguir identificar todas as fotografias, foi utilizado como referência a extensão de ficheiros. Para tal foi usado o comando *find* com a indicação do caminho para o local onde incide a pesquisa, com a opção *-name*, para que possamos procurar pela extensão *png, jpeg* ou *jpg* contida no nome. O resultado desta instrução será todos os ficheiros que se encontrem no local especificado, com a extensão passada como argumento.

Na Figura 4.4, mostramos um exemplo do comando executado e o respetivo resultado.

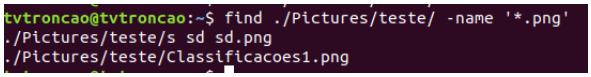

Figura 4.4: Comando *find* via linha de comandos.

Uma questão que foi tida em consideração foi o facto de o nome do ficheiro poder não conter uma extensão, apropriada, mas ainda assim ser uma fotografia. Para colmatar esta situação houve a necessidade de usar o comando *file*, que nos fornece a informação base do ficheiro, não dependendo do seu nome. Por exemplo, ao executar o comando *file* sobre a fotografia que não foi abrangida no resultado no comando anterior, *file ./Pictures/teste/Classificacao2 -b*, obtemos informações relativamente à mesma como o tipo de fotografia, as dimensões e o sistema de cores. Podemos verificar que, embora não contenha a extensão no seu nome, é de facto uma fotografia do tipo PNG.

A Figura 4.5 identifica o comando executado e respetivo resultado.

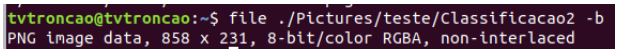

Figura 4.5: Comando file via linha de comandos.

Tirando partido deste comando *file*, o objetivo é fazer uma pesquisa sobre o resultado que este retorna. Para tal, usámos o comando *grep* aplicado ao resultado

do comando file. O comando *grep* permite pesquisar uma determinada palavra, ou conjunto de palavras. Neste caso, a pesquisa tem por base as palavras "image data". Os resultados obtidos com o comando *grep*, são os ficheiros com imagens, que serão copiados com o comando *cp*.

Em suma, o processo realiza-se com os seguintes passos:

- Montar a imagem de disco recebida para a pasta local
- Criar uma pasta temporária para onde serão copiadas todas as fotografias encontradas no disco a analisar.
- Criar um ficheiro de texto temporário com o nome *listoffiles* na pasta onde o script se encontra, onde serão gravados os nomes de todos os ficheiros encontrados no disco a analisar. Para obter esta lista de ficheiros foi utilizado o comando **find** com a opção **type** definida para **f**, o que representa uma procura de ficheiros.
- Criar um ficheiro de texto temporário com o nome *listoftypes* na pasta corrente, onde serão gravados todos os tipos de ficheiros relativos aos encontrados no passo anterior. Uma vez que o comando **file** não tem opção de recursividade entre pastas, foi necessário criar o anterior *listoffiles*, para que o comando **file** conseguisse extrair os tipos dos ficheiros encontrados. Para facilitar pesquisas futuras e conseguir distinguir um tipo de ficheiro de outro, foi definido um separador entre os ficheiros tirando partido da opção *separator* que o comando **file** disponibiliza, alterando o separador de **:** para **:end:** .
- Criar um ficheiro de texto temporário com o nome *listimagedata* na pasta onde o script se encontra, onde serão gravados todos os ficheiros que contiverem *image data* na definição. Para conseguir obter esta lista foi utilizado o comando **grep** que vai percorrer todas as entradas do ficheiro criado no passo anterior, *listoftypes*, e procurar em cada entrada **:end: *?.?*image data**. Isto é, faz-se uma análise sobre os ficheiros que contêm na sua definição *:end:* seguidos de *um ou mais espaços* e por *image data*.
- Criar um ficheiro de texto temporário com o nome *imagestocopy* na pasta corrente, onde vão ser gravados os nomes dos ficheiros que constam no ficheiro anterior, mas cada linha será cortada pelo delimitador **end**, de forma a obtermos uma lista limpa e sem informação auxiliar desnecessária. Para esta transformação foi utilizado o comando **awk** que procura o segmento **:end:** e guarda a informação que estava antes do mesmo, obtendo desta forma apenas o nome completo do ficheiro. Neste ficheiro de texto vamos encontrar todos os caminhos para as fotografias, que vamos copiar para a pasta criada inicialmente, e que serão analisadas no módulo seguinte.
- É feita a cópia de cada fotografia para a pasta criada inicialmente. Para isso foi utilizado o comando **cp**, dentro de um ciclo sobre o ficheiro *imagestocopy*, que copia cada linha deste para a pasta de destino em questão.
- Por fim, é feita a remoção de todos os ficheiros temporários.

4.2.3 Solução para classificação de fotografias

Para a classificação de fotografias foi necessária a implementação de um modelo de treino que conseguisse classificar a gama de objetos definidos para identificação. Para tal, recorremos a algoritmos de machine learning, mais especificamente através da API do Keras, descrita na secção 2.2. A gama de objetos definidos para esta classificação recaiu sobre crianças, cartões de crédito, armas e outros (as quatro *categorias*). Sendo o nosso objetivo classificar corretamente as três primeiras categorias, a última serve apenas para identificar tudo aquilo que não é classificado nas três anteriores.

Esta parte do processo está dividida em duas fases:

- A primeira fase consiste na criação de um modelo de treino. Este modelo resulta de dois passos. Primeiro, fazer várias iterações sobre o conjunto de fotografias de referência. Segundo, fazer a validação do treino feito. Estas iterações dependem da forma como o modelo foi construído, ou seja, das camadas que o constituem. Nesta fase, o dataset de fotografias utilizado corresponde ao conjunto de fotografias de referência, que foram identificadas com as categorias: pessoas, cartões de crédito, armas e outros.

- A segunda fase consiste na classificação de fotografias. Esta fase tira partido do modelo treinado na fase anterior e requer também um conjunto de fotografias alvo para classificação.

Para a construção do nosso modelo de treino, no decorrer da primeira fase foi necessário obter o dataset de fotografias de referência. Para esse fim, e por não nos ter sido facultado um conjunto de fotografias de referência por fonte externa, como era previsto inicialmente, foi necessário recorrer ao motor de pesquisa google e descarregar as fotografias resultantes de pesquisas para crianças, cartões de crédito, armas e outros.

Foram definidos datasets com mil e duzentas fotografias de cada categoria, das quais 75% são consideradas como grupo de fotografias de teste e 25% são consideradas como grupo de fotografias de validação, escolhidas aleatoriamente.

Passo 1. Preparação do input do modelo

Antes de se criar o modelo propriamente dito, existe uma fase de preparação da informação que lhe será passada. Aqui, o modelo recebe como input o conjunto de fotografias de referência e é sobre este que vai incidir a aprendizagem do modelo. O conjunto está organizado por pastas, sendo que cada pasta corresponde a uma categoria de interesse.

Embora organizadas, estas fotografias têm dimensões variadas, o que requer que ainda antes de se seguir para a fase de treino do nosso modelo se faça um tratamento relativo às mesmas, pois o modelo é definido para receber fotografias de uma determinada dimensão apenas. Assim sendo, todas as fotografias deste conjunto foram redimensionadas para as dimensões a passar ao modelo, 32x32, que são dimensões pequenas para se diminuir o tempo de processamento, de forma a que se consiga lidar com grandes quantidades de fotografias de forma eficiente.

Depois deste ajuste é necessário fazer um mapeamento entre a fotografia e a categoria correspondente, uma vez que a categoria está presente no caminho da fotografia, já que foi extraída do mesmo. Esta associação é importante pois o modelo quando está a ser criado precisa de saber que todas as fotografias dentro de uma determinada pasta pertencem à categoria cujo nome da pasta a identifica.

Para auxiliar esta associação, usamos a classe **label binarizer** do Keras. Esta permite criar uma matriz identificadora de categorias a partir de uma lista. Isto é, dado o input de fotografias e o input das categorias, esta função gera como output uma matriz binária, onde atribui 1, se a fotografia se identifica naquela categoria, ou 0, caso contrário. Na Figura 4.6 encontramos um exemplo simplificado do resultado da aplicação da função label_binarizer [43].

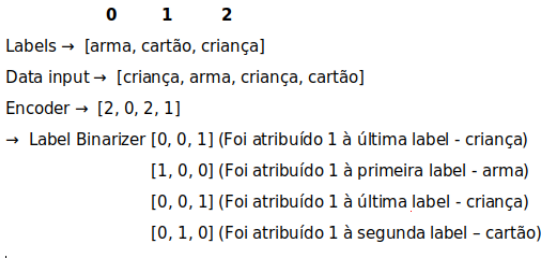

Figura 4.6: Exemplo simplificado da aplicação da função **label binarizer**. Tendo uma lista de categorias definida e uma lista de dados de entrada, é feita uma análise a cada item de entrada relativamente às categorias. Isto é, vamos ver a que posição da lista a fotografia recebida corresponde, e depois deste processo estar feito para todos os itens de entrada, é convertido para uma matriz binária. Uma linha corresponde a cada item de entrada e cada coluna corresponde a cada categoria. Caso seja 1, representa a categoria em questão, caso contrário será 0.

O treino do modelo requer que exista um conjunto de fotografias de treino, mas também um conjunto de fotografias de teste. Para esse efeito, é feita uma divisão no conjunto de fotografias de treino ficando uma parte para treino e outra para teste. As dimensões destes dois conjuntos foram definido para 75% - 25%, ficando o valor maior para treino e o menor para teste. Esta divisão foi feita com a função **train_test_split**, que recebe como argumentos o local onde estão armazenadas as fotografias, as categorias, e a fração que é pretendida para o conjunto de teste (neste caso, 25%). Esta função faz a escolha dos elementos aleatoriamente. Desta forma, os conjuntos resultantes não seguem a ordem que tinham originalmente [53].

Passo 2. Criação do modelo

Nesta fase tirámos partido novamente da API do Keras, que disponibiliza uma vasta gama de *layers* (designada futuramente por camadas), que podem ser conjugadas e aplicadas consoante o alvo de treino. De forma a obter-se o melhor resultado em termos de precisão foram necessárias várias experimentações, conjugando as camadas de formas diferentes. Esta API permitiu a construção de um modelo de treino

adequado às categorias definidas com um output cuja precisão média de teste foi de aproximadamente 84%, como veremos no Capítulo 5.

Para criar um modelo, é necessário, primeiramente, definir o tipo de modelo que queremos elaborar. Para este caso, foi escolhido o modelo sequencial, de forma a ser possível adicionar camadas ao mesmo até que o output corresponda ao que é procurado. Para se chegar ao melhor output foi necessário realizar várias experiências, através do método de tentativa-erro, experimentando cada camada, variando os argumentos de cada uma (como por exemplo o kernel_size, a função de ativação, e até mesmo a ordem das camadas). Estas alterações influenciam o output isolado de cada camada, o que irá, consequentemente, afetar o output das camadas seguintes, levando a que o resultado seja diferente de configuração para configuração.

Assim sendo, podemos concluir que treinar um modelo é encontrar o melhor conjunto de pesos, ou seja, a configuração de camadas, para mapear inputs para outputs num determinado conjunto.

A primeira camada de um modelo necessita de receber informação relativa às dimensões do input, ou seja, necessita que lhe sejam passadas as dimensões das fotografias, sendo que as camadas seguintes já não requerem receber diretamente essa informação, uma vez que conseguem inferir a partir da camada anterior. A primeira camada é responsável por definir o input, ou a camada visível, e de definir a primeira hidden layer [34]. Uma fotografia é traduzida numa matriz de pixéis, e é sobre essa matriz que as camadas fazem o seu processamento, gerando outras matrizes como output, com base na que lhe é passada.

Após várias tentativas para alcançar o modelo cujo output tivesse a melhor precisão possível, foi construído um modelo que teve por base camadas dos tipos **Conv2D, BatchNormalization, MaxPooling2D, Flatten, Dropout e Dense.** Segue-se uma descrição de cada camada, detalhando a sua função e a razão dos seus argumentos e outputs.

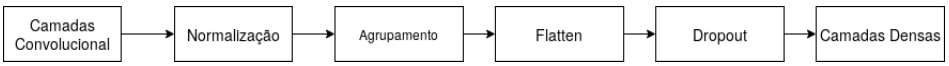

Figura 4.7: Diagrama da rede convolucional

Camada 1. Conv2D(32, (3, 3), input_shape=(32, 32, 3), activation="relu", kernel_regularizer=l2(0.01))

Conv2D pertence à classe das camadas convolucionais, conhecidas por serem apropriadas para classificação de fotografias. Estas aplicam uma série de filtros à fotografia que lhe é passada, fazendo percorrer uma matriz filtro sobre a mesma. Por cada área que a matriz abrange são feitas várias operações matemáticas que resultam num único valor na matriz de output. Por fim, é aplicada uma função de ativação.

Conv2D é vista como uma matriz bidimensional, que se envolve com o input, conseguindo alterá-lo de forma a gerar um output, que é outra matriz de onde é possível extrair mais informação. São feitas operações matemáticas para que consigamos alterar a função original de forma a conseguir obter mais informação

de camada para camada. Como esta camada está a ser utilizada como a primeira do modelo, ela tem de receber como input o argumento **input_shape** que está definido para (32, 32, 3), o que corresponde às dimensões das fotografias. Neste caso, tratam-se de fotografias de 32x32. O último argumento (3) diz respeito ao sistema de cores (tratam-se de fotografias RGB, isto é, com três canais, **red, green** e **blue**, o que significa que são fotografias a cores).

O primeiro argumento que é passado para esta camada, diz respeito ao número de filtros que vão passar pela fotografia a analisar e que serão gerados após a camada ser processada. O segundo argumento refere-se às dimensões do kernel, (**kernel_size**), que definem a altura e a largura da matriz filtro que passará pelas fotografias. As dimensões podem ser (1,1), (3,3), (5,5) e (7,7), mas foram definidas (3,3), uma vez que a dimensão das fotografias é inferior a 128x128. Caso fossem superiores, o **kernel_size** deveria ser superior a (3, 3) para os filtros terem maior abrangência a nível espacial e para ajudar a reduzir o volume de informação.

O terceiro argumento é referente à funcão de ativação, **activation**. Neste caso, foi definida a função relu, **"Rectified Linear Unit"**, sendo que esta é a que deve ser aplicada depois de ter sido realizada a convolução. Esta função de ativação é aplicada para que se consiga introduzir não-linearidade no modelo. Relu é a função de ativação mais utilizada atualmente e, embora ainda não seja perfeita, é a mais confiável [40].

O quarto argumento é relativo ao **kernel_regularizer**, que é um regularizador que permite aplicar penalizações às camadas, ajudando a reduzir o *overfitting*. Esta é uma técnica que realiza pequenas modificações ao algoritmo de aprendizagem, de forma a que o modelo generalize melhor, promovendo assim o desempenho deste relativamente a informação que nunca viu. A função definida para este argumento foi **l2**, que representa uma função de perda que tenta minimizar o erro do modelo. Esta regularização penaliza o somatório do quadrado dos pesos [54] [55].

As primeiras camadas aprendem menos do que as camadas mais próximas do output final, por essa razão foram aplicadas várias camadas deste tipo, tendo as primeiras menos filtros do que as últimas. Ou seja, o número de nós vai aumentando com a profundidade da rede. Temos duas camadas Conv2D de 32 nós, sendo que entre elas temos uma camada de normalização e outra de pooling. A seguir à segunda camada Conv2D de 32 nós temos novamente camadas para normalização, pooling e também de dropout. Depois destas foi adicionada uma camada Conv2D de 64 que seguiu o processo igual à primeira camada Conv2D 32. No total, temos 6 camadas Conv2D, duas para cada número de nós escolhido (32, 64 e 128), tendo sempre camadas de normalização e pooling entre elas, e de dropout entre camadas de diferentes nós. Estas seis camadas estão separadas por outras camadas, por forma a introduzir normalização e reduzir a dimensão do output de cada camada anterior. Foi utilizada esta estrutura de forma a conseguir obter a melhor aprendizagem possível. Na Figura 4.7 ilustramos esta arquitetura.

Camada 2. BatchNormalization()

Com esta camada estamos a tentar regularizar o output gerado anteriormente, pelo que esta função é responsável por normalizar a ativação da camada anterior.

Isto é, ela aplica uma transformação de forma a manter a média da ativação próxima de 0 e o desvio padrão da ativação próximo de 1. Para tentar estabilizar o modelo, esta camada normaliza o output da anterior subtraindo a média de ativação e dividindo pelo desvio padrão [56].

Camada 3. layers.MaxPooling2D((2, 2))

MaxPooling2D pertence à classe das camadas de agrupamento, que têm como função a redução da resolução do output da camada Conv2D anterior, de forma a reduzir o tempo de processamento. A função Max Pooling, que consiste numa matriz cujas dimensões lhe são passadas nos argumentos, vai iterar sobre cada posição e retirar o valor máximo dos pixéis abrangidos pela matriz. À medida que vai obtendo os valores máximos de cada iteração, vai construindo uma nova matriz com base nos mesmos. Na Figura 4.8 está ilustrado o resultado da aplicação da função em questão, sendo que foi possível reduzir as dimensões da matriz inicial para metade [40].

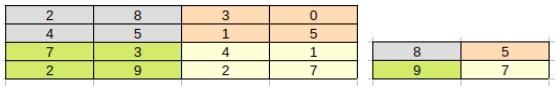

Figura 4.8: Resultado da aplicação da função MaxPooling. Do lado esquerdo temos a matriz inicial (alusiva à fotografia inicial) e do lado direito temos a matiz resultante da aplicação do MaxPooling 2x2.

Camada 4. Flatten()

Esta camada tem o objetivo de preparar o output da camada anterior, Conv2D, para a camada Dense que se segue. Isto é necessário uma vez que o output da camada anterior é multidimensional, está na forma 4-D, ou seja, (x,y,z,w), e a camada que se segue só consegue receber inputs da forma uni-dimensional, 1-D, (x*y*z*w,). Então é utilizado o Flatten, para que faça esta transformação e se consiga aplicar camadas do tipo Dense ao output da camada Conv2D. No fundo, esta camada faz a ponte entre a Conv2D e a Dense [36].

Camada 5. (layers.Dropout(0.5))

Esta camada, Dropout, é uma técnica de regularização que evita a questão de **overfitting**, que acontece quando um modelo está muito bem treinado para o conjunto de fotografias de treino, mas não se comporta como é esperado para o conjunto de fotografias de teste, ou para as que ainda não viu. Esta função recebe uma fração, que varia entre 0 e 1, e é esse número que dita quantos "neurónios" serão apagados, fazendo com que os neurónios que ficam tenham de lidar com a ausência daqueles que foram apagados. Os que são apagados são escolhidos de forma aleatória [37] [38].

Camada 6. Dense(1024, activation="relu")

Dense pertence à classe das camadas densas. Estas realizam uma classificação das características que foram extraídas pelas camadas convolucionais e reduzidas pelas camadas de agrupamento. Estas camadas densamente ligadas são definidas usando a classe Dense do Keras. É uma camada de nós regular numa rede neural, onde cada nó recebe informação de todos os nós da camada anterior, estando desta forma todos os nós densamente ligados [39] [40].

O primeiro argumento recebido por esta camada é o número de nós, neste caso a camada densa terá 1024 nós e o segundo argumento diz respeito à função de ativação. A função definida é a relu, de forma a ser introduzida não-linearidade no modelo.

Camada 7. Dense(512, activation="relu")

Esta é uma camada com descrição igual à camada anterior, sendo que as alterações recaem sobre os argumentos de entrada. Esta camada terá apenas 512 nós e a função de ativação será a mesma que a da camada anterior.

Camada 8. Dense(len(lb.classes_), kernel_regularizer=l2(0.05), activation="softmax")

Finalmente, esta camada com descrição igual à anterior, terá o número de nós igual ao número de categorias para o qual o modelo foi treinado. Neste caso, como são quatro categorias(armas, cartões de crédito, crianças e outros) teremos então uma camada densa de quatro nós. Relativamente à função de ativação foi escolhida a *softmax*, já que esta função normaliza os valores de input num vetor onde valores seguem uma distribuição probabilística cujo somatório é igual a 1. Assim sendo, a aplicação desta função vai gerar um número entre 0 e 1 para cada nó da camada, fazendo com que a soma de todos os nós resultem em 1.

O resultado desta função pode ser interpretado como uma medida que determina a probabilidade de uma fotografia pertencer a cada uma das categorias, isto é, para cada fotografia a ser analisada irá resultar uma probabilidade de fazer parte de cada uma das categorias existentes [41] [42].

Passo 3. Compilação do modelo

Para compilar o modelo foi usada a função **compile** da API do Keras. Esta função recebe como argumentos de entrada:

(loss="mean_squared_error", optimizer=SGD(lr=0.01, momentum = 0.9), metrics=["accuracy"])

- loss: é um argumento obrigatório para a função compile e corresponde a uma função de perda ou de custo. O resultado desta função traduz-se no número que é proveniente da análise de todos os aspetos do modelo. O objetivo é melhorar, neste caso, minimizar o quanto possível este número, ou seja, a perda do

modelo, para se obter um modelo melhor. A escolha desta função deve ser cuidadosa, uma vez que tem de ir de encontro aos objetivos do modelo, pois caso não se enquadre no que é esperado, a perda do modelo pode ser elevada, não por culpa da função, mas sim pela escolha da função a usar não ser a mais apropriada. A função escolhida para este argumento foi **mean_squared_error**, a função mais utilizada em problemas de regressão, a qual é calculada como a média do quadrado da diferença dos valores de treino e de teste. Isto faz com que diferenças grandes nos conduzam a mais erro/perda do que diferenças pequenas. Consequentemente, o modelo é "castigado"por cometer grandes diferenças entre os valores de treino e de teste, uma vez que resulta em maior erro. Quanto menor esta média for, mais próximo estamos de obter a linha que melhor se encaixa no modelo [45] [46].

- optimizer: é um dos argumentos obrigatórios para a função **compile**, sendo usada para atualizar iterativamente os pesos da rede com base nos dados de treino. O optimizador escolhido foi o **SGD** - *Stochastic gradient descent*, que recebe como argumentos uma taxa de aprendizagem, *lr* - *learning rate*, e um *float* de valor superior ou igual a zero, que acelera o optimizador na direção relevante e ajuda a reduzir as oscilações [49]. A taxa de aprendizagem pode variar entre 0 e 1, sendo que esta taxa controla a rapidez com que o modelo se adapta ao problema: uma taxa pequena vai fazer pequenas modificações nos pesos a cada iteração, e por essa razão taxas pequenas requerem mais iterações, ou seja, mais epochs. Pelo contrário, taxas grandes levam a mudanças mais rápidas o que, consequentemente, requer menos iterações. Este argumento controla quão rápido ou lento é a aprendizagem do modelo. É adicionada ainda inércia ao procedimento de atualização, para que várias atualizações do modelo continuem na mesma direção, ajudando o modelo a aprender mais rápido, ao manter-se na direção correta de aprendizagem. Este tem em consideração os gradientes anteriores para suavizar as iterações com descida de gradiente [57].

- metrics: Metrics é uma função que é utilizada para avaliar o desempenho do modelo. Os valores resultantes são registados no final de cada iteração sobre os dados de treino. Esta função baseia-se no número de previsões corretas sobre o total de previsões feitas, e depende do número de elementos que temos para cada categoria. Caso tenhamos 90% de elementos da categoria A e 10% da categoria B, facilmente chegamos a uma precisão de 90%, uma vez que será previsto que 90% dos elementos sejam da categoria A. Para existir uma boa utilização desta métrica o ideal é ter o mesmo número de elementos para cada classe [50].

Passo 4. Treino e avaliação do modelo

(trainX, trainY, validation_data=(testX, testY), epochs=EPOCHS, batch_size=32)

Com o modelo compilado, podemos passar à fase de treino do modelo, sendo que para este passo vamos tirar partido da função **fit**, do Keras. Esta função é usada quando a RAM consegue lidar com todo o conjunto de treino, não sendo

necessário utilizar geradores do Keras. Para conjuntos de treino muito grandes é necessário tirar partido da função **fit_generator**, que exige aumento de dados uma vez que a RAM não consegue lidar com o conjunto de treino dada a sua dimensão, evitando assim o overfitting (é também uma forma de regularização, ou seja, leva a que o modelo generalize melhor). A função **fit** irá receber os conjuntos de treino e teste que já foram previamente divididos como argumentos de entrada, a variável **epochs**, ou seja, o número de iterações que o modelo deverá fazer, e por último o **batch_size**.

Relativamente aos conjuntos de treino e teste, é necessário passar, além do conjunto de fotografias a analisar, o conjunto de categorias. O número de **epochs** tem de ser definido antes de treinar o modelo, pois o modelo necessita saber quantas vezes deve iterar sobre o conjunto de entrada. Foram definidos 100 **epochs** e o **epoch** inicial é zero, ou seja, começa no índice zero e irá iterar até alcançar o índice 100 [52] [51]. Foram realizados testes com maior número de epochs, e os resultados foram semelhantes, uma vez que a perda e a fiabilidade tendem a estabilizar por volta da vigésima iteração.

Após o nosso modelo estar treinado, temos o modelo finalizado e é desejável realizar sobre ele uma previsão, sendo que para isso foi utilizada a função **predict** da API do Keras. Esta função gera previsões sobre o conjunto de teste. O modelo nunca viu este conjunto e, por essa razão, vamos utilizá-lo para fazer algumas previsões de algo que ele não viu na fase de treino [51].

Após estas fases, o modelo está treinado e testado, pode ser guardado para que se possa utilizar em futuras classificações de fotografias para o qual o modelo foi criado. Para guardar o modelo foi utilizada a função **save** do Keras, que o guarda na pasta especificada, passada via argumento. Esta função gera um ficheiro HDF5 que contém a arquitetura, pesos e configurações do modelo, bem como as funções de loss e optimizer.

Passo 5. Classificação

Com o modelo criado e guardado, podemos classificar fotografias com o mesmo. Para isso é preciso passar um conjunto de fotografias alvo ao programa destinado à classificação. Este carrega o modelo e gera uma precisão para cada uma das fotografias recebidas.

Antes da previsão da precisão de cada elemento de input, é necessário tratar cada fotografia para que se enquadre com a gama de fotografias que o modelo consegue analisar. Assim sendo, cada fotografia foi redimensionada para 32x32, uma vez que o modelo foi treinado em fotografias da mesma dimensão.

Depois deste ajuste, passamos à fase de previsão, sendo que para isso tiramos partido da função **predict** do Keras. Esta função gera previsões para as quatro categorias disponíveis, sendo necessário obter a posição onde a precisão foi maior e, a partir desse índice, verificar a que categoria corresponde. A categoria correspondente será aquela à qual a fotografia analisada pertence. Depois de analisarmos todas as fotografias do conjunto alvo, temos identificada uma categoria para cada uma, associada a uma precisão.

4.3 Sumário

Neste capítulo apresentamos o desenho da solução assim como a sua implementação, sendo detalhados os passos necessários para o desenvolvimento da solução final. No próximo capítulo iremos apresentar os resultados obtidos na avaliação da solução desenvolvida, explicando os testes realizados à mesma e discutindo os seus principais resultados.

Capítulo 5

Avaliação da solução

Neste capítulo são apresentados os testes realizados à aplicação e uma discussão dos resultados gerados. São consideradas quatro métricas: falsos positivos, verdadeiros positivos, fiabilidade e perdas.

5.1 Setup experimental

O desenvolvimento e validação do modelo de aprendizagem exigiu dois conjuntos de fotografias, um para a sua criação e o outro para a sua avaliação, definidos como conjunto de fotografias de referência e conjunto de fotografias alvo, respetivamente. Estes são descritos a seguir.

5.1.1 Dataset para criação do modelo de aprendizagem - Conjunto de referência

As fotografias para gerar o dataset de treino foram obtidas através do motor de busca Google. Foram obtidas 1200 fotografias para cada categoria, num total de 4800 fotografias. Foram escolhidas quatro categorias, de forma a identificar armas, cartões de crédito e crianças. A quarta categoria inclui tudo aquilo que não pertence às outras três, englobando assim a categoria "Outros". Esta categoria é constituída por uma série de fotografias díspares, escolhidas aleatoriamente.

5.1.2 Dataset para avaliação do modelo de aprendizagem - Conjunto alvo

Este conjunto, constituído por 178 fotografias, foi também obtido aleatoriamente através do motor de busca Google, uma vez que não foi possível obter dados de organizações externas para testar o modelo, como estava previsto inicialmente.

Estas fotografias representam assim as que seriam recolhidas do computador de um suspeito, e que são objeto de análise pela nossa solução.

5.2 Definição das métricas usadas

Foram escolhidas as seguintes métricas para avaliar a nossa solução:

- Fiabilidade("accuracy"): Representa o nível de confiança na classificação obtida pela solução. Isto é, fotografias com nível de fiabilidade elevado têm mais probabilidade de estarem bem classificadas.
- Perda("loss"): Representa as perdas que existiram na criação do modelo, durante o processo de aprendizagem.
- Verdadeiros positivos(VP): Representam todas as fotografias que foram classificadas corretamente na categoria à qual pertencem.
- Falsos positivos(FP): Representam todas as fotografias que foram incorretamente classificadas em dada categoria.
- Precisão: Representa as fotografias corretas do total de fotografias avaliadas pela solução. Calculada com base na fórmula P = VP/(VP+FP).

Foi com base nas duas primeiras métricas que foi escolhido o modelo de aprendizagem. As últimas três métricas foram usadas para avaliar a solução.

5.3 Escolha do modelo de aprendizagem

Para escolher o melhor modelo de aprendizagem a utilizar na solução final, foram testados vários modelos de treino. Especificamente, foram testadas camadas convolucionais diferentes, e variações da sua ordem. Depois da escolha das camadas convolucionais e respetiva ordem, foram realizados alguns testes nas funções de perda e nos otimizadores. O objetivo de todas estas experiências prende-se com a procura do modelo com a precisão mais elevada.

Depois de vários testes, o modelo escolhido tem a seguinte configuração: camada convolucional, **Conv2D**, camada de normalização, **BatchNormalization**, camada de pooling, **MaxPooling**, camada **Flatten**, camada **Dropout** e por fim três camadas densas, **Dense**.

A precisão do modelo, para o conjunto de fotografias de referência (o conjunto com 4800 fotografias), foi de aproximadamente 84%, com uma perda de cerca de 13%. O gráfico apresentado na Figura 5.1 mostra a evolução da precisão do modelo, durante a sua criação. Além disso, mostra também a diminuição da perda ao longo das iterações do treino. O eixo do **x** representa o número de iterações. Neste caso, o modelo foi treinado em 100 iterações. O eixo do **y** representa a precisão/perda. O último passo do processo de criação do modelo corresponde ao treino e validação do modelo. Neste passo, o conjunto de fotografias de referência (4800) é dividido em duas partes: 75% das fotografias são usadas para treino (o que corresponde a 3600 fotografias) e 25%, que corresponde a 1200, fotografias usadas para validação do modelo.

No gráfico estão representadas quatro medidas diferentes, analisadas no processo de criação do modelo, sendo: perda no treino, perda na avaliação, precisão no treino e precisão na avaliação. Conseguimos observar que ambas as linhas que identificam as **perdas**, caem para menos de 50% antes da vigésima iteração e ambas as linhas representativas da **precisão** sobem para valores acima dos 70% a partir, também, da vigésima iteração.

Figura 5.1: Gráfico com a progressão da precisão e da perda ao longo das iterações durante o treino do modelo. Foram definidas 100 iterações.(Recordar secção 4.2.3)

Ainda com base na Figura 5.1, podemos verificar que a linha que representa a precisão da fase de avaliação tende a aproximar-se de 1, ou seja, a precisão do modelo esteve em média sempre a progredir, tendo estabilizado ao fim de 60 a 70 iterações nos 84%. Podemos ainda verificar que a função de perda tende a aproximar-se de 0, estabilizando ao fim de 20 a 30 iterações nos 13%. Segundo a mesma figura, é possível verificar que as métricas para treino e para avaliação estão muito semelhantes.

Na próxima secção vamos apresentar os resultados obtidos dos testes realizados ao modelo, com o conjunto de fotografias alvo (conjunto com 178 fotografias). Estes resultados têm uma condição de fiabilidade associada, isto é, apenas são apresentadas as fotografias que têm uma fiabilidade superior a 75%, excluído as restantes. Na ótica dos analistas forenses e para facilitar a leitura dos resultados, eliminamos assim algum **ruído**, ao apresentar apenas as fotografias com mais de 75% de fiabilidade, já que os analistas devem dar mais ênfase às fotografias com maior fiabilidade e menos àquelas que tenham um valor baixo na semelhança à categoria.

5.4 Resultados

Agora com o modelo já escolhido e devidamente treinado (ver secção 5.3), passamos à fase de avaliação da solução. Para isso, testámos o modelo de aprendizagem escolhido com o conjunto de fotografias alvo.

Para ser possível testar a aplicação foi necessário simular um possível caso real, incluindo a obtenção de uma imagem de disco onde será feita uma análise e recolha de fotografias. Neste caso prático, foi criada uma imagem de um drive USB de 16GB,

onde se encontram 178 fotografias escolhidas aleatoriamente. A nossa solução vai classificar estas fotografias e apresentar um resultado ao utilizador.

Após a classificação ser executada é-nos apresentado o resultado geral, identificado na Figura 5.2, de onde conseguimos inferir que 38 fotografias das 178 iniciais foram excluídas por terem uma fiabilidade inferior a 75%, sendo então consideradas para o resultado apresentado ao utilizador 140 fotografias.

Das 140 fotografias a classificação gerou o seguinte output:

- 19 fotografias foram classificadas na categoria cartão de crédito
- 45 fotografias foram classificadas na categoria criança
- 29 fotografias foram classificadas na categoria outros
- 47 fotografias foram classificadas na categoria arma

Figura 5.2: Tabela resumo apresentada via página web ao utilizador, após a classificação das fotografias ter sido feita.

Na Figura 5.3 apresentamos uma tabela mais detalhada de cada fotografia que foi analisada, permitindo fazer uma análise por fotografia. Cada linha representa uma fotografia analisada, incluindo a categoria e respetiva percentagem de precisão que lhe foi atribuída pelo programa.

Figura 5.3: Tabela detalhada dos resultados, apresentada ao utilizador.

Depois de fazer uma verificação manual das fotografias com precisão superior a 75%, ou seja, depois de analisar manualmente as 140 fotografias contabilizadas, verificou-se que existem falsos positivos, mesmo contabilizando apenas as fotografias com precisão mais elevada. Traduzindo para valores mais concretos, em 140 fotografias classificadas com precisão elevada (superior a 75%), 7 são falsos positivos, ou seja, existem 7 fotografias que não foram classificadas corretamente, sendo-lhes atribuídas outras categorias às quais não pertencem. Este número de falsos positivos representa 5% do total de fotografias consideradas.

Foram realizados vários testes onde as variáveis das camadas foram variando assim como as fotografias do conjunto alvo, de modo aleatório. Na Figura 5.4, é apresentado o gráfico relativo à média, mínimo e máximo de falsos positivos de cada categoria, e do total.

Figura 5.4: Gráfico com a média, mínimo e máximo de falsos positivos, de cada categoria e do seu total.

A categoria que obteve maior quantidade de fotografias mal classificadas, falsos positivos, foi a **outros**, com uma média de 22%. A categoria **cartão de crédito** foi a que obteve melhor precisão, uma vez que o seu máximo de falsos positivos corresponde a 6%. Os resultados correspondem ao que esperávamos, dado a categoria **outros** conter fotografias de muitos tipos diferentes, gerando portanto classificações menos precisas. Como total dos testes obtemos uma média de cerca de 10% de falsos positivos.

Na Figura 5.5, apresentamos um gráfico com a média, mínimo e máximo de verdadeiros positivos por categoria, e também do total.

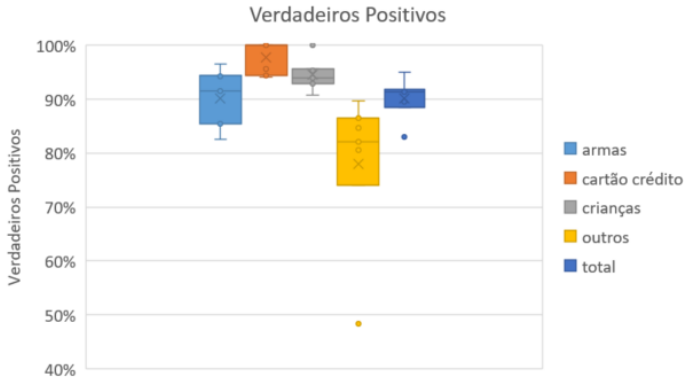

Figura 5.5: Gráfico com a média, mínimo e máximo de verdadeiros positivos, de cada categoria e do seu total.

As categorias que obtiveram maior taxa de fotografias corretamente classificadas, foram as categorias **cartões de crédito e crianças**, com uma média de aproximadamente 98% e 95% de verdadeiros positivos, respetivamente.

A categoria com menor taxa de verdadeiros positivos, é a **outros**, com uma média de 78%. O facto desta categoria ser tão abstrata e tão dispersa, incluindo fotografias sem um padrão que se possa seguir, interferiu com a precisão do modelo, uma vez que este não tem uma base para conseguir aprender um padrão ou traços minimamente semelhantes entre as fotografias da categoria "Outros". Daí ser a categoria com menor número de verdadeiros positivos e maior número de falsos positivos. Como total obtemos uma média de, aproximadamente, 90% de verdadeiros positivos.

Na Figura 5.6, podemos analisar a precisão de cada categoria. A precisão foi calculada de acordo com os valores dos dois gráficos anteriores (Figura 5.4 e 5.5), de acordo com a fórmula P = TP/(TP+FP).

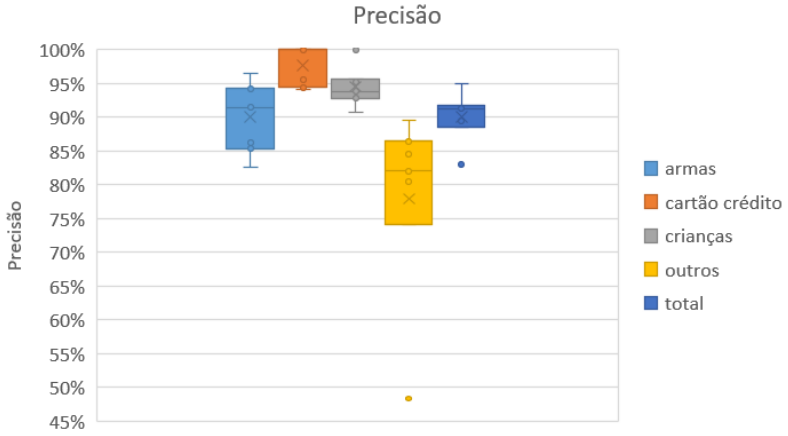

Figura 5.6: Gráfico com a precisão das categorias nos testes efetuados e o seu total.

Podemos observar que a categoria com maior precisão foi **cartão de crédito**, com precisão média de 97%. A categoria **outros** obteve a precisão mais baixa, com

uma média de 78%, pelas razões já explicadas. Como total obtivemos uma média de 90% de precisão, um resultado positivo neste contexto.

Depois da análise e discussão dos resultados obtidos, iremos descrever na próxima secção o cumprimento dos requisitos definidos no Capítulo 3.

5.5 Cumprimento dos requisitos

Todos os requisitos definidos no capítulo 3 foram cumpridos. Relativamente aos requisitos funcionais, a solução consegue receber uma imagem de disco e apresentar os resultados gerados, relativos à precisão das fotografias, numa interface gráfica, para o utilizador. Quanto aos requisitos não funcionais, a solução desenvolvida suporta imagens de disco de tipo dd, img e e01, e reconhece fotografias do tipo png, jpeg e jpg. Além disto, é executável em Linux, é baseada em ferramentas open-source, e consegue analisar uma imagem de disco de 16GB com mais de 100 fotografias e apresentar os resultados dentro de 20/25 segundos.

5.6 Sumário

Neste capítulo definimos o setup experimental identificando os datasets de treino e avaliação, e as métricas utilizadas para avaliar o modelo. Depois de vários testes para escolher o modelo, realizámos uma análise aos resultados dos testes efetuados, apresentando dados estatísticos para as métricas verdadeiros positivos, falsos positivos, e precisão. No próximo capítulo, iremos fazer uma sucinta descrição de todo o trabalho realizado, apresentando também pontos relevantes para melhorias futuras.

Capítulo 6

Conclusão e Trabalho Futuro

O objetivo deste trabalho era o desenho, implementação e avaliação de uma solução integrada, que conseguisse analisar uma imagem de disco, extrair as fotografias nela contidas, classificá-las com base em quatro categorias pré-definidas e mostrar o resultado ao utilizador numa interface amigável.

Desse modo, neste trabalho desenvolvemos uma ferramenta que integra todas as fases do processo, necessitando como entrada apenas do caminho para uma imagem de disco. A partir daí, classifica as fotografias e gera um resultado, e apresenta-o ao utilizador. Desta forma, a solução final permite facilitar a tarefa dos especialistas forenses, que anteriormente necessitavam de múltiplas ferramentas individuais, fazendo a sua integração de modo manual.

Após o desenvolvimento da solução, foram realizados inúmeros testes de forma a analisar a sua conformidade com o desejado, e a precisão nos resultados gerados. Com base nesta análise, foram implementadas algumas melhorias no modelo de treino elaborado, tendo a solução final gerado resultados com precisão aceitável para o contexto.

Um dos objetivos do trabalho era também testar a aplicação em ambiente real, mas devido à falta de resposta por parte das organizações externas, que era previsto colaborarem, este teste não foi possível realizar. Como o conjunto de dados de referência não nos foi, assim, fornecido, o modelo foi treinado com fotografias obtidas através do Google. Acreditamos que com fotografias de entidades forenses a precisão dos resultados seria maior, pois os padrões das fotografias seriam mais semelhantes. Futuramente seria interessante testar a aplicação num ambiente real para testar esta hipótese. Além disso, deveríamos aumentar o número de fotografias para treinar o modelo, pois os algoritmos de machine learning usados tendem a aumentar a precisão com o aumento da quantidade de dados.

Glossário

Imagem de disco: Ficheiro que contém o conteúdo de um disco externo

Fotografias alvo: Conjunto de fotografias contidas da imagem de disco

Fotografias de referência: Conjunto de fotografias obtidas de fonte externa, que serão base de comparação com o conjunto de fotografias alvo

Bibliografia

[1] Tech Talk. Top 20,
https://bit.ly/2BfpAej

[2] Opentext. Product Overview,
https://bit.ly/2HM6JNn

[3] Guidance Software. EnCase Forensic Product Overview,
https://bit.ly/2WwOkHw

[4] Infosec Institute. SANS Investigate Forensic Toolkit (SIFT),
https://bit.ly/2MIunsE

[5] SANS Institute. User manual,
https://media.readthedocs.org/pdf/sift/latest/sift.pdf

[6] Digital Forensics. Community: Downloads,
https://bit.ly/2Q2sAQE

[7] Crowdstrike. Crowdresponse,
https://bit.ly/2MIDgCN

[8] Volatility Foundation,
https://www.volatilityfoundation.org/

[9] Open source foru. Volatility,
https://opensourceforu.com/2016/10/volatility/

[10] Forensics focus. RAM Forensics analysis,
https://bit.ly/2WBzj7a

[11] Sleuth kit. Description Autopsy,
https://www.sleuthkit.org/autopsy/desc.php

[12] Sleuth kit. Download Autopsy,
https://www.sleuthkit.org/autopsy/download.php

[13] Phoenixts. Incident Response Tools
https://phoenixts.com/blog/forensics-tools-ftk-linux/

[14] Tecnologiaesi. Computação forense - Memória ram,
https://bit.ly/2Uw9Lqh

[15] Access Data. Download,
https://bit.ly/2RxpdQZ

[16] Longin Jan Latecki. Template Matching,
https://bit.ly/2UAecR6

[17] OpenCv. Template Matching,
https://bit.ly/2DzBcul

[18] Becoming Human. AI, Machine Learning and Deep Learning,
https://bit.ly/2NtCvgq

[19] Zendesk. Machine Learning,
https://bit.ly/2DQoeIl

[20] Microsoft. Photo DNA,
https://www.microsoft.com/en-us/photodna

[21] Wikipedia. PhotoDNA,
https://en.wikipedia.org/wiki/PhotoDNA

[22] Microsoft. PhotoDNA - Step by Step,
https://bit.ly/2G0SMsY

[23] Common Vision Blox. CVB Polimago,
https://bit.ly/2GiOhcD

[24] OpenCV. About,
https://opencv.org/about.html

[25] MathWorks. MATLAB and OpenCV,
https://www.mathworks.com/discovery/matlab-opencv.html

[26] Lear OpenCV. Install OpenCV3 on Ubuntu,
https://www.learnopencv.com/install-opencv3-on-ubuntu/

[27] Bolide Software. Image Comparer,
https://bit.ly/2DN67CP

[28] Amazon. Amazon Rekognition,
https://aws.amazon.com/rekognition/

[29] YOLO with OpenCV,
https://bit.ly/2GycV99

[30] COCO Labels,
https://bit.ly/2WMSStq

[31] API Bing Image Search - About
https://bit.ly/2IPbvKz

[32] API Bing Image Search - Download
https://bit.ly/2BVQ4kV

[33] Keras e redes neurais,
https://bit.ly/2OftpTT

[34] Keras, Develop Your First Neural Network
https://bit.ly/2BkXb6O

[35] Keras, MaxPooling
https://bit.ly/2S1jZit

[36] Keras, Flatten
https://bit.ly/2NBoZvt

[37] Neural Networks, Overfitting
https://bit.ly/2XRJl7D

[38] Keras, Dropout
https://bit.ly/2Lb52d8

[39] Quora, Keras - Dense and Dropout layers
https://bit.ly/2LJGRBZ

[40] TensorFlow. Convolutional Neural Networks - Layers
https://bit.ly/2D8byvs

[41] Towards Data Science. Softmax funtion
https://bit.ly/2KzB7Ij

[42] Machine Learning. Multi-Class Neural Networks: Softmax
https://bit.ly/2xI3zSB

[43] Kite. Label Binarizer
https://bit.ly/2S3mEIk

[44] Medium. Data splitting functions
https://bit.ly/2XZZuIg

[45] Machine Learning Mastery. Loss and Loss Functions
https://bit.ly/2G7GugQ

[46] Keras. Losses fuctions
https://keras.io/losses/

[47] Statistic How To. Loss Funtion
https://bit.ly/2CZhFSS

[48] Medium. Data splitting functions
https://bit.ly/2XZZuIg

[49] Keras. Optimizers
https://keras.io/optimizers/

[50] Towards Data Science. Metrics to Machine Learning
https://bit.ly/2x2e4D4

[51] Keras. Sequential model
https://keras.io/models/sequential/

[52] Py Image Search. Fit and fit generator
https://bit.ly/32rRiA4

[53] Keras. Preditct
https://keras.io/models/sequential/

[54] After Academy. Função de perda L2
https://afteracademy.com/blog/what-are-l1-and-l2-loss-functions

[55] Analytics Vidhya. Tecnicas de regularização
https://bit.ly/2kt8oZA

[56] Towards Data Science. Batch Normalization
https://bit.ly/2kt8oZA

[57] Machine Learning Mastery. Learning Rate
https://bit.ly/2MKil4O