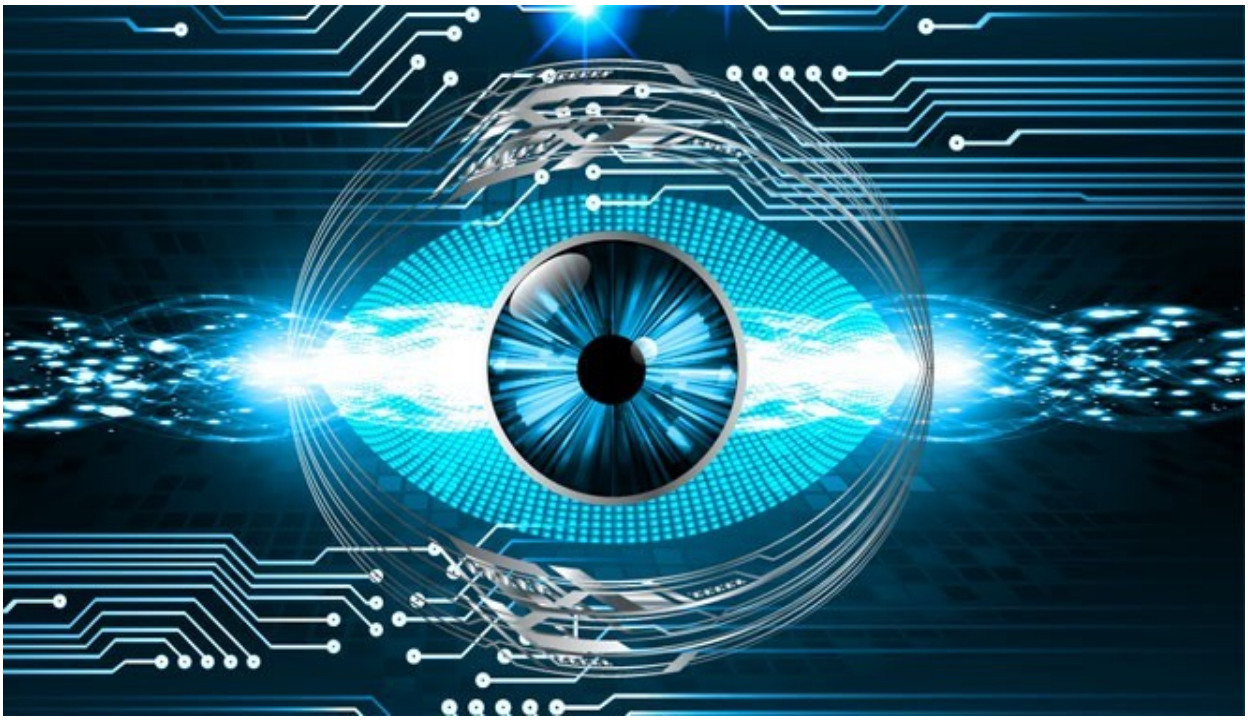
Universidade Fernando Pessoa

06/2020

Deteção de eletrodomésticos através da utilização de Visão Computacional



Luís Aguiar | 36452

Disciplina:
Laboratório de Projeto Integrado

Orientadores:
Prof. Pedro Sobral (pmsobral@ufp.edu.pt)
Prof. José Torres (jtorres@ufp.edu.pt)

Docente da disciplina L.P.I.:
Prof. Paulo Rurato (prurato@ufp.edu.pt)

Índice

1. Introdução....3
Enquadramento/aplicação no mundo real....3
Objetivos do projeto....3
2. O que é a Visão Computacional/Deteção de Objetos?....3
Diferença entre classificação de imagens e deteção de objetos....4
3. Implementação (Fases do desenvolvimento)....5
Fase de análise e escolha de uma estrutura de rede neuronal....6
Fase de recolha do *Dataset*....8
Fase de treino da rede neuronal utilizando o *Dataset*....9
Fase de testes com exemplos reais....12
4. Avaliação do modelo....14
5. Conclusão....16

1. Introdução

Enquadramento/aplicação no mundo real

O projeto desenvolvido dedica-se à entrega de informações relativas aos eletrodomésticos com os quais nos deparamos no nosso dia-a-dia, a pessoas com dificuldades cognitivas, encaixando-se então no Ambient Assisted Living (AAL), um tópico em exponencial crescimento.

Mais especificamente, foca-se em, **através da implementação de Visão Computacional, distinguir uma vasta categoria de eletrodomésticos** e recorrer a uma base de dados que contém as instruções de utilização referentes a cada um deles, disponibilizando-as para o utilizador, desta forma, tornando possível, de uma forma fácil e intuitiva ao cliente manipular os equipamentos em segurança e eficientemente.

Objetivos do projeto

O objetivo final do projeto é possibilitar a um utilizador, com o simples apontar de uma câmara de um telemóvel ou de um computador, obter a informação que precisa, acerca daquele eletrodoméstico que já não utiliza há algum tempo, ou simplesmente por razões de saúde se esqueceu de como proceder para o manusear. Para além deste objetivo mais "palpável", havia também a meta final de aprender novas tecnologias e explorar áreas desconhecidas e que podem ser benéficas no futuro. Tendo ingressado então neste projeto sem qualquer experiência/conhecimento em nenhuma das linguagens e *frameworks* recomendadas pelos orientadores para a sua realização.

2. O que é a Visão Computacional/Deteção de Objetos?

A Visão Computacional é o campo da inteligência artificial responsável pelo treino de computadores com o objetivo de lhes possibilitar a compreensão e interpretação do mundo real. Isto só é possível utilizando *deep learning*, ciência que permite às

máquinas, através do treino com imagens reais aliado a uma rede neuronal, reagirem e tomarem decisões.

A Deteção de Objetos (*Object Detection*) é uma tecnologia relacionada com a Visão Computacional que, tal como o nome indica, se dedica apenas à deteção de objetos em imagens digitais e vídeos. Dentro desta é possível identificar dois domínios de aplicação, a classificação de imagens e a deteção de objetos. Perceber o que estes são e optar por um deles é um passo fulcral em qualquer projeto nesta área.

Figura 1: Diferença entre classificação de imagens (esquerda) e deteção de objetos (direita).

Através da análise da figura 1, chega-se à conclusão que a distinção entre estas duas categorias é bastante simples e intuitiva. Na classificação de imagens, toda a imagem é classificada com uma única etiqueta. Na deteção de objetos, a rede neuronal localiza (potencialmente mais que um) objeto dentro da mesma imagem.

Quando se implementa **classificação de imagens**, ao entregar a imagem à rede neuronal treinada (modelo), esta devolve apenas **uma etiqueta** que indica a classe, podendo também retornar a probabilidade associada a essa análise. Esta classe retornada descreve o conteúdo total da imagem ou, pelo menos, o elemento mais facilmente identificado.

Alguns exemplos de redes neuronais deste tipo são: AlexNet, GoogleNet, SmallVggNet…

Por outro lado, quando se implementa **deteção de objetos**, a nossa rede neuronal treinada identifica especificamente em que parte da imagem estão as classes e devolve as respetivas coordenadas.

Quando se utiliza deteção de objetos, dada uma imagem, a rede neuronal treinada pretende obter:

- Uma **lista de caixas delimitadoras,** que indica as coordenadas para cada objeto identificado na imagem.
- A **classe associada** a cada uma das caixas.
- A **probabilidade associada** à interpretação de cada caixa.

Na figura 1 (direita) é possível verificar que tanto o ser humano como o cão estão identificados com a sua caixa delimitadora, classe e probabilidade associada.

Alguns exemplos de redes neuronais deste tipo são: Single Shot Detector (SSD), You Only Look Once (YOLO)…

3. Implementação (Fases do desenvolvimento)

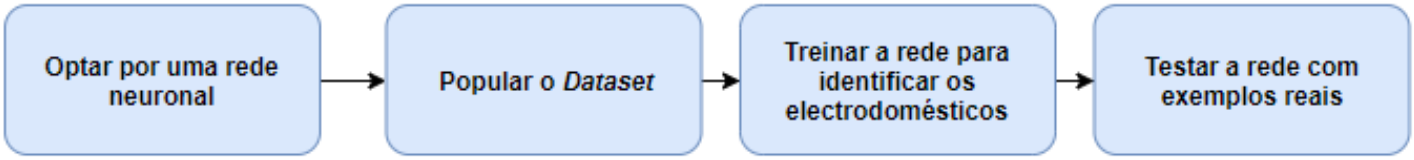


Figura 2: Diagrama *pipeline* das fases de desenvolvimento do projeto.

Depois de investido algum tempo na obtenção de conhecimento nas linguagens e *frameworks* e nos pilares teóricos fundamentais para o desenvolvimento do projeto,

partiu-se então para a aplicação. O **desenvolvimento do projeto foi dividido em 4 grandes fases** (figura 2), sendo estas:

- A **escolha de uma rede neuronal** – análise de várias hipóteses e escolha da rede que melhor se enquadra nas necessidades do projeto e capacidades computacionais disponíveis.
- O **preenchimento do *Dataset*** – obter um extenso conjunto de imagens de qualidade para cada classe.
- O **treino da rede –** elaborar um código que permita, através da utilização da rede neuronal escolhida e das imagens obtidas no passo anterior, treinar a rede neuronal, obtendo um modelo eficaz.
- O **teste com casos reais –** elaborar um código que permita analisar casos do mundo real, de forma a testar a qualidade/assertividade do modelo.

As principais **ferramentas (linguagens e *frameworks)* utilizadas no desenvolvimento do projeto** foram:

- **Python –** linguagem de programação de alto nível, criada em 1991.
- **OpenCV** (*Open Source Computer Vision Library*) - é uma biblioteca multiplataforma para o desenvolvimento de aplicações na área da Visão Computacional.
- **TensorFlow –** é uma biblioteca para criação e treino de redes neuronais, criada em 2015.
- **Keras -** é uma biblioteca de rede neural de código aberto escrita em Python que permite realizar experiências rapidamente com *deep neural networks.*

Fase de análise e escolha de uma estrutura de rede neuronal

Uma rede neuronal consiste numa estrutura de conexão, na qual o processamento se encontra distribuído por um grande número de pequenas unidades (camadas) interligadas.

Nesta primeira fase, o **objetivo era selecionar uma rede neuronal que fosse capaz de cumprir todos os requisitos do projeto**. Depois de muita investigação acerca das dezenas de redes existentes, foi-se filtrando e excluindo algumas delas até sobrarem

apenas 3 (SSD, YOLO, SmallVggNet) que eram as únicas que após a análise pareceram melhor se enquadrar nas necessidades do projeto. Entre estas 3, duas delas eram do tipo deteção de objetos (SSD, YOLO) e a outra era do tipo classificação de imagens (SmallVggNet).

Optou-se então, inicialmente, pela implementação da rede YOLO, mais especificamente a Tiny YOLO, que consegue obter uma taxa de 222 *frames* por segundo durante a classificação de vídeos, que é mais do que o triplo da YOLO normal. Porém, depois de alguns testes com esta, chegou-se à conclusão que **havia dois fatores que incapacitavam a sua aplicação**. O primeiro fator era a falta de *Dataset* de qualidade, uma vez que as redes do tipo deteção de objetos precisam de um *Dataset* "especial" (as imagens de treino têm de ter as caixas delimitadoras). O segundo fator, e mais incapacitante, eram as reduzidas capacidades computacionais computador utilizado, que estimava períodos de mais de 20 horas para realizar um simples processo de treino de um modelo com um conjunto de imagens reduzido.

Depois de todos estes percalços, **optou-se então pela rede SmallVggNet**, que também é uma **excelente opção e cumpre todas as necessidades do projeto.** Esta rede é bastante eficaz e tem a capacidade de analisar um vídeo a uma taxa de 80 *frames* por segundo, o que é mais do que suficiente para o projeto.

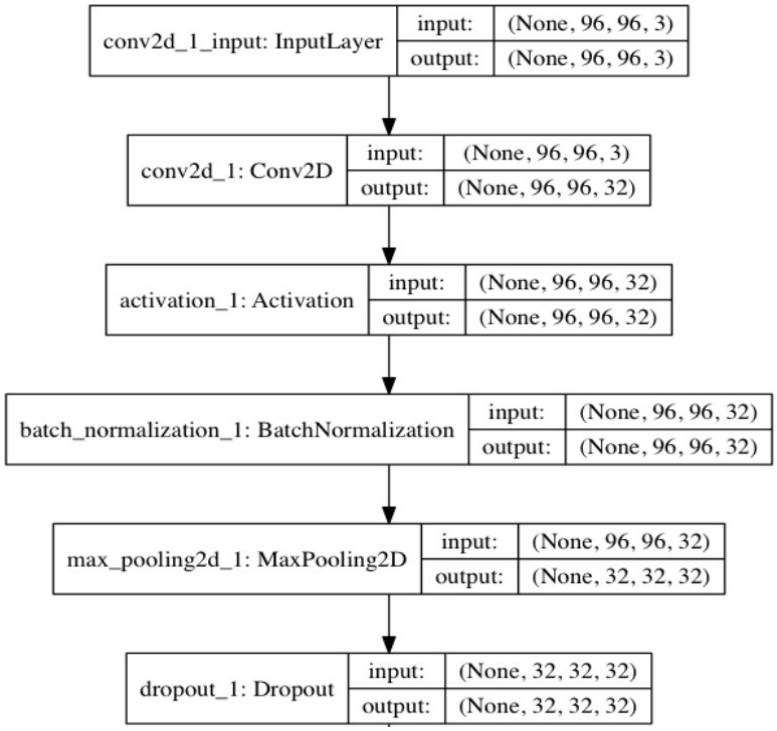


Figura 3: Esquema das camadas iniciais da rede SmallVggNet

Fase de recolha do *Dataset*

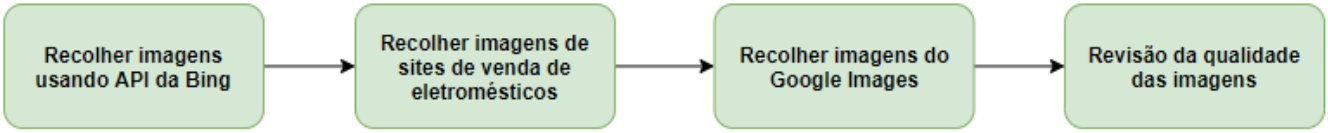


Figura 4: Diagrama *pipeline* da fase de recolha do *Dataset*

Esta fase foi dedicada à aquisição de imagens relevantes e de qualidade.

Para possibilitar o treino da rede para a identificação dos eletrodomésticos **(micro-ondas, frigorífico, fogão, máquina de café, chaleira, torradeira, liquidificador)** teve de se utilizar um conjunto de imagens (*Dataset)* e, tendo em conta que a qualidade deste influência imenso os resultados (quanto melhor e maior este for, melhor será o modelo), é uma etapa que precisa de um cuidado especial .

Para adquirir o *Dataset*, seguiu-se **4 passos** (figura 4):

- **Recolher imagens usando API da Bing** – num primeiro passo, utilizou-se um código simples que recorre à API da Bing e permite obter um conjunto de imagens através da palavra-chave de pesquisa, que neste caso são os nomes das classes (eletrodomésticos).
- **Recolher imagens de sites de venda de eletrodomésticos –** como o passo anterior não retornava imagens de qualidade suficientes para cada classe, recorreu-se aos sites de venda de eletrodomésticos (Worten, Media Markt, Radio Popular…) e através da utilização de um *Web Scraper* (mecanismo que permite extrair dados de sites da web, convertendo-os em informação estruturada) as imagens referentes aos eletrodomésticos foram recolhidas desses sites.
- **Recolher imagens do Google –** mesmo após os dois passos anteriores, ainda não se tinha obtido a quantidade de imagens almejada (1000 imagens para cada classe), então, utilizando o mesmo mecanismo de *Web Scraping* do passo anterior recorreu-se ao *Google Images* para obter mais algumas imagens para cada classe, completando assim o objetivo.
- **Revisão da qualidade das imagens –** neste passo analisou-se todas as 7 mil imagens (1000 imagens e 7 classes) e estas foram filtradas, removendo as imagens em que era difícil identificar a classe e que iriam reduzir a eficiência do modelo, ficou-se assim com cerca de 650 imagens para cada classe.

O *Dataset* recolhido irá ter um papel fundamental no próximo passo (treino do modelo), uma vez que, **todo e qualquer modelo eficiente é originado a partir de um *Dataset* exímio.**

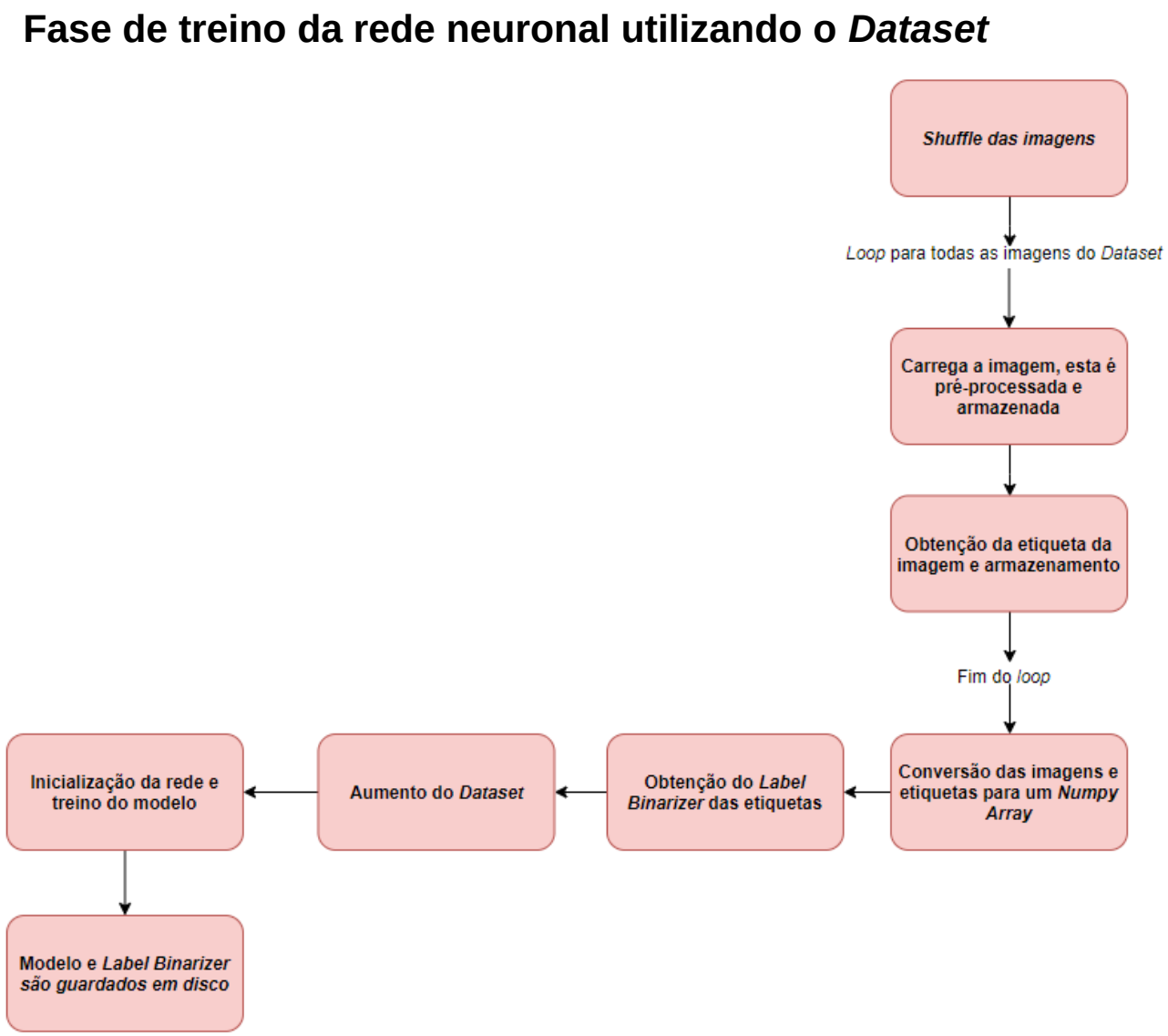


Figura 5: Diagrama *pipeline* da fase de treino da rede

Nesta fase foi elaborado um código em Python que se dedica à criação do modelo, modelo este que é criado através do treino da nossa rede, selecionada anteriormente (SmallVggNet), utilizando as imagens do nosso *Dataset*. A rede vai analisar todas as

imagens e procurar padrões em cada eletrodoméstico que lhe permitam distinguir este dos outros. **Este processo espelha o processo de aprendizagem do ser humano**, uma vez que, o ser humano também identifica padrões nos objetos, que lhe permite distingui-los entre si. Quantos mais objetos este vir, com mais facilidade este os distingue, da mesma forma acontece com o processo de treino de uma rede neuronal para obtenção de um modelo.

Este processo de treino dará origem a um modelo que será capaz de identificar e distinguir os eletrodomésticos. Para concretizar esta tarefa, o código elaborado faz o seguinte (figura 5):

- ***Shuffle* das imagens –** tendo em conta que as imagens do *Dataset,* que o programa lê, são lidas de uma forma sequencial e ordenada, ou seja, todas as imagens de cada classe estão seguidas, o que gera um treino viciado. Para combater este problema, tem de se baralhar as imagens entre si, gerando assim uma sequência aleatória de imagens de diferentes tipos de eletrodomésticos, o que melhora os resultados do modelo.
- **Para cada imagem do *Dataset*:**
  - **Armazenamento da imagem processada –** a imagem é carregada para a memória, processada conforme os requisitos da rede, convertida para um *array* de bits compatível com Keras e armazenada numa lista.
  - **Extração da etiqueta da imagem** – a partir do diretório da imagem (que segue o seguinte padrão "dataset/{ETIQUETA}/{NOME_IMAGEM}.jpg"), é possível retirar a etiqueta para toda e qualquer imagem, etiqueta essa que é armazenada numa lista.
- **Conversão das listas de imagens e labels para um *array*** – a lista de imagens é convertida num *Numpy Array* (*array* que permite fazer cálculos numéricos) e os bits de todas as imagens são normalizados (passam de 0 → 255 para 0 → 1). A lista de etiquetas (*labels*) também é convertida num *Numpy Array.* Com estas alterações fica mais fácil a manipulação destas estruturas em processos futuros.
- **Conversão do *array* de etiquetas em *Label Binarizer*** – o *array* de etiquetas obtido anteriormente é convertido para *Label Binarizer.* O *LB* atribui a cada uma das classes (eletrodomésticos) uma ligação com um valor inteiro (cada valor identifica uma classe), desta forma, tornando possível converter o nome da classe num inteiro e vice-versa. Esta funcionalidade é essencial, uma vez que ao analisar

uma imagem do mundo real, o modelo apenas retorna um inteiro e, recorrendo ao *LB (Label Binarizer),* é possível obter o nome da classe através desse inteiro.

- **Aumento do *Dataset* –** uma vez que se está a trabalhar com um conjunto de imagens relativamente reduzido, recorreu-se ao mecanismo de *data augmentation* durante o processo de treino, fornecendo assim mais imagens (baseadas nas já existentes) para o treino do modelo. O processo de *data augmentation* gera novas imagens a partir das já existentes (figura 6), através de modificações na rotação, ampliação...

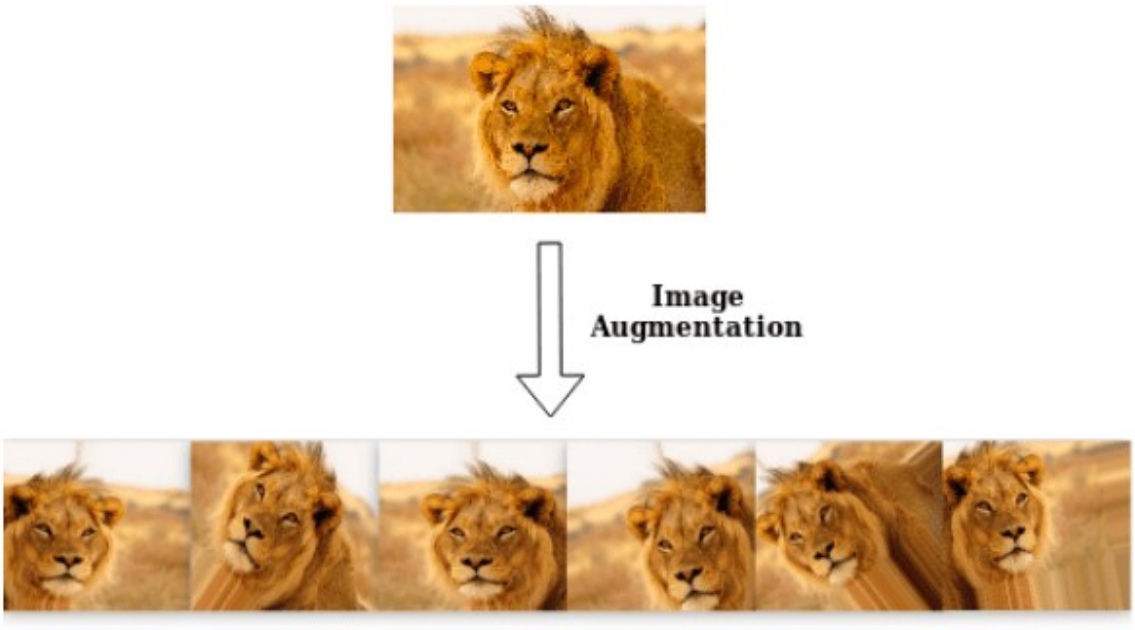


Figura 6: Exemplo ilustrativo do processo de *data augmentation.*

- **Inicialização da rede e treino do modelo –** depois de finalizar todas as preparações para tornar possível a criação de um modelo eficiente, inicia-se então a rede neuronal (SmallVggNet) e faz-se o treino utilizando a rede neuronal aliada às imagens processadas para este fim.
- **O modelo e o *Label Binarizer* são guardados em disco –** após a finalização do treino, obtém-se um modelo treinado para a identificação dos eletrodomésticos, esse é guardado em disco, da mesma forma que o *LB* para serem utilizados, à posteriori, na análise de imagens reais.

No fim deste processo, é obtido um modelo treinado, capaz de identificar os diferentes tipos de eletrodomésticos. Agora é necessário testar este modelo para perceber se realmente tem capacidades para o fazer com exatidão, testes esses que vão ser feitos na próxima fase.

Fase de testes com exemplos reais

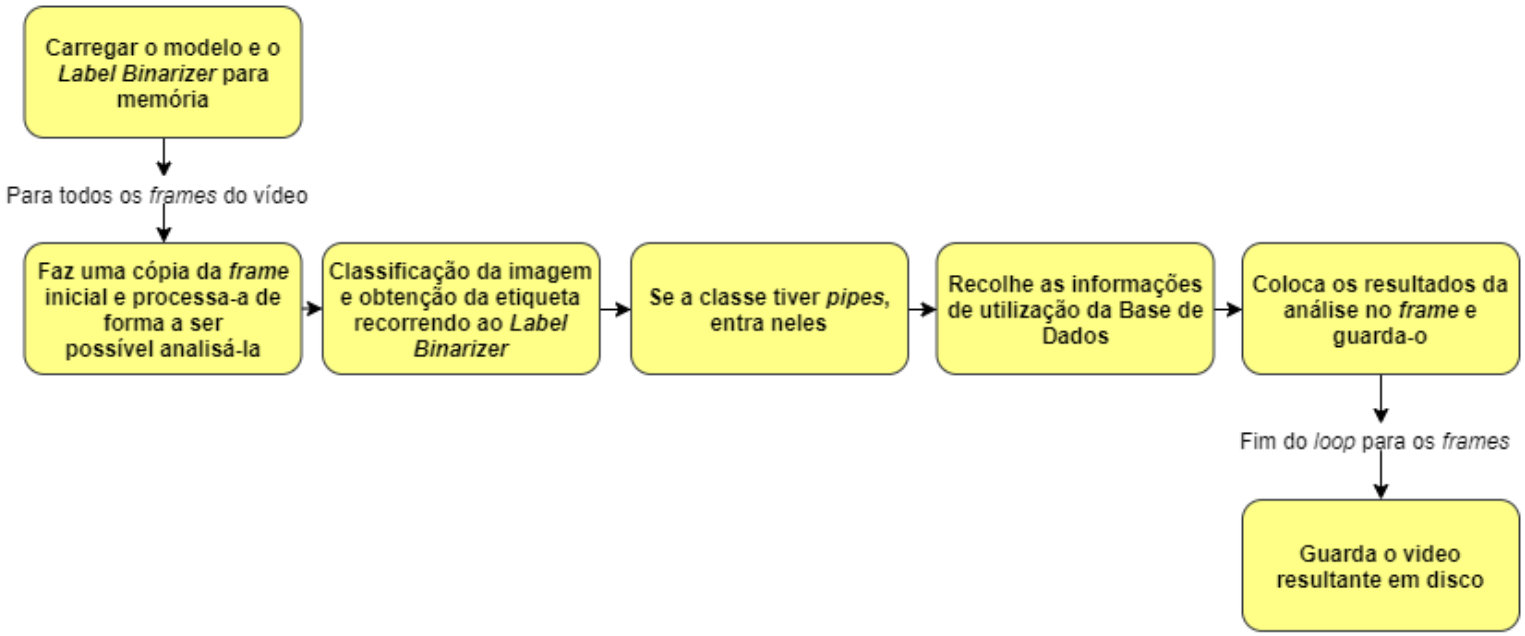


Figura 7: Diagrama *pipeline* da fase de testes com exemplos reais

Esta fase foi dedicada à criação de um código em Python com a capacidade de, em cada *frame* de um vídeo, identificar qual é o eletrodoméstico presente.

Primeiro, de forma a simplificar o processo, **foi realizado um código que apenas conseguia classificar uma imagem.** Depois de atingir essa meta com sucesso e recorrendo ao código para testar alguns casos reais, chegou-se à conclusão que o treino anterior tinha sido bem sucedido e que se tinha conseguido obter um modelo de qualidade que oferece bons resultados.

Tendo obtido estes resultados favoráveis por parte do modelo, decidiu-se aumentar a complexidade da classificação, **aplicando *pipes* às classes máquina de café e fogão**, tornando possível **distinguir um fogão a gás de um elétrico e uma máquina de café**

de cápsulas de uma manual. Estes *pipes* são sub-modelos treinados para distinguir apenas entre estas duas opções e são aplicados da seguinte forma: primeiro a imagem é analisada pelo modelo principal, obtendo-se a classificação, e se este eletrodoméstico possuir um *pipe*, a imagem é analisada pelo respetivo sub-modelo e é obtida uma classificação mais precisa.

Depois, partiu-se então para a **elaboração de um código que fosse capaz de analisar um vídeo e guardar em disco uma cópia deste, devidamente classificado**. Para concluir esse objetivo, utilizou-se o código de classificação de imagens, adaptando-o para vídeo, ficando com a seguinte estrutura (figura 7):

- **Carregar o modelo e o *Label Binarizer* para memória** - inicialmente, para ser possível ter acesso aos recursos necessários à classificação dos *frames* do vídeo, são carregados para memória o modelo e o *Label Binarizer.* Estes recursos são essenciais para as próximas etapas do código.
- **Para todos os *frames* do vídeo:**
  - **É feita uma cópia do *frame* e esta é processada** - é feita uma cópia do *frame*, cópia esta em que se vai aplicar a etiqueta (nome do eletrodoméstico) e respetivas anotações para depois a guardar no vídeo classificado. O *frame* é processado, tal como foi feito na fase anterior, para desta forma ser possível classificá-lo.
  - **Classificação e obtenção da classe através do *Label Binarizer* –** utilizando o modelo carregado em memória anteriormente, procedemos à classificação do *frame*, obtendo um inteiro retornado pela análise. Esse inteiro é descodificado pelo *LB* e é obtido o nome do eletrodoméstico presente no *frame*.
  - **Se a classe tiver *pipes,* entra neles –** através da classificação anterior é possível determinar se a classe obtida possui *pipes* ou não, e se possuir, a *frame* é analisada pelo respetivo sub-modelo obtendo-se um resultado mais minucioso. Caso contrário, fica com a classificação inicial.
  - **Recolher informações de utilização da Base de Dados –** sabendo então qual é o tipo de eletrodoméstico presente no *frame,* é possível utilizar essa classificação para obter as informações de utilização deste eletrodoméstico, recorrendo à base de dados.
  - **Colocação das anotações no *frame* e armazenamento deste –** a identificação do eletrodoméstico e as respetivas indicações de utilização são escritas na cópia realizada inicialmente e este *frame* devidamente classificado é armazenado.
  - **Armazenamento do vídeo resultante em disco –** depois de classificar todos os *frames* e os guardar sequencialmente, obtemos então um vídeo em que todas

as suas *frames* estão claramente classificadas. Este vídeo é armazenado em disco.

Esta fase, para além de permitir a classificação de imagens, demonstra também a qualidade do modelo e sub-modelos que obtemos com o treino, uma vez que ao aplicá-los em casos reais, os resultados são bastante assertivos e satisfatórios.

4. Avaliação do modelo

Após a obtenção de um modelo treinado e realização de um código que permite testá-lo, **é importante estudar a assertividade obtida** com vídeos reais, e se esta não for a desejada, procurar melhorá-la. Para este estudo, foram filmados vários vídeos dentro de casa, nomeadamente na cozinha, em que são capturados diversos eletrodomésticos em diferentes perspetivas, tendo assim um reportório de testes variado e de qualidade.

Esses vídeos foram então analisados pelo modelo e os resultados corresponderam ao que era desejado. A taxa de erro era bastante reduzida, sendo que quando esta "confusão" entre eletrodomésticos acontecia era devido principalmente à poluição da imagem (muitos objetos na mesma imagem). **As imagens seguintes ilustram os testes realizados:**

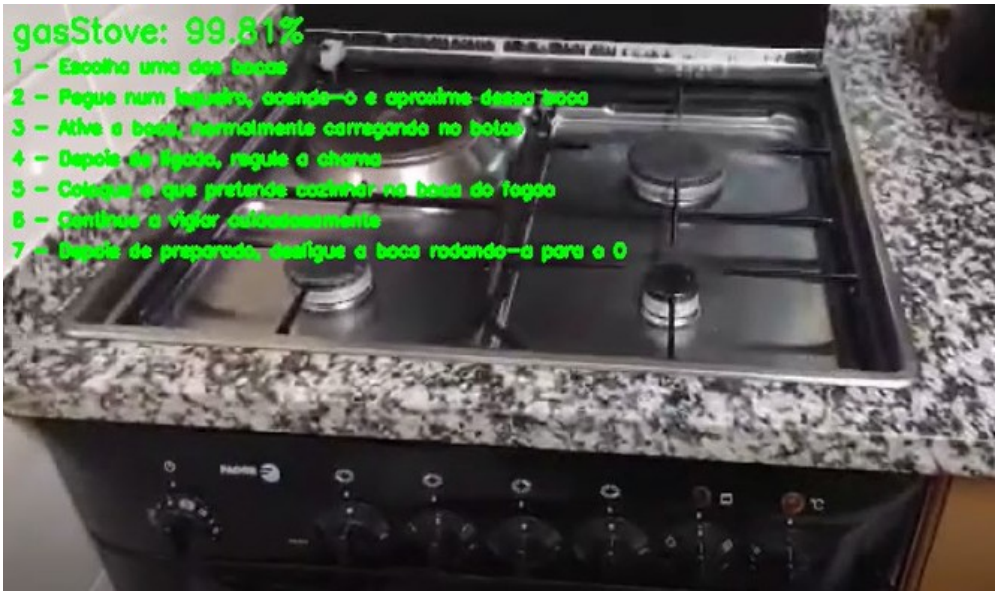


Figura 8: Classificação de um fogão a gás.

- **Classificação de um fogão a gás –** Na figura 8 está ilustrado um *frame* de vídeo, no qual é identificado um fogão a gás. É possível observar que a identificação está correta e que a percentagem de certeza é extremamente elevada, esta percentagem caracteriza a "confiança" que o modelo tem que esta imagem é referente à respetiva classe (99,81%).

Figura 9: Classificação de uma máquina de café de cápsulas.

- **Classificação de uma máquina de café de cápsulas –** Na figura 9 está ilustrado um *frame* de vídeo, no qual é identificada uma máquina de café de cápsulas. Ao observar a figura, é possível verificar que a identificação está correta e que a percentagem desta certeza é bastante elevada (98,18%).

À semelhança do que aconteceu com todos os exemplos de teste, em ambas as classificações ilustradas é possível notar que os resultados são bastantes satisfatórios, concluindo que o modelo tem a capacidade de realizar classificações de confiança, objetivo principal do projeto em questão desde início. **Tendo em conta que o objetivo do projeto foi concretizado, podemos dar este como concluído.**

5. Conclusão

Neste projeto foi abordado o tema *Ambient Assisted Living*, tendo recorrido às tecnologias da Visão Computacional, mais precisamente da Deteção de Objetos, para o desenvolver e conseguir obter resultados relevantes. Com o desenvolvimento deste, conclui-se a importância desta tecnologia, em exponencial crescimento nos dias de hoje, e a elevada e variada quantidade de aplicações que esta pode ter.

Este projeto foi essencial para o aprofundamento do conhecimento nesta área, permitindo a compreensão de como funcionam os processos de:

- recolha de um *Dataset* de qualidade;
- escolha de uma rede neuronal;
- obtenção de um modelo através do treino de uma rede neuronal aliada ao *Dataset* recolhido;
- teste da assertividade desse novo modelo e estratégias de melhoramento.

Tendo em conta que estas são as fases essenciais para todo e qualquer projeto na área da visão computacional, com a sua realização, foi possível ficar apto para abraçar novas experiências e projetos nesta área.

Apesar de ter aprendido as bases, ainda muito ficou por esmiuçar, tendo então ficado com uma “fome de conhecimento” nesta área que se procurará satisfazer num futuro próximo.

Planeamento

Luís Aguiar | 36452

Tarefa	Horas	Início	Fim
• Definição dos requisitos e objetivos do projeto	6	25/02/2020	28/02/2020
• Estudar a linguagem Python	5	30/02/2020	12/03/2020
• Aprender a manipular a *framework* OpenCV	4	13/03/2020	20/03/2020
• Estudar e compreender as frameworks Tensorflow e Keras	9	21/03/2020	10/04/2020
• Definição de um *pipeline* de abordagem do projeto	5	10/04/2020	19/04/2020
• Escolha de uma rede neuronal para aplicar no projeto	5	20/04/2020	25/04/2020
• População do *Dataset*	9	26/04/2020	11/05/2020
• Elaboração de um código de treino do modelo	11	12/05/2020	28/05/2020
• Elaboração de um código de testes do modelo	7	29/05/2020	02/06/2020
• Realização de testes do modelo com exemplos reais	6	02/06/2020	07/06/2020
• Preparação do *PowerPoint* para apresentação	4	10/06/2020	12/06/2020
• Elaboração do relatório de projeto	12	13/06/2020	22/06/2020

Infelizmente, devido a todos os contratempos que aconteceram durante a realização do projeto, o planeamento não foi seguido com o devido rigor, de qualquer das maneiras, este foi seguido ao máximo, dentro das possibilidades.

LOGBOOK

Luís Aguiar | 36452

Semanas iniciais:

- Definição do objetivo e requisitos do projeto
- Elaboração de um planeamento de desenvolvimento do projeto
- Estudo dos pilares teóricos e das ferramentas necessárias para a realização do projeto

Semanas 30/03 → 05/04 e 06/04 → 12/04:

- Realização de uma versão de prototipagem usando uma pequena biblioteca de eletrodomésticos. desta forma ajudando-nos a perceber o funcionamento das bibliotecas e APIs que estamos a utilizar, ficando assim mais elucidados para o que temos de fazer na proposta final.
- Reunião com os orientadores

Semana 13/04 → 19/04:

- Comparação de rendimento de diferentes redes neuronais
- Implementação e análise do *dataset* MSCOCO
- Implementação do modelo neuronal YOLO

Semana 20/04 → 26/04:

- Optar entre redes “object detector” e “object identifier”
- Estudo aprofundado da rede SmallVGGNet
- Reunião com os orientadores

Semana 27/04 → 03/05:

- Implementação da rede SmallVGGNet para análise de imagens
- Elaboração de comentários aprofundados do código, de forma a facilitar a interpretação
- Reunião com os orientadores

Semanas 04/05 → 10/05 e 11/05 → 17/05:

- Implementação da análise de vídeo utilizando o modelo já treinado
- Adicionado um *pipe* para um sub-modelo que distingue maquinas de café de cápsulas de maquinas de café manuais
- Utilização dessa nova funcionalidade para analise de um vídeo
- Deteção de alguns problemas na rede neuronal e exploração de possíveis resoluções
- Reunião com os orientadores

Semana 18/05 → 24/05:

- Melhoramentos ao nível da qualidade e dimensão do *Dataset*
- Treino da rede tirando partido destes melhoramentos

Semana 25/05 → 31/05:

- Adicionado um *pipe* para um sub-modelo que distingue fogões a gás de fogões elétricos
- Reunião com os orientadores

Semana 01/06 → 07/06:

- Testes do nosso modelo com casos reais
- Reunião com os orientadores

Semana 08/06 → 14/06:

- Inicio da elaboração do relatório de projeto
- Mais testes do modelo com casos reais
- Reunião com os orientadores

Semana 15/06 → 21/06:

- Elaboração do *powerpoint* para apresentação do projeto
- Continuação da elaboração do relatório

Semana 22/06 → 28/06:

- Conclusão do relatório
- Entrega dos documentos de projeto (Planeamento, Logbook e relatório)