Estágio

Mestrado em Engenharia Informática – Computação Móvel

Protótipo para Extração de Metadados de Vídeo

Daniel dos Santos Vieira

Leiria, agosto de 2017

Estágio

Mestrado em Engenharia Informática – Computação Móvel

Protótipo para Extração de Metadados de Vídeo

Daniel dos Santos Vieira

Estágio de Mestrado realizada sob a orientação de Vítor Manuel de Oliveira Pegado de Noronha e Távora**,** Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, agosto de 2017.

Esta página foi intencionalmente deixada em branco

Dedicatória

Aos meus pais.

Esta página foi intencionalmente deixada em branco

Agradecimentos

Com o término do presente Relatório de Estágio curricular não posso deixar de agradecer a algumas pessoas que, direta ou indiretamente, me ajudaram nesta caminhada tão importante da minha vida pessoal, académica e profissional.

Em primeiro lugar, agradeço a orientação que o Professor Vitor Távora disponibilizou durante a elaboração do presente Relatório. Obrigada pelo profissionalismo que demonstrou ter. Foi um privilégio ser seu orientando.

Agradeço ao corpo docente e não docente da Escola Superior de Tecnologia e Gestão de Leiria por terem feito com que me sentisse em "casa" na "vossa casa".

Um muito obrigado aos amigos que fiz e trouxe comigo. Agradeço à Ana, ao Nélson e ao Diogo por suportarem o meu mau feitio.

Por fim, mas não menos importante, agradeço à minha família pois sem ela dificilmente conseguiria chegar até aqui. Por estarem sempre lá para me amparar, para me criticar, e para me felicitar, fazendo-me sentir uma pessoa melhor. Agradeço à minha mãe por ter um gostinho especial de me colocar sempre à prova, e ao meu pai por esperar mais aquele bocadinho no carro enquanto eu terminava qualquer coisa na sala de estudo.

Agradeço também à Suri, pois sem ela esta caminhada teria sido bem mais difícil.

Esta página foi intencionalmente deixada em branco

Resumo

Face ao grande crescimento que tem ocorrido com a disponibilização e a visualização de vídeos *online*, torna-se necessário melhorar o resultado dos milhões de pesquisas diariamente efetuadas. Para tal terão de se criar ferramentas que auxiliem no processo de identificação automática dos conteúdos dos vídeos. Atualmente para efetuar essa identificação terá de se consultar a informação introduzida manualmente (título e/ou palavras-chave) ou visualizar o seu conteúdo.

No âmbito do estágio de natureza profissional foi desenvolvido um protótipo para o reconhecimento de objetos em vídeos, e para a extração automática dos metadados correspondentes à classe de objetos detetados. O reconhecimento de objetos é efetuado em cada uma das *frames* (imagens) que compõem o vídeo, e inclui a deteção e a classificação dos objetos.

Na primeira versão do protótipo são utilizados classificadores *Haar Cascade*, e a API OpenCV, para efetuar o reconhecimento e classificação de objetos em cada imagem. Foram detetados alguns problemas com as taxas de acerto no reconhecimento dos objetos, com exceção das faces humanas, e com o desempenho do protótipo. Os problemas foram ultrapassados com uma nova versão do protótipo, que utiliza a rede neuronal convolucional GoogLeNet, suportada pela API Tensorflow e treinada com o *dataset ImageNet*, para efetuar o reconhecimento e a classificação dos objetos em imagens.

Palavras-chave: Reconhecimento de objetos, Vídeo, Classificadores *Haar Cascade*, OpenCV, Redes neuronais convolucionais, Tensorflow

Esta página foi intencionalmente deixada em branco

Abstract

The increasing availability of videos online and video visualization makes it necessary to improve the results of the millions of daily searches. For that we need to create tools to assist the process of automatic identification of the video content. Nowadays to make that identification we have to access the information that was manually inserted (title and/or key words) or visualize its content.

As part of the professional internship a prototype was developed for video object recognition and for automatic extraction of metadata related with the class of the identified objects. The object recognition is performed in every frame of the video and it includes object detection and classification.

In the first version of the prototype, Haar cascade and API OpenCV are used to perform the object recognition and classification in each frame. Some problems were detected with the object recognition accuracy rates, apart from the human faces and the prototype performance. These problems were overcome with a new version of the prototype which uses the GoogLeNet convolutional neural network supported by API Tensorflow and trained with the dataset ImageNet to perform object recognition and classification in images.

Keywords: Object Recognition, Video, Haar Cascade classifiers, OpenCV, Convolutional neural network, Tensorflow

Esta página foi intencionalmente deixada em branco

Lista de siglas

- API - *Application Programming Interface*
- CNN - *Convolutional Neural Networks*
- GPS - *Global Positioning System*
- GPU - *Graphics Processing Unit*
- IVR - *Interactive Voice Response*
- JavaCV - *Java interface to OpenCV*
- Java EE - *Java Enterprise Edition*
- JVM - *Java Virtual Machine*
- MMS - *Multimedia Messaging Service*
- OpenCV - *Open Source Computer Vision Library*
- PDAs - *Personal Digital Assistants*
- RCS - *Rich Communication Services*
- REST - *Representational State Transfer*
- RGB - *Red*, *Green and Blue*
- SMS - Short Message Service
- URL - *Uniform Resource Locator*
- VoIP - *Voice over Internet Protocol*
- WAP - *Wireless Application Protocol*

Esta página foi intencionalmente deixada em branco

Índice

1. Introdução

Durante os próximos anos é expectável que continue a aumentar o tráfego na internet, e, de acordo com o *Cisco Visual Networking Index* [1], os conteúdos de vídeo serão um dos que mais contribuirão para esse aumento de tráfego. Na realidade o vídeo tem vindo a ser cada vez mais utilizado na internet tanto por empresas, para efetuar negócios, como por particulares, para comunicar e para aceder a conteúdos disponibilizados por várias entidades.

Um relatório elaborado pela empresa ZenithOptimedia [2] previu, para o ano 2016, um crescimento de 19,8% para o tempo médio que os consumidores dedicam à visualização de conteúdos vídeo, utilizando para tal sobretudo plataformas móveis. Para esse ano foi também previsto um crescimento de 34,8% para o consumo de vídeo a partir de plataformas móveis, e um crescimento de 6,5% para o consumo de vídeo a partir de computadores.

De acordo com esse relatório, a maior vítima do crescimento do vídeo será a televisão tradicional, que terá iniciado em 2016 o decrescimento de audiências. Esta quebra da audiência televisiva já se fazia sentir há algum tempo em diversos países, como a França, a Rússia, o Reino Unido e os Estados Unidos. Esse fenómeno encontra-se diretamente relacionado com o aumento dos conteúdos digitais, tanto em plataformas de formato curto, designadamente o *YouTube* ou o *Vimeo*, como em plataformas de formato longo, como o *Netflix*. Previa-se também um aumento no número de consumidores de vídeo *online* em 12 mercados, entre os quais se incluem o francês (50%), o alemão (27,5%), o chinês (27,2%) e o norte americano (12,3%).

Como resultado da redução da audiência televisiva, e do aumento do consumo de vídeos *online*, estimava-se que o investimento publicitário neste formato crescesse 12,8% em 2017, o correspondente a um oitavo do total do investimento digital mundial. Os Estados Unidos e a Itália lideravam o *ranking* de mercados com maior proporção de gastos com publicidade em vídeo *online*.

Associado ao crescimento do mercado móvel e à existência de ferramentas que facilitam a partilha de conteúdos, tais como o *Facebook* ou o *Instagram*, torna-se necessário disponibilizar informação para melhorar o resultado dos milhões de pesquisas diariamente efetuadas aos conteúdos dos vídeos *online*.

Tradicionalmente os conteúdos vídeo contêm a informação desorganizada, desindexada e não pesquisável. Não é possível identificar o conteúdo de um vídeo sem consultar informação introduzida manualmente (título e/ou palavras-chave), ou sem visualizar o seu conteúdo.

Para tentar ultrapassar esse problema, no âmbito do estágio de natureza profissional proposto pela entidade de acolhimento, foi projetado e desenvolvido um sistema de criação automática de metadados (*tags*) a partir da informação extraída de um vídeo. Esta informação é obtida através do reconhecimento de objetos nas *frames* (imagens) que constituem o vídeo, e que abrange a deteção e a classificação dos objetos.

Foi também desenvolvido um mecanismo para identificar, e ultrapassar automaticamente, os espaços publicitários incluídos nos vídeos, por forma a evitar que os utilizadores tenham de o efetuar manualmente.

1.1. Entidade de Acolhimento

O estágio decorreu durante nove meses na filial de Leiria da empresa WIT Software, que desenvolve soluções avançadas e produtos de para o setor de telecomunicações móveis e que se encontra atualmente estabelecida em Coimbra (sede), Lisboa, Porto e Leiria, e com filiais em Reading (Reino Unido) e San José (Califórnia, EUA).

As principais áreas de mercado da WIT são o desenvolvimento de aplicações para telemóveis e PDAs, o desenvolvimento de plataformas para suporte a serviços móveis (SMS, MMS, WAP), o desenvolvimento de plataformas de middleware para Operadores Móveis, a integração de aplicações empresariais com soluções de mobilidade, a consultoria no desenho e desenvolvimento de serviços móveis, o desenvolvimento de soluções baseadas em localização (GPS), o desenvolvimento de soluções VoIP (Voz sobre IP, do inglês *Voice over Internet Protocol*), o desenvolvimento de aplicações de videochamada com integração em redes móveis, e o desenvolvimento de soluções RCS (*Rich Communication Services*).

A WIT Software tem atualmente diversos clientes nacionais e internacionais abrangendo o sector das telecomunicações móveis, banca, media e transportes. Desde a sua criação que mantém uma parceria tecnológica com a Vodafone Portugal para a qual desenvolve vários serviços móveis nas áreas de mensagens móveis e de soluções à medida.

A nível nacional tem como principais clientes a SIC, Alcatel, Caixa Geral de Depósitos e BANIF, estes últimos em parceria com a empresa EF-Tecnologias. No mercado europeu tem como parceiro a empresa Vodafone Global. No mercado americano possui uma parceria estratégica com a empresa canadiana Airborne Entertainment que tem possibilitado à o desenvolvimento de serviços móveis para os vários operadores Norte-Americanos, incluindo a Verizon Wireless, Cingular, AT&T Wireless, T-Mobile, Sprint, RogersAT&T, Microcell, Bell-Mobility, Sasktel, Telus Mobility, Vodafone US.

A WIT Software possui vários prémios e certificações, podendo destacar-se que, em 2012, ganhou um concurso internacional para o desenvolvimento de clientes RCS de referência para Android e iOS.

1.2. Motivação e Objetivos do Estágio

Face ao grande crescimento que tem ocorrido com a disponibilização e a visualização de vídeos *online*, torna-se necessário criar ferramentas que auxiliem no processo de identificação automática dos seus conteúdos. Atualmente para efetuar essa identificação torna-se necessário consultar o título ou as palavras-chave manualmente introduzidas, ou, em alternativa, visualizar o conteúdo do vídeo.

Para auxiliar na resolução desse problema, a entidade de acolhimento propôs um estágio que apresentava como objetivo principal o desenvolvimento de um protótipo para a extração automática de metadados de vídeo. Como caso de uso pretendia-se também desenvolver um mecanismo para identificar os espaços publicitários incluídos nos vídeos.

De forma a poder alcançar esse objetivo pretendia-se identificar e avaliar as ferramentas existentes para a análise de conteúdos de vídeo e de áudio, e que apresentem soluções para o reconhecimento de fala e para o reconhecimento de objetos. Posteriormente, pretendia-se o desenvolvimento de um protótipo que demonstrasse a viabilidade da utilização dessas ferramentas em casos de uso particulares, tais como a deteção de objetos específicos num vídeo, a categorização da informação, e a extração de fala e de texto presentes num vídeo.

Cerca de um mês após ter-se iniciado o estágio, realizou-se uma reunião com o intuito de analisar o âmbito do estágio. Considerando a amplitude das duas principais áreas de

intervenção, o reconhecimento de fala e o reconhecimento de objetos, optou-se por focar apenas no reconhecimento de objetos em imagens e vídeo.

1.3. Estrutura do documento

O presente documento apresenta a seguinte estrutura nos próximos capítulos:

- No capítulo dois é efetuado o levantamento do estado da arte nas principais áreas de intervenção do estágio: reconhecimento de fala e reconhecimento de objetos. Neste âmbito foram analisadas diversas soluções existentes.

- No capítulo três é apresentada a arquitetura aplicacional do protótipo para a criação automática de metadados (tags) a partir do reconhecimento de objetos efetuado em vídeos.

- No capítulo quatro é efetuada a descrição e avaliação da primeira versão do protótipo desenvolvido para a deteção e a classificação de objetos em vídeos, e que utiliza classificadores *Haar Cascade* [3]. Encontra-se também descrita a solução desenvolvida para a deteção de intervalos publicitários.

- No capítulo cinco encontra-se descrita a segunda versão do protótipo desenvolvido para a deteção e classificação de objetos em vídeos, e que utiliza redes neuronais convolucionais. São também apresentados os resultados obtidos com a nova versão do protótipo.

- Finalmente, no capítulo seis são apresentadas as principais conclusões obtidas com a realização do estágio.

Durante a elaboração do documento existiu a preocupação de assegurar a sua acessibilidade, pelo que se optou por não incluir figuras nem tabelas.

2. Estado da Arte

No presente capítulo é efetuado o levantamento do estado da arte nas que se previam ser as principais áreas de intervenção do estágio: reconhecimento de fala e reconhecimento de objetos. Para além do respetivo enquadramento, são também apresentadas diversas soluções existentes.

Atendendo a que os sistemas de reconhecimento de fala acabaram por não ser desenvolvidos no estágio, será sobretudo descrita a sua evolução ao longo do tempo.

Na sequência da análise efetuada às várias soluções para o reconhecimento (deteção e classificação) de objetos em ficheiros de vídeo, foi selecionada a API (*Application Programming Interface*) OpenCV [4] para o desenvolvimento da primeira versão do protótipo.

2.1. Reconhecimento de Fala

O reconhecimento de fala (*speech recognition*) e o reconhecimento de voz (*voice recognition*) referem-se à capacidade das máquinas em executar ações solicitadas através de comandos de voz. Poderão ser fornecidas instruções para controlar dispositivos móveis, bem como para efetuar a transcrição automática de textos ditados ou de conteúdos de áudio (*speech to text*).

Apesar de por vezes serem indiferenciadamente utilizados, o reconhecimento de fala e o reconhecimento de voz são tecnologias diferentes e possuem objetivos muito distintos [5]. O reconhecimento de fala é o processo de conversão da voz humana em informação digital, focando-se na deteção e na identificação de palavras ou frases. O reconhecimento de voz tem como objetivo principal a identificação da pessoa que fala.

Em 1932 já existia investigação na área do reconhecimento da fala, sobretudo em universidades e centros de investigação [6][7][8]. Em 1976 Raj Reddy [9] efetuou uma revisão abrangente do estado da arte no reconhecimento da fala, numa época em que a capacidade computacional disponível condicionava a evolução dessa tecnologia. Nessa altura, utilizando computadores com 4MB de memória, eram necessários cerca de 100 minutos para processar 30 segundos de áudio. O autor do artigo previu que na década

seguinte seria possível construir um sistema de reconhecimento de fala por cerca de $20.000. Embora tenha demorado mais tempo do que o previsto, os objetivos foram alcançados com custos muito inferiores aos previstos por Raj Reddy. Atualmente muitos dispositivos móveis já possuem sistemas de reconhecimento de fala.

O ano de 1990 é considerado como o da grande mudança no panorama de reconhecimento de fala para texto (*speech to text*), tendo aparecido as primeiras ofertas comerciais nessa área. A evolução dos sistemas ocorreu sobretudo devido ao aumento da capacidade computacional que permitiu o processamento de diálogos, em vez de palavras soltas, e o controlo de ruídos externos. A utilização da inteligência artificial também contribuiu bastante para a evolução dos sistemas.

Em 1995 a Microsoft Speech API (SAPI) foi adicionada ao Windows com o intuito de se poder utilizar o reconhecimento de voz em aplicações *desktop*. Quatro anos mais tarde, surge o primeiro sistema IVR (*Interactive Voice Response*), denominado VoiceXML [10], que tem como objetivo a interação com clientes em chamadas telefónicas.

Em 2000 a empresa Dragon Systems foi considerada a líder do mercado. Sendo atualmente conhecida por Nuance, foi esta empresa que forneceu a tecnologia para o desenvolvimento do sistema de reconhecimento da Siri que é utilizada nos equipamentos Apple [11].

Em 2001 a Microsoft apresentou o MiPad [12], um protótipo capaz de efetuar o reconhecimento de fala em diversos contextos. Atualmente o Google Now [13], a Siri [11] e o Cortana [14] utilizam soluções semelhantes.

Em 2007 a Google desenvolveu o seu primeiro produto de reconhecimento de fala denominado GOOG-411 (ou *Google Voice Local Search*) [15]. Utilizando a grande quantidade de informações áudio que tem a capacidade para obter diariamente, atualmente a Google é uma das empresas de topo na área, suportando o reconhecimento de fala em mais de 30 idiomas diferentes.

Em 2010 a Universidade de Toronto, a Microsoft, a Google e a IBM criaram um consórcio que desenvolveu um sistema de reconhecimento de fala utilizando redes neuronais baseadas em *feedforward* [16] [17]. Este tipo de sistemas é atualmente utilizado, e consegue processar uma grande quantidade de informação, proveniente de todo o mundo, e falada em diferentes idiomas.

A aquisição de informação foi-se tornando um processo simples, embora o seu tratamento ainda se traduzisse num preço bastante elevado. A eficácia das redes neuronais [18] mudou drasticamente o modo de obtenção de resultados, traduzindo-se em melhor qualidade e fiabilidade, ainda que seja um processo demasiado pesado para um simples computador. Não sendo possível processar toda a informação em memória, encontram-se a ser aplicadas diferentes técnicas para auxiliar no processamento da informação, nomeadamente com a utilização de GPUs (*Graphics Processing Units*, ou Unidades de Processamento Gráfico) [19] e como recurso à computação distribuída [20].

O avanço do processo de reconhecimento de fala continuará a estar dependente do aumento da capacidade computacional, pois, mesmo com a utilização de sistemas distribuídos, são necessários vários meses para treinar um modelo acústico.

O treino do modelo acústico perfeito ainda estará longe de ser alcançado, pois existem muitas variáveis a ter em conta, tais como como dialetos, pronuncias e significados. Também será necessário lidar com a utilização de palavras estrangeiras na conversação efetuada num determinado idioma.

É expectável que a médio prazo possa ser possível estabelecer-se um diálogo entre seres humanos e máquinas, sendo para tal necessário aguardar-se pela evolução dos sistemas de reconhecimento de fala a integrar nas máquinas.

No Anexo 1 é apresentado o levantamento efetuado de várias soluções que efetuam o reconhecimento de fala.

2.2. Reconhecimento de objetos

O reconhecimento de objetos é um dos maiores desafios na área da visão computacional. O objetivo principal desta área é o de automatizar as tarefas realizadas pelo sistema visual humano, por forma a poder compreender imagens digitais e vídeos. Para tal, foram desenvolvidos métodos para a aquisição, o processamento, a análise e a extração de informação de imagens digitais e de vídeos.

No âmbito do reconhecimento de objetos numa imagem digital ou num vídeo, que é composto por uma sequência de imagens digitais, torna-se necessário utilizar um conjunto

de algoritmos para realizar as tarefas (*tasks*) de classificação, localização e deteção dos objetos.

A tarefa de classificação de um objeto visa identificar e categorizar o objeto de acordo como seu tipo. Por exemplo, pode-se identificar uma pessoa, um determinado tipo de animal (por exemplo o cão), um determinado tipo de fruta (por exemplo a maçã).

A tarefa de localização consiste na utilização de uma figura geométrica, normalmente um retângulo, para delimitar um objeto classificado. Esta tarefa apenas poderá ser utilizada num único objeto.

Para se efetuar a identificação de vários objetos classificados numa determinada imagem poderá utilizar-se a tarefa de deteção de objetos ou a tarefa de segmentação de objetos *(Instance segmentation)***.** A tarefa de deteção de objetos delimita cada objeto classificado com uma figura geométrica, enquanto que a tarefa de segmentação contorna cada um dos objetos classificados.

Caso se pretenda associar um conjunto de palavras (legendas) a uma ou várias imagens poderá também utilizar-se a tarefa de legendagem de imagens (*Image Captioning*). Por exemplo, a partir de uma, ou de várias, imagens referentes a um jogo de futebol, em que são detetados objetos referentes a vários objetos classificados, poderá associar-se a essa(s) imagem(ens) a legenda "jogo de futebol entre as equipas X e Y".

O foco principal do presente estágio é o da classificação de objetos, tendo sido efetuada uma pesquisa de aplicações, de serviços e de APIs (*Application Programming Interfaces*) existentes para a realização dessa tarefa. Verificou-se que existem soluções que produzem bons resultados na execução dessa tarefa, e que geram receitas para os seus proprietários.

No Anexo 2 encontram-se descritas as várias soluções analisadas para o reconhecimento de objetos. Optou-se por efetuar uma pesquisa ao nível do *software* existente, pelo que não foram incluídas as soluções que se encontram a ser incorporadas no desenvolvimento de carros autónomos.

A única aplicação analisada foi o CamFind, que se trata de uma aplicação móvel gratuita para o reconhecimento de objetos. Os utilizadores podem carregar na aplicação imagens ou fotografias, obtendo a informação referente ao tipo de objetos identificados.

Recentemente esta aplicação também passou a ser disponibilizada como um serviço de *cloud*.

A maioria das soluções analisadas encontra-se disponibilizada como serviço, sendo a realidade aumentada a funcionalidade maioritariamente disponibilizada pelas soluções. Para o seu funcionamento torna-se necessário efetuar o reconhecimento de objetos. O princípio de funcionamento dos serviços é semelhante, existindo um cliente que captura e envia as imagens (ou vídeo), sendo o serviço responsável pela identificação e classificação dos objetos, assim como pelo envio da resposta ao cliente. Alguns dos serviços apenas suportam determinadas categorias de classificação, por exemplo na área da moda ou dos livros.

Alguns dos serviços são sustentados pela comecialização de pacotes com um número limitado de reconhecimentos, mensais ou anuais, o que os torna muito pouco atrativos para utilizar com conteúdos de vídeo (normalmente compostos por milhares de imagens).

A utilização dos serviços disponibilizados para o reconhecimento de objetos apresenta como principal inconveniente a dependência de entidades externas, que poderão alterar as condições em que fornecem o serviço, ou mesmo, descontinuá-lo. Esta situação ocorreu até à data com os seguintes serviços analisados: Vufind Recognize, 6px.io, API Moodstocks e Orbeus ReKognition.

Foi também analisada uma API multiplataforma de código aberto (*open source,* em inglês), designada por OpenCV (*Open Source Computer Vision Library*) [4]. Essa biblioteca, originalmente desenvolvida pela Intel em 2000, inclui módulos para o processamento em tempo real de imagens e de vídeo. Encontram-se implementados centenas de algoritmos na área da visão computacional, incluindo o suporte para a deteção e a classificação de objetos. Existe uma grande comunidade de utilizadores do OpenCV, incluindo grupos de investigação, empresas e instituições governamentais.

Considerando as caraterísticas do OpenCV, e as limitações da aplicação e dos serviços analisados, incluindo a grande dependência de outras entidades, optou-se por utilizar essa biblioteca no desenvolvimento do protótipo para extração de metadados de vídeo.

Esta página foi intencionalmente deixada em branco

3. Arquitetura Aplicacional

Na sequência da análise efetuada às várias soluções que efetuam o reconhecimento de objetos, optou-se por uma arquitetura cliente/servidor para o protótipo de extração de metadados de vídeo. O cliente será o responsável pela obtenção dos conteúdos multimédia (vídeos ou imagens) e por apresentar ao utilizador a informação (metadados) referente aos objetos detetados. O servidor processará os conteúdos multimédia recebidos, efetuando o reconhecimento dos objetos e criando os correspondentes metadados, e remeterá esta informação ao cliente.

O cliente será constituído por um sistema de captura de vídeo em tempo integral, e por um sistema de captura de vídeo em tempo real.

O sistema de captura de vídeo em tempo integral obterá os ficheiros multimédia (vídeos ou imagens) carregados pelo utilizador, enquanto o sistema de captura de vídeo em tempo real poderá obter vídeos captados, em tempo real, a partir de uma câmara de vídeo digital.

O servidor possuirá um módulo de processamento dos conteúdos multimédia (vídeos ou imagens) e um motor de reconhecimento. O módulo de processamento encarregar-se-á de obter os conteúdos multimédia captados pelo cliente, decompondo os vídeos nas *frames* (imagens) que o compõem, e remete-as sequencialmente para motor de reconhecimento.

O motor de reconhecimento receberá as imagens remetidas pelo módulo de processamento, efetuará a deteção de objetos, criará os correspondentes metadados, e enviará esta informação ao módulo de processamento.

O motor de reconhecimento utilizará o conhecimento previamente adquirido para efetuar a deteção e a classificação dos objetos. Os metadados serão associados ao tipo de informação correspondente à classe do objeto detetado. Assim, caso seja detetado um objeto da classe X (por exemplo carro), a informação contida nos metadados será X ("carro").

Esta página foi intencionalmente deixada em branco

4. Reconhecimento de objetos com classificadores *Haar Cascade*

No presente capítulo é descrita a versão inicial do protótipo desenvolvido para a deteção e classificação de objetos em conteúdos de vídeo, e que utiliza classificadores *Haar Cascade*. O protótipo, designado por *WitSoftware Recognition Tool v1.0*, possui a arquitetura descrita no capítulo anterior, e utiliza a API OpenCV que foi selecionada na sequência do levantamento do estado da arte. É também apresentada a avaliação desse protótipo que, apesar de se encontrar funcional, não apresentou os resultados e o desempenho desejados.

Inicialmente é efetuada uma introdução aos classificadores *Haar Cascade*, e descrita a forma como poderão ser criados e testados utilizando duas ferramentas desenvolvidas (*Haar Cascade creator tool* e *Haar Cascade performance tool*) para automatizar esses processos.

Encontra-se também descrita a solução desenvolvida para a deteção automática de intervalos publicitários em vídeos, e que também utiliza a API OpenCV.

4.1. Classificadores *Haar Cascade*

A deteção e classificação de objetos utilizando classificadores *Haar Cascade* foi proposta em [3], e consiste num processo de aprendizagem automática em que um classificador é treinado a partir de um conjunto de imagens (*dataset*) contendo o tipo de objeto a reconhecer (imagens positivas). Esse classificador é posteriormente utilizado na deteção desse tipo de objetos em outras imagens.

De um modo geral, um classificador *Haar Cascade* é treinado através da extração das características *Haar* do objeto, que são identificadas nas imagens utilizadas no seu treino. Por exemplo, é calculada a intensidade dos pixéis em cada zona da imagem. Durante esse processo, numa janela de deteção, são consideradas as regiões (retangulares) vizinhas de um local específico, sendo somadas as intensidades dos pixéis (valor do sistema RGB – *Red*, *Green* e *Blue*) pertencentes a cada região, e calculadas as diferenças entre essas somas. Esta diferença é utilizada para categorizar as várias subseções de uma imagem.

Em [3] são apresentadas duas características utilizadas na deteção de faces humanas. A primeira característica indica que a zona dos olhos é mais escura do que a zona das bochechas (e do nariz), pelo que se poderá obter uma característica *Haar* formada por duas zonas retangulares vizinhas contendo, respetivamente, os olhos (mais escura) e as bochechas (mais clara). A segunda característica indica que a zona a zona dos olhos é mais escura do que a zona da cana do nariz, pelo que se poderá obter uma característica *Haar* formada por três zonas retangulares vizinhas contendo, respetivamente, os olhos (duas zonas mais escuras nas extremidades) e a cana do nariz (zona mais clara no interior). Estas duas características *Haar* poderão ser identificadas e posteriormente utilizadas pelos classificadores na deteção de faces humanas.

O processo de treino de um classificador *Haar Cascade* é constituído por um conjunto de iterações realizadas sobre os milhares de imagens positivas (contendo o tipo de objeto a reconhecer) utilizadas no *dataset* de treino, por forma a identificar as características *Haar* do objeto a reconhecer. O processo de treino é repetido até ocorrer umas das seguintes situações:

a) alcançada a taxa de precisão, ou de erro, previamente configurada;

b) obtida a quantidade suficiente de características *Haar* identificativas do objeto a reconhecer;

c) atingida a última iteração (definida pelo utilizador).

As características *Haar* identificadas serão utilizadas em todas as imagens de treino. Para cada característica é calculado o valor a partir do qual o objeto a reconhecer é classificado como positivo ou como negativo. Durante este processo ocorrerão erros de classificação. Serão selecionadas as características *Haar* com menor taxa de erro, pois são as características que melhor classificam as imagens contendo o objeto a reconhecer.

As características *Haar* identificadas durante o processo de treino são agrupadas em vários classificadores. Para aumentar a eficiência do processo de deteção de objetos, evitando que se tenha de aplicar todos os classificadores a cada zona da imagem, foi criada uma cascata de classificadores *Haar* (*Haar Cascade*). Assim, os classificadores serão aplicados por etapas, um a um, em que o classificador da etapa seguinte apenas será aplicado caso se tenha obtido um resultado positivo com o classificador da etapa corrente.

Durante o processo de deteção de objetos uma janela é movida ao longo da imagem. A zona abrangida pela posição corrente da janela é avaliada por um classificador como positiva, se encontrar o objeto, ou como negativa, caso não o encontre. Se o resultado for negativo é interrompido o processo de deteção nessa zona, e a janela é movida para a posição seguinte da imagem. Se o resultado for positivo, essa zona será avaliada pelo classificador que se encontra na etapa seguinte da cascata.

O classificador *Haar Cascade* considera que o objeto foi detetado numa determinada zona da imagem quando todos os classificadores avaliarem positivamente essa zona da imagem.

Concluído o processo de deteção de objetos numa imagem, poderão ocorrer os seguintes casos:

- Verdadeiro Positivo: ocorre quando o objeto procurado se encontra na imagem, os classificadores detetam, e consideram a imagem como positiva.

- Falso Positivo: ocorre quando o objeto procurado não se encontra na imagem, os classificadores (incorretamente) detetam, e consideram a imagem como positiva.

- Falso Negativo: ocorre quando o objeto procurado se encontra na imagem, e os classificadores não o conseguem detetar, e consideram a imagem como negativa.

- Verdadeiro Negativo: ocorre quando o classificador deteta (incorretamente) um objeto que julga ser o objeto procurado, e marca a imagem como positiva.

Por forma a que o processo de deteção corra bem, os classificadores integrados no classificador *Haar Cascade* devem ter uma baixa taxa de casos falsos negativos. Se um classificador incorretamente considerar que um objeto não existe, o processo de deteção é interrompido, não sendo possível corrigir o erro cometido. No entanto, um classificador pode ter uma alta taxa de casos falsos positivos pois esse erro poderá ser corrigido pelos classificadores que se encontrem nas etapas seguintes [21].

Para se obter um classificador *Haar Cascade* eficaz e eficiente, o classificador que se encontra em cada etapa da cascata deverá reduzir a taxa de casos falsos positivos e diminuir

a taxa de deteção de objetos. Deverão ser fixadas metas para a alcançar para cada um desses parâmetros.

4.2. Criação de Classificadores *Haar Cascade*

O processo de criação e de treino dos classificadores *Haar Cascade* implica a realização das seguintes tarefas [22]:

- criação do *dataset* de imagens positivas e do *dataset* de imagens negativas para auxiliar no treino dos classificadores.
- treino dos classificadores
- execução dos classificadores

Para auxiliar neste processo a API OpenCV disponibiliza as ferramentas *opencv_createsamples* [23] *e opencv_haartraining* [24], podendo ambas ser executadas através da linha de comandos.

A ferramenta *opencv_createsamples* auxilia na preparação do *dataset* de imagens positivas, enquanto que a ferramenta *opencv_haartraining* é a responsável pelo treino efetuado sobre esse *dataset*, dando origem a um classificador do tipo *Haar Cascade*. Esta segunda ferramenta também poderá indicar o desempenho do classificador.

4.2.1. Definição de *datasets*

Para além do *dataset* de imagens positivas, no processo de treino dos classificadores *Haar Cascade* também poderá ser utilizado um *dataset* de imagens negativas. Este é constituído por um conjunto de imagens aleatórias que não contenham o objeto a reconhecer.

O *dataset* de imagens negativas apenas poderá conter imagens com um tamanho superior ao da imagem contendo o objeto a reconhecer, pois as imagens negativas poderão ser utilizadas no processo de criação de novas imagens positivas. O *dataset* de imagens negativas encontra-se identificado num ficheiro, designado *bg* ou *background*, que contém a localização de cada uma das imagens.

O *dataset* de imagens positivas poderá ser gerado a partir de uma imagem, ou a partir de um conjunto de imagens, previamente marcada(s) através de um processo de *sampling*, ou através de um processo supervisionado.

O processo de *sampling* permite criar um conjunto de novas imagens tendo por base uma determinada imagem. Essas imagens são criadas com a ferramenta *opencv_createsamples*, que utiliza o método de transformação de um para vários. Este método sobrepõe uma imagem contendo o objeto a detetar sobre um conjunto de *N* imagens negativas. Assim, definindo várias variações nos eixos cartesianos, é possível gerar um conjunto de *N* novas imagens positivas.

O processo supervisionado para a criação de um ficheiro contendo o *dataset* de imagens positivas requere a intervenção do utilizador na marcação da localização de cada objeto. Para facilitar esta tarefa foi desenvolvida uma ferramenta (em *Java*) durante o estágio.

Após obter-se um conjunto de imagens (positivas) contendo um determinado objeto, será armazenado no ficheiro de *dataset* a localização de cada imagem juntamente com a quantidade, e a localização exata de todos os objetos identificados. Note-se que poderão existir imagens que contenham vários objetos da mesma classe, por exemplo no caso do objeto ser da classe carro, e na imagem aparecerem vários carros.

A informação armazenada no *dataset* é gerada com a utilização da ferramenta desenvolvida, permitindo ao utilizador marcar a localização dos vários objetos existentes na imagem. Após ter sido criado um retângulo em torno de cada objeto, é atualizada a informação armazenada no *dataset*. O formato do ficheiro gerado pela ferramenta é igual ao gerado no processo de *sampling* pela ferramenta *opencv_createsamples*.

O *dataset* de treino do classificador do tipo *Haar Cascade* encontra-se num ficheiro vetor, ou .vec, que armazena a informação de todas as imagens previamente geradas, através de um processo de *sampling* ou através de um processo supervisionado. O conteúdo do vetor de imagens positivas pode ser visualizado com a ferramenta *opencv_createsamples*.

4.2.2. Automatização da criação de classificadores *Haar Cascade*

O processo de criação e teste de classificadores *Haar Cascade* requere uma grande intervenção do utilizador, pelo que, para automatizar e agilizar o processo, durante o estágio foram desenvolvidas (em *Java*) as ferramentas *Haar Cascade creator tool* e *Haar Cascade performance tool*.

A ferramenta *Haar Cascade creator tool* automatiza os processos de criação de novas imagens (processo de *sampling*), de criação do ficheiro vetor, e de criação de um ficheiro (.bat) contendo os comandos necessários para dar início e/ou continuação do processo de treino. Para controlar o sucesso da execução de cada uma dessas tarefas, a ferramenta gera um ficheiro de *log* contendo os parâmetros utilizados durante o processo de *sampling*, e a localização dos ficheiros .vec e .bat.

A ferramenta *Haar Cascade performance tool* automatiza o processo de teste dos classificadores *Haar Cascade* anteriormente criados, avaliando o seu desempenho. Neste processo são utilizadas 10 imagens positivas e 10 imagens negativas, aleatoriamente selecionadas e que não tenham sido previamente utilizadas nas fases de criação ou de treino dos classificadores.

Após a finalização dos testes, a ferramenta gera um ficheiro de texto com o relatório de desempenho contendo a informação de cada classificador, a respetiva taxa de acerto, e as imagens (positivas e negativas) selecionadas para o teste. Deste modo, será possível identificar e avaliar o desempenho dos classificadores utilizados em cada teste.

4.3. *WitSoftware Recognition Tool v1.0*

Foi desenvolvida uma aplicação *web*, denominada *WitSoftware Recognition Tool*, com o intuito de efetuar a deteção e a classificação dos objetos existentes em vídeos e em imagens fornecidos pelo utilizador.

A aplicação possui a arquitetura descrita no capítulo anterior, sendo constituída por um cliente *web* e por um servidor *web*.

O cliente *web*, denominado *WitSoftware Recognition Client*, disponibiliza ao utilizador as ferramentas para a seleção e o carregamento de ficheiros (vídeo ou imagem), e para o acesso ao vídeo capturado pela câmara digital. O cliente recebe os metadados gerados no processo de reconhecimento dos objetos existentes no conteúdo multimédia fornecido pelo utilizador.

O servidor *web*, denominado *WitSoftware Recognition Server*, processa os conteúdos multimédia capturados, e efetua a deteção e classificação de objetos recorrendo a classificadores *Haar Cascade*. Este módulo também assegura a comunicação com o cliente *web*, através da disponibilização de serviços REST (*Representational State Transfer*) e *Socket*. Foi criada uma biblioteca, denominada *WitSoftware Recognition Lib*, onde foram implementados os seguintes componentes do servidor: módulo de processamento e o motor de reconhecimento.

O utilizador poderá aceder ao cliente *web* com um navegador, e efetuar o carregamento de um ficheiro contendo uma imagem ou um vídeo integral. Em alternativa poderá ir carregando um vídeo obtido em tempo real com uma camara digital. Após ter-se iniciado o carregamento do vídeo ou da imagem, o servidor *web* encarrega-se de estabelecer a comunicação com o cliente *web*.

O módulo de processamento do servidor *web* recebe os conteúdos multimédia carregados no cliente *web*, efetua a decomposição dos vídeos nas respetivas *frames* (imagens), e remete-as sequencialmente para o motor de reconhecimento.

O motor de reconhecimento, utilizando os classificadores *Haar Cascade* previamente treinados, efetua a deteção e a classificação de objetos em cada imagem recebida. Caso os classificadores detetem algum objeto, a sua classe é adicionada a uma lista de resultados. O processo é repetido para todas as *frames* pertencentes a um vídeo. Os resultados (metadados) são enviados para o módulo de processamento, que os remete para o cliente *web* através de um serviço REST ou de *Sockets*.

De referir que para o desenvolvimento da aplicação foi utilizada a ferramenta *Maven* [25], que normalmente é utilizada pela entidade acolhedora.

Nas secções seguintes é efetuada a descrição mais detalhada dos componentes principais da ferramenta desenvolvida.

4.3.1. *WitSoftware Recognition Lib*

Esta biblioteca foi criada com o intuito de simplificar o processo de reconhecimento (deteção e de classificação) de objetos, bem como para automatizar o processo de criação e treino de classificadores. Também se pretendeu facilitar a integração de futuras funcionalidades. Foi utilizada a linguagem *Java* devido à sua portabilidade para diferentes ambientes computacionais.

A *WitSoftware Recognition Lib* implementa os seguintes componentes do servidor: módulo de processamento de conteúdos multimédia, motor de reconhecimento (com classificadores *Haar Cascade*), e módulo de deteção de *frames* contendo um objeto específico (utilizado na deteção automática de intervalos publicitários).

Durante o desenvolvimento do módulo de processamento foi detetado um problema com o processamento dos ficheiros de vídeo, nomeadamente com os que possuem o formato mp4. Esse problema surgiu devido ao facto da biblioteca (API) utilizada (OpenCV) não possuir os *codecs* de vídeo atualizados.

Para ultrapassar essa dificuldade foi utilizada a API JavaCV [26] que é uma interface *Java* para o OpenCV. Apesar de terem sido ultrapassadas as dificuldades com o processamento dos ficheiros de vídeo, surgiu um novo problema relacionado com o facto do JavaCV se ter revelado um concorrente direto do OpenCV. Apesar de ambas APIs utilizarem objetos do tipo *frame*, constatou-se que não eram compatíveis. Foi necessário desenvolver código adicional para suportar o funcionamento simultâneo de ambas as APIs. Atendendo a que existiam vários programadores que se deparavam com o mesmo problema, o código desenvolvido foi disponibilizado às comunidades do OpenCV e do JavaCV. De referir que a deteção desse problema se revelou algo complexa pois foi necessário compreender o funcionamento objetos do tipo *frame*.

O módulo de deteção de *frames* contendo um objeto específico foi desenvolvido com o intuito de auxiliar no processo de deteção automática de intervalos publicitários nos vídeos. Na solução desenvolvida assumiu-se que a publicidade seria identificada através da ausência do logotipo da estação televisiva no canto superior da imagem. De forma a aumentar a eficácia do processo, foi definido um intervalo de confiança para a deteção do logotipo. Assim, quando o logotipo for reconhecido numa determinada *frame* do vídeo, a deteção

seguinte apenas será efetuada após terem decorrido a quantidade de *frames* correspondente ao valor indicado no intervalo de confiança.

Foram também desenvolvidas algumas funcionalidades adicionais na *WitSoftware Recognition Lib*, nomeadamente para suportar a aplicação de filtros a imagens, com o propósito dos classificadores aumentarem a capacidade de deteção dos objetos, e ainda, a gravação e a regravação de conteúdos vídeo. Foi elaborada documentação de auxílio à utilização dessa API.

4.3.2. *WitSoftware Recognition Server*

O *WitSoftware Recognition Server* foi desenvolvido em *Java EE (Java Enterprise Edition*) [27], e efetua a ligação entre o cliente *web* e a API *WitSoftware Recognition Lib*. Tendo por base um sistema REST, obtém os conteúdos multimédia (vídeos ou imagens) carregados pelos utilizadores no cliente *web*, e encaminha-os para o módulo de processamento. Este efetua a decomposição dos vídeos nas respetivas *frames* (imagens), e remete-as sequencialmente para o motor de reconhecimento. Após a conclusão do processo de reconhecimento dos objetos em cada *frame*, os resultados (metadados) são enviados ao módulo de processamento, que os remete para o cliente *web* através de um serviço REST ou de Sockets.

O motor de reconhecimento utiliza classificadores *Haar Cascade* para efetuar a deteção e a classificação de objetos em cada *frame*. Caso os classificadores detetem algum objeto, a sua classe é adicionada a uma lista de resultados. O processo é repetido para todas as *frames* pertencentes a um vídeo.

Durante o desenvolvimento do projeto, foram encontrados alguns problemas com a integração da biblioteca *WitSoftware Recognition Lib* no servidor *Java*, e que se encontravam relacionados com o facto da biblioteca OpenCV utilizar ficheiros do sistema. Foi necessário desenvolver algum código adicional para que essa biblioteca pudesse ser executada num servidor *web*, carregando os ficheiros do sistema somente no caso de não terem sido anteriormente carregados. Uma vez que se está perante um processo que apenas ocorre dentro da JVM (*Java Virtual Machine*), foi necessário criar um outro processo para efetuar esse controlo. A resolução do problema ocupou um período de tempo considerável sobretudo por não se tratar de um erro explícito.

Para efetuar o controlo de acessos por parte de utilizadores, e para efetuar o controlo de erros, foi desenvolvido um sistema de *Logging*.

Por forma a facilitar a configuração do projeto em diferentes ambientes de produção, foi desenvolvido um módulo de configuração de conteúdos. Neste módulo são definidas a localização das diferentes APIs utilizadas, a localização do diretório de armazenamento dos vídeos enviados pelos utilizadores, a localização de ficheiros de *logging*, etc.

4.3.3. *WitSoftware Recognition Client*

O cliente *web* foi desenvolvido com a biblioteca *Javascript AngularJS 1.0* [28], e assegura a captura dos conteúdos multimédia carregados pelo utilizador. É constituído por um sistema de captura de vídeo em tempo integral, por um sistema de captura de vídeo em tempo real e por um sistema de treino de classificadores do tipo *Haar Cascade*.

O sistema de captura de vídeo em tempo integral apresenta ao utilizador um formulário para selecionar o ficheiro (vídeo ou imagem) em que pretende efetuar o reconhecimento (deteção e classificação) de objetos.

O sistema captura de vídeo em tempo real permite ao utilizador disponibilizar o acesso à imagem capturada pela sua câmara digital, sendo tal possível através da tecnologia *Web RTC*.

Após a conclusão do processo de deteção e de classificação dos objetos, o cliente *web* recebe um relatório com os resultados obtidos, e que inclui a quantidade de objetos detetados e a identificação dos classificadores utilizados.

Foi também disponibilizada no cliente *web* a ferramenta desenvolvida para facilitar a criação do *dataset* de imagens positivas (processo supervisionado) utilizado no treino de classificadores do tipo *Haar Cascade*. Com um navegador (*browser*) o utilizador poderá selecionar uma imagem, associar-lhe uma determinada classe, marcar os objetos identificados, e enviar essa informação para o servidor *web*.

4.4. Avaliação da *WitSoftware Recognition Tool v1.0*

Após ter sido confirmada a funcionalidade do protótipo *WitSoftware Recognition Tool v1.0*, e das ferramentas complementares descritas nas secções anteriores, foram efetuados testes para avaliar o seu desempenho e a sua capacidade para detetar de objetos em vídeos e imagens carregados pelo utilizador.

Nos cerca de 50 testes efetuados apenas conseguiram-se obter bons resultados na deteção de faces humanas, sem as conseguir associar a uma determinada pessoa no reconhecimento. Para os restantes tipos de objetos (peças de fruta, porta chaves, canetas, etc.) obtiveram-se taxas de acerto inferiores a 30%.

Também se constatou que a execução do processo de reconhecimento era muito lenta, sendo necessários cerca de 10 minutos para processar um vídeo com 40 segundos de duração.

O processo treino dos classificadores era bastante moroso, demorando cerca de três dias a ser executado. A este tempo terá de ser acrescido o tempo necessário para a recolha e seleção de imagens de qualidade, e a marcação manual dos objetos existentes em milhares de imagens.

Face aos fracos resultados obtidos com o protótipo, e que são originados pela utilização de classificadores do tipo *Haar Cascade* e da API OpenCV na extração de metadados dos vídeos, concluiu-se que seria necessário encontrar uma solução alternativa para a implementação do motor de reconhecimento. Optou-se por explorar a utilização de redes neuronais convolucionais.

4.5. Deteção de intervalos publicitários

Apesar do objetivo principal do estágio ser o de desenvolver uma ferramenta capaz de extrair automaticamente metadados de vídeos, a entidade de acolhimento definiu a deteção automática de intervalos publicitários como sendo o seu caso de uso.

Tal como foi anteriormente referido, os classificadores *Haar Cascade* só poderão ser utilizados no reconhecimento de objetos após terem sido devidamente treinados. Neste

processo de treino tem de ser utilizada uma grande quantidade de imagens distintas para cada tipo de objetos a reconhecer (*dataset* de imagens positivas).

Considerando a enorme diversidade de objetos utilizados nos intervalos publicitários, concluiu-se que não seria possível treinar os classificadores *Haar Cascade* para efetuarem a deteção dos intervalos publicitários. Face a esta situação, ficou comprometida a utilização da deteção automática de intervalos publicitários como caso de uso do protótipo desenvolvido para efetuar o reconhecimento (deteção e classificação) de objetos em conteúdos de vídeo.

Como solução alternativa, optou-se por explorar o facto das estações televisivas habitualmente removerem o seu logotipo quando emitem publicidade. Assim, para efetuar a deteção do logotipo da estação televisiva no canto da imagem, foi desenvolvido o módulo de deteção de *frames* contendo um objeto específico.

Nesse módulo de deteção de *frames* foram utilizadas as seguintes técnicas de visão computacional disponibilizadas na biblioteca OpenCV: deteção de cantos (*corner detection*) [29] e correspondência de modelos (*template maching*) [30].

A técnica de deteção de cantos é utilizada para caracterizar uma imagem, nomeadamente para detetar os seus cantos. Poderá definir-se um canto como o local de interceção de duas arestas, representando o ponto em que se altera a direção dessas arestas. Assim, o gradiente da imagem terá uma grande variação em ambas as direções, sendo esta informação utilizada na deteção dos cantos. Uma das características dessa técnica é a de permitir a deteção do mesmo canto em várias imagens semelhantes, mesmo que existam alterações na iluminação, rotação, translação ou outras transformações.

A técnica de correspondência de modelos identifica partes de uma imagem que correspondem a um modelo pré-definido de imagem (*template*). Trata-se de uma das técnicas mais utilizadas para localizar objetos numa imagem, embora a sua aplicabilidade esteja muito dependente da capacidade computacional disponível, sobretudo para a identificação de modelos grandes ou complexos.

Utilizando a biblioteca na *WITSoftware Recognition Lib*, onde se encontra implementado o módulo de deteção de *frames* com um objeto específico, foi testada a solução desenvolvida para a deteção de intervalos publicitários.

Constatou-se que com a solução desenvolvida foi possível detetar com sucesso o logotipo da estação televisiva, tanto com a aplicação da técnica de deteção de cantos, como com a aplicação da técnica de correspondência de modelos. Verificou-se que nos vídeos existe uma pequena falta de sincronização entre o início do intervalo publicitário e o desaparecimento do logotipo, e entre o final do intervalo publicitário e o aparecimento do logotipo.

O maior problema surgiu com o tempo que a solução necessita para detetar os intervalos publicitários, tendo-se verificado que vídeos com a duração de alguns segundos demoravam alguns minutos a ser processados. Esta situação ocorre com ambas as técnicas utilizadas, e com a utilização do intervalo de confiança na deteção do logotipo da estação televisiva (descrito na secção 4.3.1).

Deste modo, considera-se que esse problema de desempenho compromete a utilização da solução desenvolvida para a deteção de intervalos publicitários em vídeos.

Esta página foi intencionalmente deixada em branco

5. Reconhecimento de objetos com redes neuronais convolucionais

No presente capítulo é descrita a versão 2.0 do protótipo desenvolvido para a deteção e classificação de objetos em conteúdos de vídeo, e que utiliza uma rede neuronal convolucional (a GooGleNet) no motor de reconhecimento, em substituição dos classificadores *Haar Cascade* (utilizados na versão 1.0 do protótipo).

É também apresentada a avaliação efetuada à nova versão do protótipo que resolveu os principais problemas da versão anterior, nomeadamente com as taxas de acerto e com o desempenho no reconhecimento de objetos.

Inicialmente é efetuada uma introdução às redes neuronais e às redes neuronais convolucionais (CNNs, do inglês *Convolutional Neural Networks*), incluindo as suas arquiteturas e métodos de aprendizagem. Encontram-se também descritas as ferramentas desenvolvidas, e os testes efetuados, para avaliar a viabilidade de incorporar no motor de reconhecimento do protótipo redes neuronais convolucionais suportadas pelas APIs *open source* DL4J (*Deep Learning for Java*) e TensorFlow.

5.1. Redes Neuronais Artificiais

Uma rede neuronal artificial trata-se de um modelo computacional inspirado no sistema nervoso central dos animais, em particular o cérebro, e que possui capacidade para efetuar a aprendizagem automática e o reconhecimento de padrões. As redes neuronais artificiais geralmente são apresentadas como sistemas de neurónios interligados com capacidade para efetuar o processamento dos sinais de entrada recebidos. Uma rede neuronal artificial também poder ser definida com sendo um sistema computacional constituído por um conjunto de unidades de processamento (neurónios) interligadas, e que processam a informação através de respostas dinâmicas a entradas externas.

O primeiro modelo conceptual de uma rede neuronal artificial foi desenvolvido, em 1943, pelo neurocientista Warren S. McCulloch e pelo matemático Walter Pitts [31]. A sua ideia era a de modular o neurónio humano com circuitos lógicos e reuni-los para formar uma rede neuronal artificial. Consideravam que a capacidade de processamento do cérebro se

devia ao seu extenso número de neurónios que, apesar de individualmente apenas poderem dar uma resposta binária (ativo ou inativo), em conjunto poderiam dar respostas muito mais complexas.

A primeira geração de redes neuronais surgiu com o modelo designado por *Perceptron* proposto por Frank Rosenblatt em 1958 [32], e analisado por Minsky e Papert em 1969 [33]. O *Perceptron* era definido por duas camadas de neurónios, uma de entrada e outra de saída, e cada neurónio da camada de entrada só possuía ligação com a camada de saída. Não existia ligação entre os neurónios de uma mesma camada, nem ligação da saída para a entrada.

Um neurónio artificial é capaz de efetuar um único processamento. Cada entrada recebe somente um tipo de sinal, ou informação, e possui um peso que determina a influência do sinal no processamento (função de ativação) que calcula o resultado de saída. Cada neurónio pode possuir várias entradas, pelo que poderá receber diferentes sinais. A ligação de vários neurónios permite ao sistema processar mais informações, e produzir mais resultados.

Por exemplo, poderá criar-se um sistema para identificação das seguintes frutas: bananas e maçãs vermelhas. Para tal, terão de se criar neurónios sensíveis à cor e à forma. Os neurónios sensíveis à cor para identificar o amarelo (da banana) e o vermelho (da maçã). Os neurónios sensíveis à forma identificarão o redondo e o comprido. Assim, cada neurónio possuirá quatro entradas, uma para cada informação (amarelo, vermelho, redondo e comprido). Para obter um melhor rendimento do sistema poderá criar-se uma rede em camadas: uma primeira camada com quatro neurónios (um para cada tipo de informação), uma segunda camada oculta de processamento com três neurónios, e uma camada de saída com dois neurónios, um para avisar quando for detetada uma maçã, e outro para avisar quando for detetada uma banana.

Uma rede neuronal é um sistema computacional cognitivo, em contraste com um sistema que normalmente efetua o processamento sequencial de instruções. Nas redes neuronais a informação é processada em paralelo pelos vários nós da rede, ou neurónios, sendo de destacar a sua capacidade de aprendizagem. Este tipo de rede é adaptativo pois tem a capacidade de alterar a estrutura interna com base na informação que processa, nomeadamente através das respostas que classifica como "incorretas" em função do objetivo pretendido.

Esse processo de aprendizagem é geralmente designado como o treino da rede, e pode ocorrer de forma supervisionada ou não supervisionada. Se não forem indicados os resultados ideais dos seus processamentos, um determinado treino é designado como não supervisionado. Se forem indicados os resultados esperados do processamento da rede neuronal, o treino é designado como supervisionado.

Ambos os tipos de treino são processos iterativos. No treino supervisionado é calculado o erro entre o resultado esperado e o resultado obtido, e os pesos das entradas dos neurónios são ajustados de forma a reduzir esse erro a um valor aceitável.

No treino não supervisionado não é possível efetuar o cálculo desse erro devido à inexistência de um resultado esperado. Em alternativa, realiza-se um determinado número de iterações com o intuito de identificar padrões na informação processada e a treinar a rede neuronal. Caso a rede não produza os resultados desejados, poderá aumentar-se a quantidade de iterações efetuadas durante o treino.

Nem todas as redes neuronais são construídas para resolver todos os tipos de problemas. O programador deverá identificar o tipo de rede neuronal mais adequado ao domínio de problema que pretende resolver.

5.1.1. Arquitetura de uma rede neuronal

Uma rede neuronal artificial é normalmente constituída por várias camadas, a camada de entrada (*input layer*), as camadas escondidas (*hidden layers*) e a camada de saída (*output layer*).

Os neurónios da camada de entrada são responsáveis por introduzir na rede os dados a processar, dado que o seu potencial vai ser o valor dos dados de entrada. Após o carregamento destes dados na camada de entrada, os neurónios da primeira camada escondida utilizam-nos para calcular o seu potencial. No cálculo são utilizados os pesos das ligações entre os neurónios de diferentes camadas, podendo esses pesos ser ajustados em cada ciclo (*epoch*), de forma a minimizar o erro da rede na resolução de um determinado problema. Para além dos pesos é também utilizado o valor do *bias* associado a cada neurónio, e que normalmente é modificado durante a fase de treino da rede, tal como sucede com os pesos.

O processamento efetuado em cada neurónio é designado como função de ativação, que processa o produto interno entre os pesos e as entradas dos neurónios, acrescido do valor do *bias*. Existem diversas funções de ativação que podem ser utilizadas [34], dependendo da aplicação e da forma como a rede neuronal artificial for projetada.

Algumas das funções de ativação mais utilizadas pelos neurónios são as seguintes:

- Função de ativação Linear

  É a função de ativação mais básica pois não altera a saída de um neurónio, refletindo na saída o valor da soma pesada das entradas. Esses neurónios são utilizados em problemas de regressão linear.

- Função de ativação com limite (degrau)

  Esta função define uma saída binária (0 ou 1) de acordo com um limite (*threshold*) estabelecido. Nas aplicações iniciais de redes neuronais, estas funções eram designadas por *Perceptrons*. Essas aplicações apenas ativavam a saída (valor 1) quando a combinação de entradas ponderadas por pesos fosse maior ou igual a um determinado limite. Caso contrário, a saída era desativada (valor 0).

- Função de ativação Sigmoide

  Este tipo de função de ativação não linear é normalmente utilizado em redes neuronais com propagação positiva (*feedfoward*) que necessitam de ter a sua saída no intervalo]0, 1[. Apesar de ser muito utilizada, nomeadamente na realização de treinos, a função de ativação Sigmoide poderá ser substituída por funções de ativação que utilizam a tangente hiperbólica ou a ReLU (Unidade Linear Retificada).

- Função de ativação Tangente Hiperbólica

  Este tipo de função de ativação tem a sua saída no intervalo [-1,1], e o valor da saída corresponde à tangente hiperbólica da entrada. Essa função costuma ser utilizada na realização de treinos, especialmente os que envolvam derivativos, como o de retro-propagação (*Back Propagation*).

- Unidade Linear Retificada (ReLU)

  Este tipo de função de ativação foi introduzido no ano de 2000, e é atualmente uma das mais populares sobretudo em processos de aprendizagem profundos. A ReLU é bastante recomendada por ser linear e não saturante, sendo implementada com

funções menos complexas que a sigmoide e a tangente hiperbólica, que utilizam exponenciais.

- Função de ativação Softmax

  Este tipo de função é utilizado em redes neuronais de classificação, e é adequada para bases de dados não balanceadas por possuir um neurónio de saída para cada classe definida. A função obriga a que a saída da rede represente a probabilidade dos dados serem de uma das classes definidas. Sem a utilização deste tipo de função, as saídas dos neurónios são simplesmente valores numéricos em que o maior valor indica a classe vencedora.

5.1.2. Método de aprendizagem

A aprendizagem da rede neuronal consiste em ajustar os pesos da rede por forma a minimizar o erro cometido entre a saída da rede e o objetivo pretendido. Para tal utiliza-se um conjunto de treino e um conjunto de teste para avaliar o desempenho da rede.

A atualização dos pesos da rede faz-se através da minimização do erro dado por uma função de custo, sendo as mais comuns o erro quadrático para a regressão, e a *cross-entropy* para a classificação. Os algoritmos baseados na descida do gradiente são os mais utilizados no processo de minimização do erro. O treino da rede pode ser feito usando a abordagem *Batch* ou a abordagem *Online*. Na abordagem *Batch* cada atualização dos pesos é efetuada após a apresentação à rede de todos os casos do conjunto de treino, enquanto que na abordagem *Online* a atualização dos pesos é efetuada após a apresentação de cada caso. À passagem completa de todos os casos pela rede dá-se o nome de ciclo (*epoch*).

O algoritmo de retro-propagação (*Back Propagation*) do erro é utilizado nos treinos de redes neuronais, e pretende que a rede otimize os seus pesos em função do problema. Este algoritmo tem como objetivo introduzir nos pesos da rede o ajustamento necessário para que, com o decorrer do treino, seja cada vez menor a diferença entre a resposta obtida e a resposta esperada.

Após a fase de propagação é necessário verificar quão longe está a rede das respostas corretas para os vários casos de treino, sendo o erro dos neurónios das camadas finais calculado por essa diferença, enquanto que o erro dos restantes neurónios é calculado a partir dos neurónios da sua camada seguinte. Deste modo, os neurónios da penúltima camada

utilizam o erro dos neurónios da última camada, e os neurónios da camada *n* utilizam o erro dos que se encontram na camada *n*+1, sendo o erro propagado de camada em camada até chegar à primeira. Para o cálculo deste erro é também necessário calcular a derivada da função de ativação em relação ao somatório dos pesos da rede.

Em [35] [36] são apresentadas diversas redes neuronais que, com a utilização do algoritmo de retro-propagação, alcançam resultados de aprendizagem muito mais rapidamente do que com as abordagens anteriormente utilizadas.

5.2. Redes Neuronais Convolucionais (CNNs)

As redes neuronais convolucionais surgiram devido à motivação biológica presente no trabalho de Hubel e Wiesel [37] que, ao efetuarem testes com o Córtex Visual de gatos, descobriram que tanto as células simples, como as complexas, são sensíveis a certos padrões e orientações.

O interesse comercial nas redes neuronais convolucionais (CNNs) ocorreu quando, em 2012, Krizhevsky et al. [38] ganharam o concurso de reconhecimento de objetos promovido pela *ImageNet* (maior base de dados mundial de imagens), embora as redes convolucionais já tivessem anteriormente vencido vários concursos, com menor impacto, na área da visão computacional.

As redes neuronais convolucionais consistem num conjunto de camadas que extraem características das imagens de entrada através de sucessivas convoluções e redimensionamentos. No final desse processo pretende-se ficar apenas com a marca da classe a que a imagem de entrada pertence.

Uma das características mais importantes das CNNs é que são redes fáceis de treinar, e com muito menos parâmetros do que redes totalmente ligadas contendo o mesmo número de camadas escondidas. As redes neuronais multicamada descritas nas secções anteriores, quando são utilizadas com imagens geram um problema de alta magnitude. Por exemplo, utilizando uma rede totalmente ligada com uma camada oculta de 5000 neurónios em conjunto com o *dataset* CIFAR-10 [39], que possui imagens de 32x32x3 pixels, geraria 153.600.000 pesos para ter uma de taxa de acerto de 60% na classificação das suas imagens

em 10 classes diferentes [40]. Enquanto isso, uma arquitetura simples de CNN com 4.210.090 parâmetros pode alcançar uma taxa de acerto de 71,82% nessa classificação [41].

5.2.1. Arquitetura de uma CNN

Uma CNN é composta por várias camadas convolucionais, normalmente intercaladas com camadas de sub-amostragem, ligadas entre si, exatamente como numa rede neuronal multicamada padrão. A arquitetura de uma CNN foi concebida para tirar partido da estrutura bidimensional (2D) de uma imagem, ou de um ficheiro áudio digital, usando-os como entradas. Tal processo é conseguido através de ligações locais controladas por pesos, seguido por um tipo de agrupamento, o que resulta na tradução de características em informação invariante.

O objeto de entrada numa camada convolucional é uma imagem com três parâmetros referentes à sua altura, largura e número de canais (possui valor 1 para imagem a preto e branco, e o valor 3 para imagem a cores). A camada convolucional possui um conjunto de filtros (*kernels*) com um tamanho inferior à dimensão da imagem de entrada. O objetivo de cada filtro é o de extrair uma determinada característica da imagem, criando um mapa de ativação com duas dimensões, e com tamanho inferior ao da imagem de entrada. Após a fase da convolução segue-se uma sub-amostragem de modo a diminuir, para cerca de metade, a quantidade de informação e a agregar as características recolhidas pela convolução. O objetivo é o de reduzir a quantidade de mapas de ativação que transitam para a camada de convolução seguinte [42]. Os processos de convolução e de sub-amostragem são repetidos até se obter uma camada final com o tamanho desejado, normalmente 1x1.

O *Neocognitron* proposto Fukushima [43] é um bom exemplo de uma rede neuronal convolucional utilizada para o reconhecimento de dígitos. Nesta rede foram criadas as camadas S e C cujos neurónios têm funções bastantes semelhantes, respetivamente, às células simples e às células complexas do córtex visual primário. Deste modo, o *Neocognitron* tenta simular o processamento de informação visual que o nosso cérebro realiza tentado que as camadas S retirem características da imagem como bordas, arestas e orientações, enquanto que as camadas C tentam generalizar as informações enviadas pelas camadas S, garantindo alguma invariância em relação à posição do padrão que as células S tentam transmitir à camada seguinte.

Numa rede convolucional como o *Neocognitron* podem existir vários conjuntos de camadas S e C. À medida que se avança nas camadas da rede as características começam a ser cada vez mais globais em função do dígito inicial, independentemente da sua posição e escala. Com este tipo de rede Fukushima conseguiu obter resultados bastante satisfatórios no reconhecimento de dígitos manuscritos [44] [45].

5.2.2. API DL4J (*Deep Learning for Java*)

Com o intuito de utilizar as redes neuronais convolucionais (CNNs) para a extração de metadados de vídeos, e na sequência de uma pesquisa efetuada, optou-se por testar a API DL4J (*Deep Learning for Java*) [46]. Trata-se de uma biblioteca *open source*, desenvolvida com a linguagem *Java*, e que suporta a utilização de várias CNNs para a resolução de problemas associados à visão computacional.

O facto da API DL4J encontrar-se desenvolvida em *Java* foi relevante na opção tomada, pois facilitaria a sua integração no motor de reconhecimento do protótipo desenvolvido (*WitSoftware Recognition Tool*). A deteção e a classificação de objetos em imagens (*frames*) passaria a ser efetuado por uma rede convolucional, em detrimento dos classificadores *Haar Cascade*.

Inicialmente a biblioteca DL4J foi utilizada para testar a utilização de redes CNNs na resolução de problemas simples, mas exemplificativos, da área da visão computacional. Por exemplo, foi testada a deteção de objetos no *dataset* MNIST [47] que é constituído por 1000 imagens a preto e branco, com tamanho 28x28, contendo um algarismo manuscrito (0 a 9). Para tal foi utilizada a rede convolucional LeNet [48], que foi projetada para a resolução desse problema, tendo-se obtido, em cerca de 10 minutos, uma rede com uma taxa de acerto de cerca de 99%.

Atendendo a que se pretendia obter uma rede convolucional (CNN) treinada para a deteção de objetos em imagens a cores, foram analisados alguns *datasets* constituídos por imagens a cores com um tamanho superior ao utilizado no teste anteriormente descrito. Optou-se pelo *dataset* Cifar-10 [40] que é constituído por 60.000 imagens a cores, com o tamanho 32x32, distribuídas por 10 classes (tipo de objetos) diferentes. Para cada tipo de objeto existem 5.000 imagens de treino, existindo um *dataset* de teste com 10.000 imagens.

Tentou utilizar-se a rede convolucional LeNet com o *dataset* Cifar-10, tendo-se verificado que tal não era possível. Concluiu-se que as 10 camadas da rede LeNet, bem como a sua função de ativação, não são a escolha acertada para efetuar o reconhecimento de objetos em imagens a cores. Essa rede está apenas preparada para extrair informação de imagens a preto e branco de tamanho pequeno (28x28).

Face ao insucesso da utilização da rede LeNet no reconhecimento de objeto em imagens a cores, após mais uma pesquisa, optou-se por recorrer às seguintes três CNNs vocacionadas a resolver esse tipo de problema: a rede Alexnet [38], e as e as redes SVGG A [49] e SVGG B [49].

Utilizando a linguagem *Java*, foram desenvolvidas 3 classes, cada uma responsável por implementar uma dessas redes. De seguida apresenta-se um excerto exemplificativo de uma dessas implementações (omitindo grande parte do código).

```
MultiLayerConfiguration.Builder conf = new
NeuronalNetConfiguration.Builder()
.seed(seed)
.weightInit(WeightInit.DISTRIBUTION)
.dist(new NormalDistribution(0.0, 0.01))
.activation("relu")
.updater(Updater.NESTEROVS)
.iterations(iterations)
.optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
.learningRate(1e-3)
.learningRateScoreBasedDecayRate(1e-1)
.regularization(true)
.l2(5 * 1e-4)
.momentum(0.9)
.miniBatch(false)
.list()
.layer(0, new ConvolutionLayer.Builder(new int[]{11, 11}, new int[]{4, 4},
new int[]{3, 3})
.name("cnn1")
.nIn(channels)
.nOut(96)
.build())
...
.layer(2, new SubsamplingLayer.Builder(poolingType, new int[]{3, 3}, new
int[]{2, 2})
.name("maxpool1")
.build())
```

```
...
.layer(10, new DenseLayer.Builder()
.name("ffn1")
.nOut(4096)
.dist(new GaussianDistribution(0, 0.005))
.biasInit(nonZeroBias)
.dropOut(dropOut)
.build())
...
.layer(12, new
OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
.name("output")
.nOut(outputNum)
.activation("softmax")
.build())
.backprop(true).pretrain(false)
```

Ao analisar uma rede deste tipo verifica-se que existem muitos valores parametrizáveis, várias funções de ativação diferentes, e camadas com capacidade para aplicar diferentes transformações. Atendendo a que é necessário adaptar a configuração das redes neuronais (e CNNs) ao *dataset* de imagens utilizado, tornava-se necessário apurar quais os melhores valores desses parâmetros para o *dataset* Cifar-10. Assim, foi desenvolvido código para avaliar o desempenho das redes para várias configurações desses parâmetros.

Inicialmente a avaliação era efetuada de uma forma sequencial, realizando-se um teste exaustivo das possíveis configurações. Para cada um dos parâmetros foi criada uma lista de possíveis valores. Cada treino da rede foi executado durante 5 ciclos, considerando-se que são suficientes para aferir a sua capacidade de aprendizagem. No final de cada ciclo era obtido o valor numérico correspondente à capacidade de aprendizagem da rede. A fase de treino concluía-se quando o valor numérico produzido num ciclo fosse inferior ao valor do ciclo anterior, visto que se reduzia a capacidade de aprendizagem da rede, ou no final da quantidade de ciclos (5) definidos.

Após a conclusão da fase de treino, era gerado o modelo referente ao conhecimento adquirido pela rede, e que consiste num ficheiro contendo dados que a rede utilizará para efetuar o reconhecimento de objetos. O *dataset* de teste era posteriormente utilizado para avaliar essa capacidade de reconhecimento da rede, sendo calculado um valor percentual

correspondente a essa capacidade. Este valor era armazenado por forma a poder aferir-se qual a configuração que obteve o melhor resultado.

Considerando a utilização de 5 valores possíveis para cada um dos valores parametrizáveis, seria necessário testar cerca de 65.000 configurações. De forma a reduzir o tempo necessário para os testes, optou-se por testar 5 configurações em simultâneo, e selecionadas de uma forma aleatória. Foi também adicionada uma condição adicional na seleção da melhor configuração da rede convolucional, eliminando-se todas as configurações que não obtivessem nos testes uma taxa de acerto mínima de 80%.

Durante essa avaliação para tentar obter a melhor configuração para as redes, ocorreram bastantes problemas causados pela falta de memória para a conclusão dos processos, sendo estes prematuramente terminados por falta de recursos do sistema. Para tentar ultrapassar essa situação, foi aumentada a capacidade de memória da JVM de forma a disponibilizar 10 Gb ao processo. Para melhorar o desempenho do sistema alterou-se também o código de forma a reduzir a memória ocupada, substituindo o tipo de dados *double* pelo tipo de dados *float*.

Com essas alterações obtiveram-se algumas melhorias no sistema, embora só se conseguisse completar com sucesso as fases de treino e de teste com a redução da quantidade de imagens utilizadas nessas fases. Nestes casos foram obtidas taxas de acerto bastante reduzidas nos testes.

Face aos problemas anteriormente descritos, e que impossibilitaram a preparação de uma CNN para a deteção de objetos em imagens a cores (*dataset* CIFAR-10), concluiu-se que se teria de encontrar uma alternativa API DL4J. Apesar do insucesso obtido, esta fase foi útil para melhor compreender a forma de funcionamento das redes neuronais convolucionais.

5.2.3. API *TensorFlow*

Como alternativa à API DL4J, que não foi possível aplicar ao reconhecimento de objetos em imagens a cores, decidiu-se procurar uma biblioteca (API) desenvolvida numa linguagem de programação (*Python*) mais vocacionada para a resolução de problemas que necessitem de uma grande capacidade computacional.

Na sequência das pesquisas efetuadas, optou-se por analisar a API TensorFlow [50] que foi criada pela Google. Foram efetuados diversos testes para aferir a aplicabilidade desta API para o reconhecimento de objetos, e para tal foi necessário adaptar, e portar para a linguagem *Python*, muito do código desenvolvido para o teste da API DL4J .

Começou por se criar uma rede convolucional (CNN) para a deteção de objetos no *dataset* MNIST. Posteriormente foi criada uma CNN que foi treinada e testada com o *dataset* CIFAR-10. Em ambos os casos obtiveram-se excelentes resultados para as taxas de acerto das CNNs. Note-se que estes testes também haviam sido feitos na API DL4J, não tendo sido possível obter resultados satisfatórios com o *dataset* CIFAR-10 (composto por imagens a cores de 10 tipos de objetos).

No teste seguinte utilizou-se o *dataset ImageNet* [51], que é a maior base de dados mundial de imagens, catalogando milhares de classes (tipos de objetos), e que é considerada a maior referência na área das redes convolucionais. Todos os anos existe uma competição que seleciona a rede que obteve os melhores resultados utilizando esse *dataset*.

Optou-se também por aplicar a esse *dataset* a CNN GoogLeNet [52] que possui 22 camadas e é vocacionada para a deteção e classificação de objetos em imagens. Esta rede convolucional, que consegue obter uma taxa de acerto de 98%, venceu em 2014 a competição promovida pela *ImageNet*. Note-se que não foi possível utilizar a CNN que obteve os melhores resultados no ano em 2015 [53], e que alcançou uma taxa de acerto de 99.3%, pois, até ao momento em que se efetuaram os testes, a empresa que a desenvolveu (Microsoft) não possuía métodos que permitissem a sua utilização por parte da API TensorFlow.

Tal como era expectável, também se obtiveram excelentes resultados nos testes realizados com a CNN GoogLeNet treinada com o *dataset ImageNet*, pelo que se decidiu utilizar a GoogLeNet (suportada pela API TensorFlow) no motor de reconhecimento para efetuar a deteção e a classificação dos objetos existentes nos conteúdos multimédia (vídeo ou imagem) fornecidos pelo utilizador.

5.3. *WitSoftware Recognition Tool* v2.0

Foi desenvolvida a versão 2.0 da ferramenta *WitSoftware Recognition Tool* com o intuito de incorporar a rede neuronal convolucional GoogLeNet (suportada pela API TensorFlow) no motor de reconhecimento, em substituição dos classificadores *Haar Cascade* (suportados pela API OpenCV).

A nova versão da ferramenta foi desenvolvida com a linguagem de programação *Python*, tendo sido necessário implementar uma interface gráfica (*desktop*) para se poder apresentar a prova de conceito à entidade acolhedora do estágio. Foram incorporadas as restantes funcionalidades existentes na versão 1.0, nomeadamente o sistema de captura de vídeo em tempo integral, o sistema de captura de vídeo em tempo real utilizando a câmara digital, o módulo de processamento e o motor de reconhecimento.

O utilizador poderá selecionar, através de uma interface gráfica, um ficheiro multimédia (imagem ou vídeo integral) ou uma câmara digital (captura em tempo real). Os conteúdos captados são encaminhados para o módulo de processamento, que efetua a decomposição dos vídeos nas respetivas *frames* (imagens), e remetendo-as sequencialmente para o motor de reconhecimento. Após a conclusão do processo de reconhecimento dos objetos em cada *frame*, os resultados (metadados) são enviados ao módulo de processamento, que os apresenta na interface gráfica através de *labels* com a informação da classe dos objetos detetados.

5.4. Avaliação da *WitSoftware Recognition Tool v2.0*

Após ter sido confirmada a funcionalidade da ferramenta *WitSoftware Recognition Tool v2.0*, foram efetuados vários testes para avaliar a sua capacidade para detetar objetos em ficheiros de vídeos carregados pelo utilizador, e em vídeos capturados em tempo real (através da câmara digital do computador).

Com os testes realizados pretendeu-se sobretudo analisar a viabilidade de utilizar a nova versão da ferramenta na resolução do problema principal que se pretendia ultrapassar no âmbito do estágio, o de desenvolver um sistema de criação automática de metadados a partir da informação extraída de vídeos.

Após ter-se utilizado o *dataset ImageNet* no treino da rede convolucional selecionada (GoogLeNet), foram efetuados testes em 3 vídeos carregados pelo utilizador e num vídeo capturado pela câmara digital do computador.

Os vídeos carregados pelo utilizador continham, respetivamente, peças de fruta (banana, maçã, laranja, limão), viaturas (automóvel, mota, autocarro) e diferentes raças de cães (labrador retriever, pastor alemão, rafeiro). Nos três vídeos foi possível extrair os metadados com a correta identificação dos seus conteúdos, tendo apenas existido algumas dificuldades na distinção de objetos semelhantes, nomeadamente entre uma laranja e um limão.

Utilizando a câmara digital do computador foram capturados alguns objetos em tempo real (televisão, caneta, computador portátil, monitor de computador), e também foi possível extrair os metadados referentes à sua identificação. Foram confirmadas as dificuldades na distinção de objetos semelhantes, nomeadamente entre a televisão e o monitor de um computador.

Apesar das dificuldades indicadas, que por vezes também ocorrem na identificação visual de objetos semelhantes, confirmaram-se as altas taxas de acerto (superiores a 90%) na deteção e na identificação (com metadados) de objetos.

Também se constatou que o desempenho da aplicação melhorou consideravelmente, tendo sido ultrapassado o problema existente na versão anterior do protótipo, e referente à grande lentidão com que os metadados eram extraídos dos conteúdos de vídeo.

Deste modo, pode concluir-se que com a *WitSoftware Recognition Tool v2.0* foi possível desenvolver um sistema fiável de criação automática de metadados a partir da informação extraída de vídeos, e alcançar os objetivos definidos para o estágio.

6. Conclusão

Com o intuito de efetuar a identificação automática dos conteúdos de um vídeo, foi proposto um estágio cujo objetivo principal era o de desenvolver um protótipo para a criação automática de metadados a partir do reconhecimento de objetos existentes num vídeo.

Na sequência da análise efetuada a várias ferramentas para a deteção e a classificação de objetos em ficheiros de vídeo, optou-se por utilizar a API OpenCV no desenvolvimento da primeira versão do protótipo, denominado *WitSoftware Recognition Tool*. Foi também definida a arquitetura cliente/servidor para a aplicação, em que o cliente disponibiliza ao utilizador as ferramentas para o carregamento de conteúdos multimédia (vídeo e imagem). O servidor processa esses conteúdos, decompondo cada vídeo nas respetivas *frames* (imagens), e efetua a deteção dos objetos em cada *frame*, criando os metadados correspondentes à classificação efetuada.

A versão inicial do protótipo (*WitSoftware Recognition Tool v1.0*) utiliza classificadores *Haar Cascade* (suportados pela API OpenCV) para efetuar o reconhecimento dos objetos. Nos testes efetuados para avaliar o protótipo, apenas foram obtidos bons resultados na deteção de faces humanas, sendo inferiores a 30% as taxas de acerto no reconhecimento dos restantes tipos de objetos (peças de fruta, porta chaves, canetas, etc.). Também se constatou que a execução do processo de reconhecimento dos objetos era muito lenta, sendo necessários cerca de 10 minutos para processar um vídeo com a duração de 40 segundos. O processo treino dos classificadores também era bastante moroso, demorando mais de três dias a ser executado, mesmo com a utilização de ferramenta desenvolvidas para automatizar e para facilitar a criação e a marcação das imagens utilizadas.

Face aos resultados obtidos na versão inicial do protótipo, e que são originados pela utilização de classificadores do tipo *Haar Cascade* e da API OpenCV no reconhecimento de objetos, optou-se por utilizar uma rede neuronal convolucional (CNN) para executar essa tarefa. Inicialmente foi avaliada a utilização da API DL4J (*Deep Learning for Java*), que tem implementadas várias CNNs para o reconhecimento de objetos e que está desenvolvida em *Java*, o que permitiria minimizar as alterações a efetuar ao protótipo. Nos testes efetuados não foi possível treinar convenientemente uma CNN para obter bons resultados no reconhecimento de objetos em imagens a cores, tendo ocorrido vários problemas com os recursos computacionais requeridos. Como alternativa à API DL4J testou-se a API

TensorFlow, desenvolvida na linguagem *Python*, tendo-se obtido excelentes resultados com a utilização de CNNs, e dos respetivos *datasets*, no reconhecimento de objetos nos vários tipos de imagens. Atendendo às características do problema, decidiu-se utilizar a CNN GoogLeNet, treinada com o *dataset Imagenet*, para efetuar a deteção e a classificação dos objetos em imagens.

Foi desenvolvida a versão 2.0 do protótipo (*WitSoftware Recognition Tool v2.0*) que utiliza a rede neuronal convolucional GoogLeNet, suportada pela API TensorFlow e treinada com o *dataset ImageNet*, para efetuar o reconhecimento de objetos em imagens. A nova versão do protótipo teve de ser desenvolvida na linguagem de programação *Python*, e incorporou todas as funcionalidades existentes na versão inicial.

Nos testes efetuados com *WitSoftware Recognition Tool v2.0* obtiveram-se elevadas taxas de acerto (superiores a 90%) na deteção e na classificação (com metadados) dos objetos existentes em vídeos. Surgiram apenas algumas dificuldades na distinção de objetos semelhantes, nomeadamente entre uma televisão e um monitor de computador, e que por vezes também ocorrem durante a sua identificação visual. O desempenho do protótipo também melhorou significativamente, tendo sido resolvidos os problemas da morosidade no reconhecimento de objetos, e no treino do sistema de reconhecimento.

Pode-se concluir que foi desenvolvido um protótipo fiável para a criação automática de metadados a partir da informação extraída de vídeos, e que foi alcançado o objetivo principal definido para o estágio. Para tal foi necessário ultrapassar as várias dificuldades descritas no documento, sobretudo através da pesquisa e do desenvolvimento de várias soluções para a resolução dos problemas que foram surgindo.

Bibliografia

[1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2016–2021.” [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html. [Accessed: 01-Oct-2016].

[2] N. H. Anne Austin, Jonathan Barnard, “Online Video Forecasts 2015,” *ZenithOptimedia*. [Online]. Available: https://www.zenithmedia.com/product/online-video-forecasts-2016/. [Accessed: 20-Jun-2016].

[3] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Comput. Vis. Pattern Recognit.*, vol. 1, p. I--511--I--518, 2001.

[4] OpenCV, “OpenCV (Open Source Computer Vision Library).” [Online]. Available: http://opencv.org/. [Accessed: 01-Jul-2017].

[5] L. Thompson, “Key Differences Between Speech Recognition and Voice Recognition.”

[6] F. Rumsey and R. Auld, “Stokowski and the evolution of recording techniques,” *AES J. Audio Eng. Soc.*, vol. 60, no. 7–8, pp. 627–630, 2012.

[7] J. B. Allen, “Harvey Fletcher’s role in the creation of communication acoustics.,” *Jasa*, vol. 99, no. 4, pp. 1825–1839, 1996.

[8] F. Juang and L. Rabiner, “Automatic speech recognition: A brief history of the technology development,” in *Encyclopedia of Language and Linguistics*, 2005.

[9] D. R. Reddy, “Speech recognition by machine: a review,” *Proc. IEEE*, vol. 64, no. 4, pp. 501–531, 1976.

[10] J. A. Larson, “W3C speech interface languages: VoiceXML,” *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 126–131, 2007.

[11] Apple Inc, “Apple SIRI Homepage.” [Online]. Available: http://www.apple.com/iphone/features/siri.html. [Accessed: 30-Jun-2016].

[12] X. Huang *et al.*, “MIPAD: A multimodal interaction prototype,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 1, pp. 9–12, 2001.

[13] Google, “Google Now homepage.” [Online]. Available: http://www.google.co.uk/landing/now/. [Accessed: 30-Jun-2016].

[14] Microsoft, “What is Cortana?,” *https://support.microsoft.com*, 2017. [Online]. Available: https://support.microsoft.com/en-us/help/17214/windows-10-what-is.

[15] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope, “Deploying GOOG-411: Early lessons in data, measurement, and testing,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2008, pp. 5260–5263.

[16] G. Hinton *et al.*, “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[17] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: an overview,” *2013 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 8599–8603, 2013.

[18] X. Chen, A. Eversole, G. Li, D. Yu, and F. Seide, “Pipelined back-propagation for context-dependent deep neural networks,” *Proc. Interspeech*, pp. 2–5, 2012.

[19] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 20, no. 1, pp. 30–42, 2012.

[20] J. Dean *et al.*, “Large Scale Distributed Deep Networks,” *NIPS 2012 Neural Inf. Process. Syst.*, pp. 1–11, 2012.

[21] MathWorks, “Train a Cascade Object Detector.” [Online]. Available: http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html#btugex8. [Accessed: 30-Jun-2016].

[22] M. Rezaei, “Creating a Cascade of Haar-Like Classifiers: Step by Step,” 2013. [Online]. Available: https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf. [Accessed: 30-Jun-2017].

[23] OpenCV, “Cascade Classifier Training.” [Online]. Available: http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html. [Accessed: 30-Mar-2016].

[24] N. Seo, “Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features).” [Online]. Available: http://note.sonots.com/SciSoftware/haartraining.html. [Accessed: 30-Mar-2016].

[25] B. M. Mccullough, “Apache Maven,” *Syntax*, pp. 1–6, 2009.

[26] S. Audet, “JavaCV homepage.” [Online]. Available: https://github.com/bytedeco/javacv. [Accessed: 30-Jun-2016].

[27] Oracle, “Java EE at a Glance.” [Online]. Available: http://www.oracle.com/technetwork/java/javaee/overview/index.html. [Accessed: 31-Jul-2017].

[28] Google, “Angular JS homepage.” [Online]. Available: https://angularjs.org/. [Accessed: 31-Jul-2017].

[29] OpenCV, “Harris Corner Detector in OpenCV.” [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html. [Accessed: 31-Jul-2017].

[30] OpenCV, “Template Matching in OpenCV.” [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html#py-template-matching. [Accessed: 31-Jul-2017].

[31] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[32] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[33] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, vol. 165. 1969.

[34] A. Laudani, G. M. Lozito, F. R. Fulginei, and A. Salvini, “On training efficiency and computational costs of a feed forward neural network: A review,” *Computational Intelligence and Neuroscience*, vol. 2015. 2015.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. pp. 318–362, 1986.

[37] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, vol. 160, no. 1, p. 106–154.2, 1962.

[38] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9, 2012.

[39] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” *... Sci. Dep. Univ. Toronto, Tech. ...*, pp. 1–60, 2009.

[40] S. Shiba, “Cifar-10 neural-net classifiers.” [Online]. Available: https://github.com/shiba24/cifar-10. [Accessed: 15-Dec-2016].

[41] J. Brownlee, “Object recognition with convolutional neural networks in the keras deep learning library.” [Online]. Available: https://machinelearningmastery.com/object-recognition-convolutional-neural-networks-keras-deep-learning-library/. [Accessed: 15-Dec-2016].

[42] J. Cong and B. Xiao, “Minimizing computation in convolutional neural networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8681 LNCS, pp. 281–290.

[43] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[44] K. Fukushima, “Neocognitron for handwritten digit recognition,” *Neurocomputing*, vol. 51, pp. 161–180, 2003.

[45] K. Fukushima, “Restoring partly occluded patterns: A neural network model,” *Neural Networks*, vol. 18, no. 1, pp. 33–43, 2005.

[46] Skymind, “Deep Learning for Java homepage.” [Online]. Available: https://deeplearning4j.org/. [Accessed: 15-Dec-2016].

[47] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” *AT&T Labs [Online]. Available http//yann. lecun. com/exdb/mnist*, 2010.

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[49] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *Int. Conf. Learn. Represent.*, pp. 1–14, 2015.

[50] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” *Google Brain*, p. 18, 2016.

[51] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[52] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07–12–June, pp. 1–9.

[53] K. Ovtcharov, O. Ruwase, J. Kim, J. Fowers, K. Strauss, and E. S. Chung, “Accelerating Deep Convolutional Neural Networks Using Specialized Hardware,” *Microsoft Whitepaper*, pp. 3–6, 2015.

Anexos

Esta página foi intencionalmente deixada em branco

Anexo 1: Reconhecimento de voz - soluções

Durante o levantamento do estado da arte foram identificadas as seguintes soluções para o reconhecimento de voz:

1) Soluções do tipo *Desktop*

Designação: Voice Navigator
Sistema Operativo: Mac OS
Empresa: Articulate Systems
Estado: Descontinuado (1999)
Tipo de Solução: Comercial
URL: https://en.wikipedia.org/wiki/Voice_Navigator
Descrição: Primeiro software/hardware de reconhecimento de voz para controlo de uma interface gráfica.

Designação: ViaVoice
Sistema Operativo: Mac OS
Empresa: IBM
Estado: Descontinuado (2007)
Tipo de Solução: Comercial
URL: http://www-01.ibm.com/software/pervasive/viavoice.html
Descrição: Primeiro software da IBM para reconhecimento de voz, tendo sido posteriormente vendido à empresa Nuance.

Designação: Speakable items / Automator
Sistema Operativo: Mac OS
Empresa: Apple
Estado: Em desenvolvimento
Tipo de Solução: Comercial
URL: https://support.apple.com/pt-pt/HT2488
Descrição: Ferramenta do sistema operativo Mac OSX atualmente designada por Automator.

Designação: Power Secretary

Sistema Operativo: Mac OS

Empresa: G-Power

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.gpower.co/

Descrição: Gravação de lembretes e personalização de alertas.

Designação: Automator Dictate / Dragon NaturallySpeaking

Sistema Operativo: Mac OS / Windows

Empresa: Nuance

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.nuance.com/dragon/index.htm

Descrição: Gravação de lembretes e personalização de alertas.

Designação: Adobe Premiere Pro

Sistema Operativo: Windows

Empresa: Adobe

Tipo de Solução: Comercial

URL: https://www.premiumbeat.com/blog/speech-analysis-premiere-pro/

Descrição: Software de edição e análise de vídeo. Permite localizar linhas de diálogo ou assuntos.

Designação: Windows built-in speech recognition

Sistema Operativo: Windows

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.microsoft.com/enable/products/windowsvista/speech.aspx

Descrição: Permite o controlo do computador através de comandos de voz.

2) Soluções Móveis

Designação: Google Now

Sistema Operativo: Android

Empresa: Google

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://www.google.com/landing/now/

Descrição: Assistente pessoal virtual criado pelo Google para a plataforma Android. O serviço tem como função organizar a rotina do utilizador, mostrando-lhe previsões do tempo, trânsito e notícias de acordo com cada perfil. O serviço Now também responde a perguntas feitas pelo utilizador, e suporta a realização de ações no *smartphone* solicitadas através de comandos de voz.

Designação: S-voice

Sistema Operativo: Android

Empresa: Samsung

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.samsung.com/global/galaxys3/svoice.html

Descrição: Assistente pessoal virtual criado pela Samsung para a plataforma Android, e com suporte para despoletar ações através de comandos de voz.

Designação: Voice Search Advanced

Sistema Operativo: Android

Empresa: Team2E

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://play.google.com/store/apps/details?id=ru.yvs

Descrição: Permite efetuar pesquisas em portais, como o youtube e o google, através de comandos de voz.

Designação: Siri

Sistema Operativo: iOS

Empresa: Apple

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://www.apple.com/br/ios/siri/

Descrição: Assistente pessoal virtual para dispositivos com o sistema operativo iOS.

Designação: Microsoft Cortana

Sistema Operativo: Windows Phone

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://www.microsoft.com/en-us/windows/cortana

Descrição: Assistente pessoal virtual para dispositivos com o sistema operativo Windows Phone.

Designação: Lyra

Sistema Operativo: Android / iOS

Empresa: Artificial Solutions

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://portal.heylyra.com/

Descrição: Assistente pessoal virtual multiplataforma.

Designação: Assistant

Sistema Operativo: Android / iOS / Windows Phone

Empresa: Speaktoit

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://www.microsoft.com/en-us/store/p/speaktoit-assistant/9wzdncrdg01v

Descrição: Assistente pessoal virtual multiplataforma.

3) Bibliotecas (APIs)

Designação: Speechmatics

Empresa: Speechmatics

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://www.speechmatics.com/

Designação: Speech Recognizer Plugin

Empresa: Red Shift

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://redshiftcompany.com/products/speech-recognizer-plugin/

Designação: Textshark

Empresa: Textshark

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.textshark.com/

Designação: Google Speech To Text API

Empresa: Google

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://gist.github.com/alotaiba/1730160

Designação: IBM® Speech to Text

Empresa: IBM

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://console.bluemix.net/docs/services/speech-to-text/index.html

Designação: AT&T Speech API

Empresa: AT&T

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://developer.att.com/apis/speech

Designação: Is Speech

Empresa: Is Speech

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://www.ispeech.org/developers

Designação: Microsoft Speech API (SAPI)

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://msdn.microsoft.com/en-us/library/ms720151%28v=vs.85%29.aspx

Designação: My Caption

Empresa: My Caption

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://www.mycaption.com/

Designação: Microsoft Project Oxford

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://www.projectoxford.ai/

Designação: SpeechRecognizer

Empresa: Google

Tipo de Solução: *Open Source*

URL: https://developer.android.com/reference/android/speech/SpeechRecognizer.html

Designação: Bing Voice Recognition API

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: http://datamarket.azure.com/dataset/bing/speechrecognition

Designação: CeedVocal SDK

Empresa: CeedVocal

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://creaceed.com/ceedvocal

Designação: Dragon Software Developer Kits

Empresa: Nuance

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: www.nuance.com/for-developers/dragon/index.htm

Designação: Natural Language for Developers

Empresa: Wit.ai

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://wit.ai/

Designação: Speech Technologies

Empresa: Microsoft

Estado: Em desenvolvimento

Tipo de Solução: Comercial

URL: https://msdn.microsoft.com/en-us/library/hh323806.aspx

Designação: CMU Sphinx

Empresa: Carnegie Mellon University

Estado: Em desenvolvimento

Tipo de Solução: *Open Source*

URL: https://cmusphinx.github.io/

Anexo 2: Reconhecimento de fala - soluções

Durante o levantamento do estado da arte foram identificadas as seguintes soluções para o reconhecimento de fala:

Designação: Camfind
Plataforma: Móvel (Android e iOS) e API HTTP
Empresa: Camfind
Estado: Disponível
Tipo de Solução: Comercial
URL: http://camfindapp.com
Descrição: Aplicação móvel de reconhecimento de objetos em imagens, que posteriormente foi disponibilizada como uma API baseada em serviços cloud.

Designação: Vufind Recognize
Plataforma: API HTTP
Empresa: DeepVu
Estado: Indisponível
Tipo de Solução: Comercial
URL: https://web.archive.org/web/20161017015928/http://vufind.com
Descrição: Serviço de reconhecimento de imagem em tempo real para a classificação de fotos e vídeos. Criação de meta dados com fins comerciais e publicidade. O serviço foi descontinuado em 2016.

Designação: Catchoom
Plataforma: API HTTP
Empresa: Catchoom
Estado: Disponível
Tipo de Solução: Comercial
URL: http://catchoom.com
Descrição: Reconhecimento de objetos e facilidade na implementação de realidade aumentada em dispositivos móveis. Disponibilização de um plano gratuito para testes.

Designação: 6px.io

Plataforma: API HTTP

Empresa: 6px

Estado: Indisponível

Tipo de Solução: Comercial

URL: https://web.archive.org/web/20160506082602/http://www.6px.io

Descrição: Serviço para processamento e análise de imagens em grande escala baseada em Cloud. O serviço foi descontinuado em 2016.

Designação: Fashionbase

Plataforma: API HTTP

Empresa: Fashionbase

Estado: Disponível

Tipo de Solução: Comercial

URL: https://www.fashionbase.com/developers

Descrição: Serviço de reconhecimento de imagens especializado em marcas de roupa.

Designação: Moodstocks

Plataforma: API HTTP

Empresa: Moodstocks

Estado: Indisponível

Tipo de Solução: Comercial

URL: https://web.archive.org/web/20160201222457/https://moodstocks.com

Descrição: Serviço de reconhecimento de imagens de fácil utilização. Permitia um número de reconhecimentos ilimitado com o pagamento de um valor mensal. Possuía plano gratuito para teste.

O serviço foi descontinuado em 2016, pois foi adquirido pela Google.

Designação: Kooaba

Plataforma: API HTTP

Empresa: Vuforia

Estado: Disponível

Tipo de Solução: Comercial

URL: http://developer.vuforia.com

Descrição: Serviço de reconhecimento de imagem com o foco na realidade aumentada. Apresenta um plano de experimentação com um número limite de solicitações.

Designação: Clarifai

Plataforma: API HTTP

Empresa: Clarifai

Estado: Disponível

Tipo de Solução: Comercial

URL: http://www.clarifai.com

Descrição: Serviço de reconhecimento de imagem e vídeo. Versão de avaliação disponibilizada no portal da empresa.

Designação: MobileEngine

Plataforma: API HTTP

Empresa: TinEye

Estado: Disponível

Tipo de Solução: Comercial

URL: https://services.tineye.com/MobileEngine

Descrição: Serviço de reconhecimento de imagem auto escalável para aplicações móveis.

Designação: CloudSight

Plataforma: API HTTP

Empresa: CloudSight

Estado: Disponível

Tipo de Solução: Comercial

URL: http://cloudsight.ai

Descrição: Serviço baseado em Cloud que assegura uma grande taxa de acerto no reconhecimento e interpretação de imagens. Permite a realização de testes.

Designação: Orbeus ReKognition API

Plataforma: API HTTP

Empresa: Orbeus

Estado: Indisponível

Tipo de Solução: Comercial

URL: https://web.archive.org/web/20160208004620/https://rekognition.com

Descrição: Serviço que facilita a adição de análise de imagens às aplicações, permitindo a deteção de objetos, cenas e faces humanas, o reconhecimento de celebridades, além de identificar conteúdo inadequado nas imagens. O serviço foi descontinuado em 2016, pois foi adquirido pela Amazon que o passou a disponibilizar em https://aws.amazon.com/pt/rekognition/ .

Designação: OpenCV (*Open Source Computer Vision Library*)

Plataforma: *Desktop*, *Mobile*, *Web*

Empresa: OpenCV

Estado: Disponível

Tipo de Solução: *Open Source*

URL: http://opencv.org

Descrição: Biblioteca *open source* para visão a área de visão computacional. Projetada para diferentes plataformas, e com interfaces para várias linguagens de programação (*C*, *C++*, *Python* e *Java*). Encontram-se implementados vários algoritmos para o tratamento e o reconhecimento

de imagens e de vídeos em tempo real. O OpenCV possui uma grande comunidade de utilizadores.

Designação: Pastec

Plataforma: API HTTP

Empresa: VisuaLink

Estado: Disponível

Tipo de Solução: *Open Source*

URL: http://pastec.io

Descrição: Plataforma *open source* de indexação e de pesquisa para o reconhecimento de imagens, e que utiliza API OpenCV. Pode efetuar o reconhecimento de objetos planos (como capas de livros), etiquetas, fotografias e imagens . Não efetua o reconhecimento de faces, objetos 3D, códigos de barras nem de *QR codes*.