Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

ProShot - assistente pessoal de fotografia

Pedro Miguel da Luz Cabrita de Sousa Brosque

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática

Orientador

Doutor Alexandre Manuel de Castro Passos de Almeida, Professor Auxiliar

ISCTE-IUL

Co-Orientador

Doutor João Pedro Afonso Oliveira da Silva, Professor Auxiliar

ISCTE-IUL

Outubro, 2018

Resumo

Neste trabalho é proposta a criação de uma aplicação Android que sugere ao utilizador como melhorar a captura da fotografia, especificamente retratos de pessoas em tempo real, tendo por base a iluminação, o tipo de plano fotográfico presente e as características específicas para cada um, tal como o enquadramento.

De forma a determinar qual o tipo de plano presente, é utilizada uma rede neuronal convolucional (CNN), sendo que para tal foram efetuados testes com várias redes diferentes e feita uma comparação para determinar que arquitetura se adequa melhor ao problema. A rede final atinge uma precisão de 99%, utilizando uma técnica de *transfer learning*. Estes resultados foram obtidos num conjunto de imagens recolhidas e classificadas manualmente segundo cada tipo de plano fotográfico, tendo sido usado parte deste conjunto de dados no treino das próprias redes.

Para determinar o enquadramento fotográfico, é proposto um método que utiliza um algoritmo de deteção facial seguido de um algoritmo de deteção de olhos que, com base na regra dos terços dá indicações ao utilizador sobre como corrigir o enquadramento. Foram comparados vários algoritmos de deteção facial tanto ao nível da eficácia de deteção como do tempo de processamento, onde a solução final assegura o equilíbrio entre os dois atingindo uma taxa de deteção de 91%.

Foi também analisada a posição dos olhos num conjunto de imagens consideradas como tendo um bom enquadramento, as quais serviram para determinar um valor de tolerância que serviu como complemento para a regra dos terços.

Palavras-chave: Redes Neuronais Convolucionais; Processamento de Imagem; Deteção Facial; Deteção de Olhos; Fotografia; Deteção em Tempo Real; Android.

Abstract

In this work we propose the creation of an Android application that gives suggestions to the user on how to improve the capture of photography, specifically portraits in real time, based on lighting, the present type of photographic shot and their specific characteristics, such as framing.

In order to determine what's the present type of photographic shot, a convolutional neural network (CNN) is used. For this, tests were performed with several different networks and a comparison was made to determine which architecture best fits the problem. The final network obtains an accuracy of 99% using a transfer learning technique. These results were obtained on a data set of images, manually collected and classified according to each type of photographic shot, where part of this data set was also used in the training of the convolutional neural networks.

To determine the photographic framing, we propose a method that uses a facial detection algorithm followed by an eye detection algorithm which, based on the rule of thirds, gives the user instructions on how to correct the framing. Several facial detection algorithms were compared in terms of detection effectiveness as well as processing time, where the final solution ensures a balance between the two reaching 91% of accuracy.

The position of the eyes on a set of images considered as having a good framing was also analyzed, which helped to determine a tolerance value that served as a complement to the rule of thirds.

Keywords: Convolutional Neural Networks; Image Processing; Facial Detection; Eye Detection; Photography; Real-Time Detection; Android.

Agradecimentos

Esta dissertação contou com o apoio de várias pessoas, sem as quais nada disto seria possível.

Gostaria de agradecer aos meus orientadores Prof. Alexandre Almeida e Prof. João Oliveira pela orientação dada nesta dissertação.

Agradeço também o apoio da NVIDIA Corporation com a doação da GPU Titan Xp usada para esta pesquisa.

Ao Prof. Pedro Faria Lopes, agradeço a ajuda dispensada que contribuiu para uma melhor qualidade no trabalho realizado.

Ao Prof. Pedro Sebastião, agradeço profundamente as várias oportunidades que me facultou, as quais me ajudaram a desenvolver competências a nível profissional e a crescer como pessoa.

Um grande obrigado aos meus amigos, pelo enorme apoio e por me ajudarem a manter focado nos objetivos.

À minha família, agradeço toda a paciência, força e motivação dadas ao longo de todo este percurso, as quais foram fundamentais.

Por fim, ao longo do percurso do meu mestrado conheci muitas pessoas que me fizeram evoluir tanto a nível pessoal como profissional, a todas elas um grande obrigado por me ajudarem a tornar na pessoa que sou hoje.

Acknowledgements

This dissertation had the support of several people, without them none of this would be possible.

I would like to acknowledge my supervisors Prof. Alexandre Almeida and Prof. João Oliveira for the given guidance during this dissertation.

I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

To Prof. Pedro Faria Lopes, I appreciate the help given, which has contributed to a better quality of this work.

To Prof. Pedro Sebastião I'm deeply grateful for the various given opportunities, which helped me develop professional skills and grow as a person.

To all my friends, thank you for the immense support given and for always keeping me focused on my objectives.

To my family, I profoundly apreciate for all the patience, strength and motivation you have given me throughout this course, which were fundamental.

Finally, during the course of my master's degree I met many people who made me evolve both as a person and professionally, to all of them a big thank you for helping me become the person that I am today.

Conteúdo

Resumo **iii**

Abstract **v**

Agradecimentos **vii**

Acknowledgements **viii**

Lista de Figuras **xi**

Lista de Tabelas **xiv**

Abreviaturas **xv**

1 Introdução **1**
1.1 Enquadramento 1
1.2 Questões e objetivos de investigação 1
1.3 Estrutura e organização da dissertação 3

2 Trabalho relacionado **5**
2.1 Algoritmos para deteção facial 5
2.1.1 Haar Cascades 5
2.1.2 Histogram of Oriented Gradients – HOG 9
2.1.3 LBP Cascades 11
2.1.3.1 Representação facial baseada no LBP 12
2.2 Técnicas de Deep Learning usadas para a classificação dos planos fotográficos 13
2.2.1 Deep Learning 13
2.2.1.1 Overfitting 14
2.2.1.2 Data augmentation 15
2.2.1.3 Dropout 16
2.2.1.4 Transfer-Learning 16
2.3 Aplicações Semelhantes 18

3 Data Set **19**
3.1 Critérios para a definição dos planos fotográficos 19
3.2 Data set utilizado 21

3.3 Análise e Processamento do Data Set 22

4 Método para sugestões fotográficas **27**
4.1 Processamento de imagem 27
4.2 Método proposto para a melhoria do enquadramento fotográfico . . 28
4.2.1 Enquadramento fotográfico 28
4.2.2 Algoritmo Haar-Cascade para detectar olhos 29
4.3 Método para sugestões relativas à iluminação 30
4.3.1 Iluminação 31
4.3.1.1 Iluminância 32
4.3.1.2 Luminância 32
4.3.2 Características dos smartphones 33
4.3.2.1 Sensor de Luz 33
4.3.2.2 Controlo da câmara 34

5 Redes neuronais **35**
5.1 LeNet 35
5.2 VGG16 36
5.3 Modificação da rede VGG16 37
5.4 Treino das Redes Neuronais 38
5.4.1 Discussão dos resultados 46
5.5 Comparação das redes 47
5.5.1 Testes Reais 48
5.5.1.1 Resultados obtidos com Imagens Verticais 49
5.5.1.2 Percentagem de confiança com Imagens Verticais . 53
5.5.1.3 Imagens Horizontais 57
5.6 Discussão 57

6 Algoritmos de deteção facial e deteção de olhos **59**
6.1 Resultados da deteção facial 59
6.2 Resultados do algoritmo de deteção de olhos 62
6.3 Testes Reais na aplicação Android 67
6.4 Discussão 69

7 Conclusões **71**
7.1 Trabalho Futuro 72

Anexos **75**

A Planos Fotográficos **75**

Bibliografia **77**

Lista de Figuras

2.1 Exemplo de características Haar 6
2.2 Imagem integral 7
2.3 Exemplo de duas características Haar 7
2.4 Exemplo de Classificador em Cascata 8
2.5 Representação de uma característica HOG 9
2.6 Representação da estrutura de uma cara numa imagem HOG 10
2.7 Cálculo do LBP 11
2.8 Primitivas locais ou Padrões uniformes 11
2.9 Representação facial baseada no operador LBP 13
2.10 Exemplo da operação de MaxPooling 15
2.11 Exemplo de transformações de data augmentation 16

3.1 Ilustração do plano CloseUp 20
3.2 Ilustração do plano Big CloseUp 20
3.3 Ilustração do plano Médio 20
3.4 Histograma dos valores do aspect ratio das imagens do Plano não . 23
3.5 Histograma dos valores do aspect ratio das imagens do Plano Médio 24
3.6 Histograma dos valores do aspect ratio das imagens do Plano Close-Up 25
3.7 Exemplo de uma imagem do plano Close-Up 26
3.8 Exemplo de uma imagem do plano Big Close-Up 26

4.1 Pontos de ouro 29
4.2 Esboço Indicações sobre enquadramento com base na posição dos olhos e dos pontos de ouro 30

5.1 Arquitectura da rede LeNet 36
5.2 Gradient descent com um valor de *learning rate* grande (direita) e pequeno (esquerda) 40
5.3 Evolução do valor de precisão obtida pela rede LeNet no treino com imagens Verticais utilizando o optimizador ADAM 42
5.4 Evolução do valor de precisão obtida pela rede LeNet no treino com imagens Verticais utilizando o optimizador SGD 42
5.5 Evolução do valor de precisão obtida pela rede VGG16 no treino com imagens Verticais utilizando o optimizador ADAM 43
5.6 Evolução do valor de precisão obtida pela rede VGG16 no treino com imagens Verticais utilizando o optimizador SGD 43

5.7 Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense no treino com imagens Verticais utilizando o optimizador ADAM 44
5.8 Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense no treino com imagens Verticais utilizando o optimizador SGD 44
5.9 Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense usando transfer learning no treino com imagens Verticais utilizando o optimizador ADAM 45
5.10 Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense usando transfer learning no treino com imagens Verticais utilizando o optimizador SGD 46

6.1 Representação dos pontos de ouro em (a) e do valor de tolerância de 0.5 e 0.35 aos pontos de ouro em (b), representados pelos pontos vermelhos e laranja respetivamente. 64
6.2 Histograma das frações da imagem ao longo do eixo horizontal onde se localizam os olhos 65
6.3 Histograma das frações da imagem ao longo do eixo vertical onde se localizam os olhos 66
6.4 Teste do algoritmo que fornece sugestões relativo ao enquandramento 67
6.5 Teste do algoritmo que fornece sugestões relativo ao enquandramento 68
6.6 Teste do algoritmo visualizando a grelha 68
6.7 Teste nº2 do algoritmo visualizando a grelha 69

Lista de Tabelas

4.1 Valor de Iluminância e abertura em variadas condições 32

5.1 Arquitetura das redes VGG16 utilizadas 38

5.2 Comparação das CNN's pelo número de parâmetros 47

5.3 Precisão obtida por cada rede 48

5.4 Resultados obtidos na deteção de imagens com orientação vertical do plano médio 49

5.5 Resultados obtidos na deteção de imagens com orientação vertical do plano close-up 50

5.6 Resultados obtidos na deteção de imagens com orientação vertical do plano big close-up 51

5.7 Resultados obtidos na deteção de imagens com orientação vertical do plano não 52

5.8 Percentagem de confiança obtida nas imagens do plano médio . . . 53

5.9 Percentagem de confiança das imagens do plano close-up 54

5.10 Percentagem de confiança das imagens do plano big close-up 55

5.11 Percentagem de confiança das imagens do plano não 56

5.12 Modelos Horizontais Utilizados 57

6.1 Resultados dos algoritmos de deteção facial para as imagens do plano close-up utilizando 591 imagens 60

6.2 Resultados dos algoritmos de deteção facial para as imagens do plano médio utilizando 835 imagens 61

6.3 Tempo de processamento dos algoritmos de deteção facial no data set do plano médio utilizando 835 imagens 62

6.4 Resultados do algoritmo haarcascade_eye na deteção de olhos nas imagens do plano close-up utilizando 133 imagens 63

6.5 Resultados do algoritmo haarcascade_eye na deteção de olhos nas imagens do plano médio utilizando 170 imagens 63

Abreviaturas

CNN	**C**onvolution **N**eural **N**etwork (Rede Neuronal Convolucional)
SVM	**S**upport **V**ector **M**achine
LBP	**L**ocal **B**inary **P**atterns
LBPH	**L**ocal **B**inary **P**atterns **H**istograms
HOG	**H**istogram of **O**riented **G**radients
KNN	**K**–**N**earest **N**eighbour
OpenCV	**O**pen **S**ource **C**omputer **V**ision **L**ibrary
ISO	**I**nternational **O**rganization for **S**tandardization

Capítulo 1

Introdução

1.1 Enquadramento

Hoje em dia as câmaras fotográficas presentes nos dispositivos móveis (*smartphones* e *tablets*) permitem captar imagens com uma qualidade muito próxima de uma boa máquina fotográfica digital. Esta disponibilidade leva à existência de um grande número de potenciais "fotógrafos amadores". No entanto, o conhecimento sobre fotografia de grande parte destes utilizadores é pouco ou quase nenhum. Neste trabalho, pretende-se desenvolver uma aplicação para dispositivos móveis, que fornece sugestões ao utilizador sobre como melhorar a captação da fotografia. Estas sugestões são feitas em tempo real considerando a análise da imagem, e focam-se em características como o enquadramento fotográfico.

1.2 Questões e objetivos de investigação

Para abordar este tema da análise de imagem em tempo real é necessário considerar um conjunto de fatores que orientaram a continuidade deste trabalho. Foram então apontados um conjunto de questões que permitiram selecionar os objetivos para esta dissertação.

1. Em relação aos problemas da classificação de planos fotográficos e deteção facial:

   - Que tipo de arquitetura de Redes neuronais profundas (*Deep Neural Nets*) será a mais adequada para os problemas da classificação de planos fotográficos?

   - Serão os algoritmos clássicos de processamento de imagem, viáveis para deteção facial?

2. Que elementos da câmara do telemóvel é possível aceder e/ou controlar? Por exemplo: a imagem é no formato raw ou comprimida (jpg)? É possível controlar a velocidade do obturador, a abertura e o ISO (Sensibilidade fotográfica)?

3. O que diferencia uma boa imagem de uma imagem com erros?

   - Fotografia sobre-exposta e sub-exposta;

   - Fotografia mal enquadrada;

4. No caso particular das fotografias de retratos, como caracterizar os elementos descritos no ponto 3 e como dar indicação ao utilizador sobre como corrigir esses erros?

O objetivo principal é desenvolver uma aplicação Android que dê indicações ao utilizador sobre como melhorar a fotografia em tempo real (no momento em que a está a capturar).

Dos diversos tipos de fotografias existentes, esta dissertação apenas irá abordar uma em concreto, sendo ela os retratos. Nestes serão diferenciados 3 dos vários tipos de planos de fotografia (plano médio, *close-up* e *big close-up*) e analisadas as características comuns da fotografia, como o enquadramento e iluminação.

De forma a identificar qual o tipo de plano fotográfico presente é proposto o uso de uma rede neuronal convolucional (CNN), dado que a taxa de precisão destas arquiteturas é muito próximo da interpretação humana [Cireşan et al., 2012].

As CNN's tem vindo a ser utilizadas em vários tipos de problemas de classificação, assim como reconhecimento facial onde atingiram valores de precisão superiores a métodos tradicionais tais como Local Binary Patterns Histograms (LBPH), K–Nearest Neighbour (KNN) [Kamencay et al., 2017] e Suport Vector Machines (SVM's) [Singh, 2018].

A nível do enquadramento é proposto um método que utiliza um algoritmo para a deteção de olhos e com base na regra dos terços dá indicações ao utilizador sobre como corrigir o enquadramento. A regra dos terços especifica onde posicionar os pontos de interesse numa fotografia [Thirds, 2018]. De forma a garantir uma maior precisão na deteção dos olhos é utilizado um algoritmo de deteção facial e só após ser detetada uma cara procede-se para a deteção de olhos. Para tal foram comparados vários algoritmos, tanto ao nível da taxa de deteção, como do tempo de processamento.

1.3 Estrutura e organização da dissertação

Esta dissertação possui 7 capítulos e está estruturada da seguinte forma:

No segundo capítulo é apresentado o trabalho relacionado.

O terceiro capítulo descreve o processo de criação, análise e processamento do data set de imagens, utilizado para o treino das redes neuronais neste trabalho.

No quarto capítulo é apresentado o método utilizado neste trabalho para o fornecimento de sugestões ao utilizador, explicando a forma como cada elemento está interligado.

O quinto capítulo apresenta as redes neuronais convolucionais utilizadas e os resultados obtidos no treino de cada uma, sendo também apresentada uma comparação das redes com base em testes efetuados com imagens de teste.

O sexto capítulo apresenta a análise e os resultados obtidos dos algoritmos de deteção facial utilizados, assim como a análise os resultados obtidos no algoritmo

de deteção de olhos. Neste capítulo são também apresentados os testes reais efectuados na aplicação Android.

No sétimo e último capítulo são apresentadas as conclusões deste trabalho, assim como as limitações e trabalho futuro.

Capítulo 2

Trabalho relacionado

Neste capítulo são apresentadas algumas das abordagens existentes que podem ser aplicadas na resolução dos problemas de deteção facial e classificação do plano fotográfico. Para o problema da classificação do plano fotográfico foram utilizadas arquiteturas de *deep learning*. O capítulo 2.2 apresenta algumas das técnicas utilizadas nestas arquiteturas de forma a melhorar o seu desempenho.

2.1 Algoritmos para deteção facial

Nesta secção são apresentados três algoritmos estudados para a resolução do problema de deteção facial.

2.1.1 Haar Cascades

Paul Viola e Michael Jones (2001) [Viola and Jones, 2001a] propõem um método de deteção de objetos utilizando classificadores em cascata baseados em características Haar (*haar features*) que classifica as imagens com base no valor de características simples (cada característica inclui vários pixels), afirmando obter um processamento mais rápido do que um sistema que seja baseado em pixels.

Este algoritmo necessita de muitas imagens positivas e negativas do objeto a detetar (no caso de deteção facial, imagens com caras e sem caras respetivamente) para treinar o classificador. Neste método é aplicada uma janela de determinado tamanho que percorre a imagem e em cada subsecção dessa janela são calculadas as características Haar (*haar features*) demonstradas na Figura 2.1. Cada característica (*feature*) é um único valor resultante da diferença entre a soma de pixels no retângulo branco e a soma de pixels no retângulo preto.

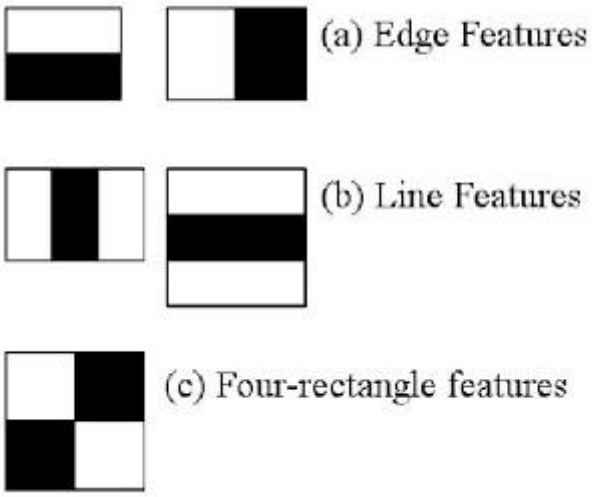


FIGURA 2.1: Exemplo de características Haar [HaarFeactures, 2018]

De forma a simplificar o cálculo destas características para cada subsecção ao longo da imagem toda, os autores apresentam o conceito de imagens integrais (*integral images*).

A imagem integral na posição x,y é a soma dos pixels acima e à esquerda de x,y inclusive:

$$ii(x, y) = \sum_{x\prime \leq x, y\prime \leq y} i(x\prime,\ y\prime)$$

Onde ii(x,y) é a imagem integral e i(x',y') é a imagem original. Usando as imagens integrais, podemos traduzir a soma de retângulos num cálculo de quatro vetores de referências (ver Figura 2.2).

Na Figura 2.2, "a soma dos pixels no retângulo D pode ser obtida com quatro vetores referências. A imagem integral no local 1 corresponde à soma dos pixels

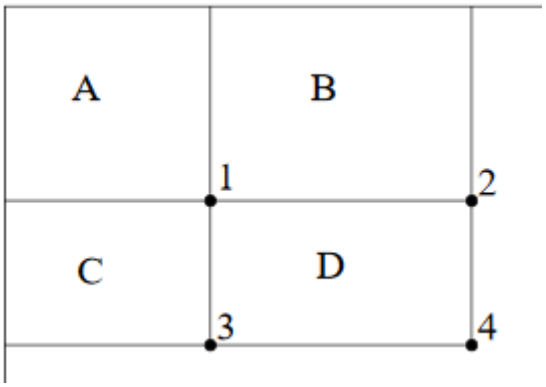


FIGURA 2.2: Imagem integral [IntegralImage, 2018]

situados no retângulo A, o seu valor no local 2 corresponde à soma A + B, na posição 3 corresponde a A + C e na localização 4 é A + B + C + D. A soma dos pixels dentro do retângulo D é igual a 4 + 1 - (2 + 3). Esta representação pode ser calculada numa única passagem pela imagem original." [Viola and Jones, 2001b]

De todas as características calculadas, a grande maioria é irrelevante. No artigo é dado o exemplo considerando a Figura 2.3. A primeira característica baseia-se na natureza da região dos olhos ser normalmente mais escura que a região do nariz e que as faces da cara. A segunda depende da particularidade de que a zona dos olhos é mais escura que a cana do nariz. No entanto, a aplicação das mesmas janelas noutra parte da imagem é irrelevante.

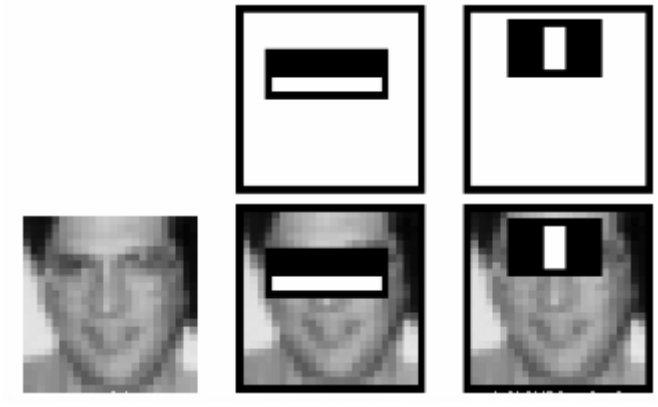


FIGURA 2.3: Exemplo de duas características Haar [HaarFeacturesB, 2018]

De forma a identificar quais as melhores características a aplicar, utiliza-se

um algoritmo chamado AdaBoost[1] e em cada característica é aplicado o melhor *threshold* (limite) que classifica as caras como positivas ou negativas (compara o valor da característica com o *threshold* obtido pelo treino das imagens).

Dado que um classificador baseado em características Haar é considerado fraco [Viola and Jones, 2001a] (a sua qualidade de deteção é ligeiramente melhor do que adivinhar), é necessário um grande número destas características para detalhar um objeto com exatidão, o que o torna ineficiente. A solução encontrada pelos autores foi organizar as características ("*Haar-like features*") em vários níveis, criando um classificador forte. Desta forma, se uma janela for rejeitada num nível (se for classificada como não sendo uma cara), é logo descartada e não são consideradas as características nela restantes. A este conceito chamou-se classificadores em cascata (*Cascade Classifiers*, ver Figura 2.4). O seu classificador final possui 38 níveis com um total de 6061 características, sendo que os primeiros 5 níveis possuem 1, 10, 25, 25 e 50 características respetivamente, onde o uso exclusivo das características presentes nos primeiros níveis leva a uma taxa de erro entre 0.1 e 0.3 na classificação de caras, e usando unicamente as características dos últimos níveis leva a uma taxa de erro entre 0.4 e 0.5 [Viola and Jones, 2001a].

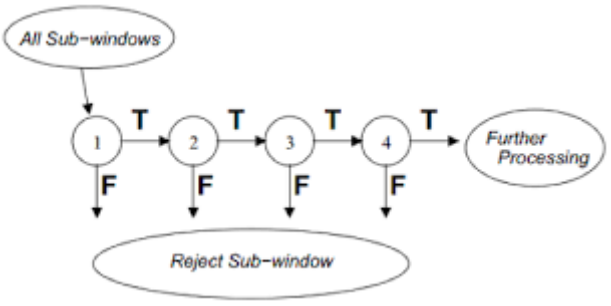


FIGURA 2.4: Exemplo de Classificador em Cascata [Classifier, 2018]

A principal vantagem das características Haar em relação a outras é a velocidade de processamento nos cálculos. Devido ao uso de imagens integrais, as características Haar podem ser calculadas em tempo constante independentemente

[1]"O algoritmo AdaBoost é muitas vezes utilizado para aumentar o desempenho de classificação de um algoritmo de *machine learning* simples (às vezes denominado fraco)."

do seu tamanho (aproximadamente 60 instruções de microprocessador para uma feature (característica) de 2 retângulos) [Viola and Jones, 2001a].

No entanto uma das limitações deste algoritmo é o facto das características Haar não serem robustas face às alterações de iluminação. [Rodriguez, 2006]

2.1.2 Histogram of Oriented Gradients – HOG

Dalal e Triggs (2005) [Dalal and Triggs, 2005] apresentam um classificador de objetos de alta precisão capaz de detetar pessoas (deteção humana) utilizando um histograma de gradientes orientados como descritor de imagens (Histogram of Oriented Gradients - HOG) e uma SVM Linear (Support Vector Machine) para o treino do classificador.

A ideia deste algoritmo é de que a aparência e a forma de um objeto local podem ser caracterizadas pela distribuição de gradientes de intensidade locais [Dalal and Triggs, 2005]. Para isto é analisado cada pixel e os seus vizinhos, tentando observar o quão escuro esse pixel é comparado com os pixels à sua volta e substituem-se os pixels por um gradiente com a direção do grupo de pixels mais claro para o mais escuro.

FIGURA 2.5: Representação de uma característica HOG [hog, 2018]

Aplicando este método a todos os pixels, estes são substituídos por setas e ficamos com uma imagem de gradientes, que mostram o fluxo do mais claro para o mais escuro.

Considerar desta forma apenas a direção da mudança do brilho atenua o problema da variação de luz ou alto contraste entre o objeto a detetar e o fundo [Cruz et al., 2015].

Analisando os gradientes para cada pixel obtemos demasiado detalhe. De forma a obter um fluxo básico de claro/escuro divide-se a imagem em quadrados de 16x16 pixels. Para cada quadrado conta-se quantos pontos de gradiente apontam em cada direção principal (quantos apontam para cima, quantos apontam na diagonal direita para cima, quantos apontam para a direita, etc...), criando assim um histograma de direções de gradientes, e substitui-se esse quadrado na imagem com uma seta na direção mais forte (característica HOG, ver Figura 2.5), resultando numa representação simples que captura a estrutura básica do objeto em questão. No caso específico de deteção facial, uma cara.

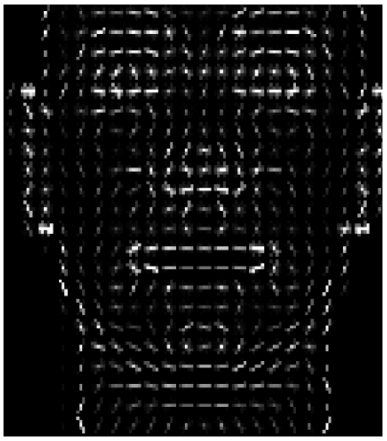


FIGURA 2.6: Representação da estrutura de uma cara numa imagem HOG [Rojas et al., 2011]

De forma a treinar este algoritmo é (também) necessário um grande conjunto de imagens do objeto a detetar e outro conjunto de imagens sem esse objeto, imagens positivas e negativas respetivamente. Em seguida é necessário extrair as características HOG de cada uma das imagens, que são depois utilizadas por uma SVM linear para o treino [Rosebrock, 2018a].

Para encontrar caras numa imagem HOG, apenas é necessário encontrar a parte da imagem mais semelhante a um padrão HOG conhecido. Este padrão é extraído de outras tantas imagens de caras de treino.

2.1.3 LBP Cascades

O conceito base do algoritmo *Local Binary Patterns* (LBP) é de que características comuns como arestas, linhas e pontos podem ser descritas num valor numérico, tornando possível a identificação de objetos numa imagem através de um conjunto de valores previamente extraídos [Cruz et al., 2015].

O operador LBP original classifica cada pixel comparando-o com os seus pixels vizinhos numa janela de 3x3. Ex: Numa janela de 3x3 compara-se o pixel central com os pixels vizinhos, se a intensidade (numa escala de cinzentos) do pixel vizinho for superior ao pixel central o pixel vizinho fica com o valor de 1, caso contrário fica com o valor de 0, resultando num número binário.

A Figura 2.7 demonstra um exemplo desta operação.

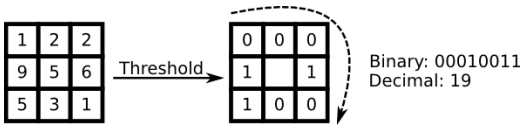


FIGURA 2.7: Cálculo do LBP [CalcLBP, 2018]

Aplicando esta técnica à imagem toda e criando um histograma da ocorrência dos padrões, obtemos um descritor de texturas. Cada padrão LBP no histograma pode ser considerado como uma microestrutura, como por exemplo linhas, pontos, áreas planas, arestas. Estes são também denominados de primitivas locais, como observado na Figura 2.8

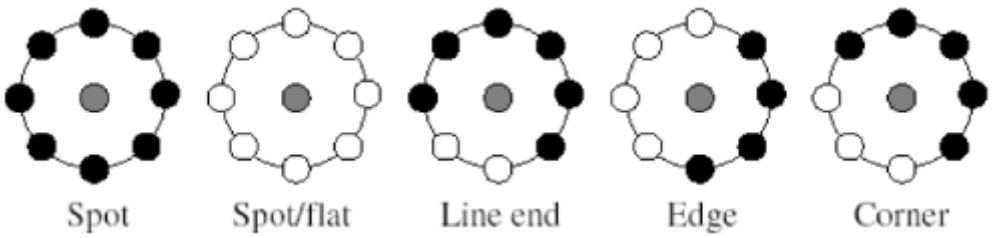


FIGURA 2.8: Primitivas locais ou Padrões uniformes [Chang-Yeon, 2008c]

Este operador foi modificado de forma a ser possível abranger um maior número de pixels vizinhos.

A designação LBPP,R representa uma vizinhança de P pixels num círculo de raio R, originando 2^P resultados diferentes, que correspondem ao número de padrões binários possíveis constituídos pelos P pixels no conjunto dos vizinhos.

Exemplo: O LBP4,1 abrange 4 vizinhos enquanto que com o LBP16,2 é possível abranger 16 pixels vizinhos num raio de 2 pixels.

Ojala et al. em [Ojala et al., 2002] mostra que certos padrões constituem propriedades essenciais de texturas, constituindo cerca de 90% dos padrões detetados, denominando-os de padrões "uniformes". Estes são padrões que possuem entre 0 e 2 transições de bits. Exemplo os números 00000000 e 11111111 contêm 0 transições e os números 00000110 e 01111110 contêm 2 transições (Ver Figura 2.8).

Ahonen et al [Ahonen et al., 2006] mostra que estes mesmos padrões "uniformes" constituem entre 85,2% e 90,6% dos padrões detetados em imagens faciais.

Um fator bastante significativo dos padrões LBP é a sua tolerância face às alterações de iluminação e a sua simplicidade de processamento de cálculos.

2.1.3.1 Representação facial baseada no LBP

Cada imagem de uma cara pode ser encarada como um conjunto de micro-padrões eficazmente identificados por um operador LBP.

Ahonen et al. apresenta uma representação para reconhecimento facial baseada em Local Binary Patterns. Esta representação passa por dividir cada imagem facial em M regiões (não sobrepostas) e para cada uma destas regiões é criado um histograma com os padrões extraídos (os padrões com mais de 2 transações são agrupados e contam como uma única entrada), sendo depois concatenados num único histograma espacial de características (Figura 2.9). O histograma de características LBP extraídas representa a forma global de uma cara.

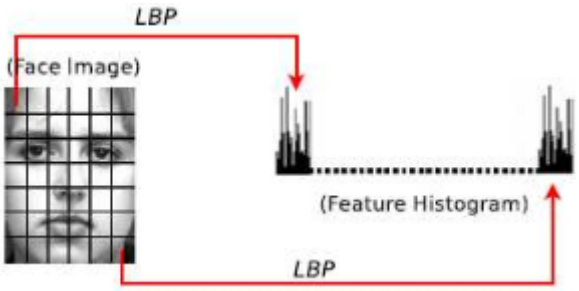


FIGURA 2.9: Representação facial baseada no operador LBP [Chang-Yeon, 2008a]

Jo Chang-yeon [Chang-Yeon, 2008b], utiliza uma variante do algoritmo Adaboost para escolher as melhores características LBP que definem uma cara, as quais são depois passadas para um classificador em cascata (cascade classifier).

A versão de LBP usada neste trabalho foi a versão disponível na biblioteca do OpenCV que é implementada com classificador em cascata.

2.2 Técnicas de Deep Learning usadas para a classificação dos planos fotográficos

2.2.1 Deep Learning

Deep Learning é um ramo de *machine learning* [Chollet, 2017], sendo uma nova abordagem na aprendizagem de representações de dados, a qual se destaca por aprender camadas sucessivas de representações relevantes dos dados.

Em *deep learning* é aplicado um grande número de camadas de representações, enquanto que noutras abordagens de *machine learning* são aplicadas apenas uma ou duas camadas de representação dos dados para aprendizagem
[Deng and Yu, 2014].

As arquiteturas de *deep learning* (redes neuronais profundas) tais como CNN's, têm sido utilizadas em tarefas de *computer vison* (visão computacional), as quais

se possuindo um número suficiente de dados de treino (imagens) corretamente classificados, são capazes de extrair tanta informação dos dados quanto uma pessoa [Chollet, 2017]. Esta extração de características é utilizada para o treino da rede, onde têm vindo atingir resultados muito próximos da interpretação humana [Cireşan et al., 2012] [Krizhevsky et al., 2012].

Por esta mesma razão neste trabalho foi proposta a utilização de CNN's para a classificação dos planos fotográficos.

2.2.1.1 Overfitting

Um dos problemas no uso de Deep neural nets é o *overfitting* (sobreajuste), onde o modelo se adapta muito bem aos dados de treino mas falha ao ser usado com dados novos ("decora" características específicas desses dados, não conseguindo generalizar).

Isto geralmente acontece quando se usam data sets (conjunto de dados) pequenos. Dado que não existe um número exato de quantos dados são necessários para o treino de uma rede neuronal (sendo também dependente da complexidade do caso a classificar) de forma a evitar o *overfitting* foram utilizadas técnicas tais como *data augmentation, dropout* ou *transfer-learning*, descritas nas secções seguintes.

Para além destas técnicas, todas as arquiteturas utilizadas neste trabalho possuem camadas de *Pooling*, aplicando a operação de *Max Pooling*. A função destas camadas é reduzir progressivamente a dimensão do *input* (as características extraídas), de forma a reduzir a quantidade de parâmetros calculados pela rede, sendo também uma forma de controlar o *overfitting* [Karpathy, 2018a]. A operação de *Max Pooling* consiste em extrair o valor máximo de cada canal dentro de uma janela de extração, que se move ao longo do comprimento e altura do *input*, reduzindo os pixels dessa janela para apenas um pixel [Chollet, 2017].

Ex: Para uma janela de dimensão 2x2 e um passo (*stride*) de 2, a dimensão da amostra é reduzida para metade, onde o *output* é constituído pelo valor máximo de cada janela (Ver Figura 2.10).

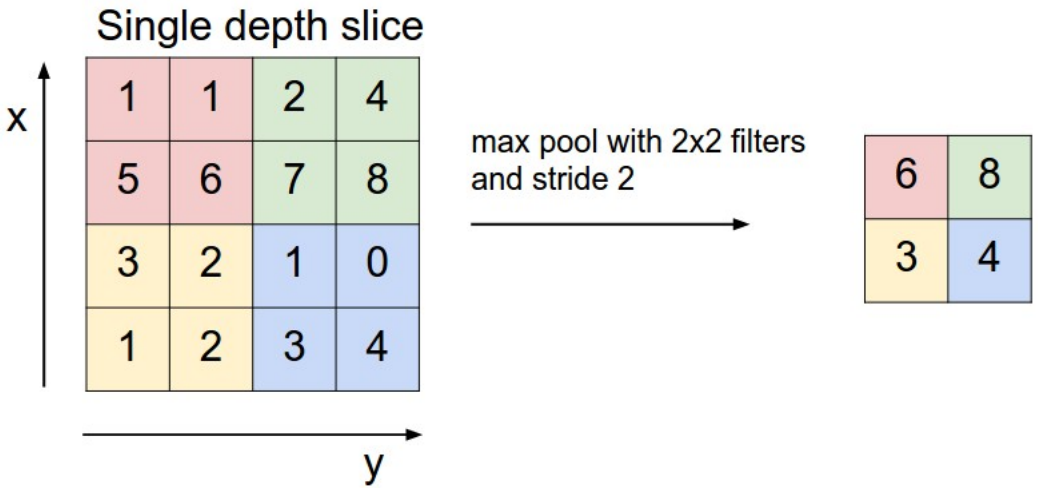


FIGURA 2.10: Exemplo da operação de MaxPooling [Karpathy, 2018a]

2.2.1.2 Data augmentation

Foi utilizado uma técnica de *data augmentation* (incremento de dados). Esta técnica aplica transformações a cada imagem, criando uma nova imagem e desta forma aumentando o data set [Keras, 2018c]. No treino das CNN's apenas foi aplicada aleatoriamente uma transformação de *horizontal_flip* (efeito de espelho) a cada imagem.

Para além de aumentar o data set, como as transformações de *data augmentation* são aleatórias, quanto maior for o número de transformações utilizadas maior é a diversidade de imagens geradas. Ou seja, durante o treino de um modelo, em cada iteração pelo data set, este nunca "vê" as mesmas imagens, o que nos permite evitar/reduzir as situações de *overfit* (sobreajuste) [Krizhevsky et al., 2012]. A Figura 2.11 apresenta algumas das transformações possíveis.

FIGURA 2.11: Exemplo de transformações de data augmentation (adaptado de [Wang, 2018])

2.2.1.3 Dropout

Em [Srivastava et al., 2014] os autores apresentam uma técnica para a resolução do *overfitting* denominada *dropout*, que consiste em descartar aleatoriamente nós (junto com suas ligações) da rede neuronal durante o treino, evitando a co-adaptação dos nós em demasia.

Neste trabalho foi utilizado *dropout* na arquitetura da rede VGG16, com um novo bloco dense, após o último bloco de convoluções e imediatamente antes da camada de *output*, tal como sugerido num exemplo em [Chollet, 2017]. (Ver Tabela 5.1).

A rede VGG16 foi proposta por [Simonyan and Zisserman, 2014] e utilizada no ImageNet Large Scale Visual Recognition Competition (ILSVRC) [ImageNet, 2018a].

2.2.1.4 Transfer-Learning

Para esta mesma arquitetura foi também utilizado *transfer-learning* (transferência de aprendizagem), utilizando pesos da rede VGG16 pré-treinada no *data set*

ImageNet[2].

Transfer-learning consiste em usar um modelo pré-treinado (os pesos da rede pré-treinada) numa tarefa e aplicá-lo numa outra tarefa semelhante, o que acelera o progresso e melhora a sua precisão no treino desta nova tarefa.[Brownlee, 2018]

Uma CNN é uma sequência de camadas. Na construção das CNN's são utilizados três tipos principais de camadas: Camada de Convolução (constituída por um conjunto de filtros de convolução, responsáveis por extrair padrões locais como arestas, texturas, entre outros), Camada de *Pooling* e Camada *Fully-Connected* [Karpathy, 2018b] [Chollet, 2017].

Uma abordagem de *transfer-learning* é extração de características (*feature extraction*) onde é apenas usada a base de convolução do modelo pré-treinado, ficando congelada durante o treino, sendo unicamente treinada a base *fully-connected* com os novos dados. Neste caso foi utilizado a base de convolução da rede VGG16 pré-treinada e foi adicionado um novo bloco *fully-connected* (ver Tabela **??**).

Quando parte do modelo não é alterada ou afetada pelo treino, diz-se que o modelo está congelado. Sendo que as suas camadas são inicializadas com os pesos pré-treinados.

Foi também utilizada uma técnica de *fine-tuning* (ajuste) ao modelo, consistindo em descongelar algumas das camadas superiores de uma base de um modelo congelado usada para a extração de características e no treino conjunto da parte recém-adicionada do modelo (o bloco *fully-connected*) com essas camadas superiores. A isto chamamos *fine-tuning* porque ajusta parcialmente as representações mais abstratas do modelo reutilizado, a fim de torná-las mais relevantes para o problema em questão [Chollet, 2017].

Estas técnicas podem ser aplicadas mesmo que as imagens que queremos classificar nada tenham a ver com o problema para o qual a rede tenha sido treinada,

[2]ImageNet é um banco de dados de imagens, constituído por centenas e milhares de imagens de várias categorias (espécies de animais, tipos de objetos, pessoas, etc...). Atualmente, possui em média mais de quinhentas imagens por categoria.[ImageNet, 2018b]

porque a base de convolução extrai características genéricas que podem ser reutilizáveis.

Na base de convolução as camadas iniciais extraem características mais gerais e reutilizáveis enquanto que as camadas finais extraem características mais específicas do problema.

Para casos em que o problema a classificar é muito diferente, é aplicado o *fine-tuning* às últimas camadas de convolução (i.e. descongelar algumas camadas de topo da base de convolução, deixando o resto do modelo congelado).

Desta forma, neste trabalho foram descongeladas as últimas 3 camadas de convolução (5º bloco de convolução) e adicionado um bloco *fully-connected*.

2.3 Aplicações Semelhantes

A *Huawei* (empresa multinacional de telecomunicações) lançou o *smartphone Huawei Mate 10 Pro* com uma aplicação para a câmara com recurso a um algoritmo de inteligência artificial.

O algoritmo identifica em tempo real e em modo *offline* 13 tipos de cenas e objetos, ajustando a cor, contraste, brilho e exposição, de forma a produzir fotos atraentes. Estas alterações são automáticas e resultam do processamento e análise de mais de 100 milhões de imagens, por algoritmos projetados pela *Huawei* [HuaweiMate10, 2018].

Capítulo 3

Data Set

Para o treino das redes neuronais é necessário um grande conjunto de exemplares (fotografias) de cada plano fotográfico, assim como um grande número de fotografias que não pertencem a nenhum dos planos, designado no decorrer deste trabalho como "plano não". De forma a ser possível classificar cada um é essencial definir o que se entende por plano e como se define/distingue cada um.

3.1 Critérios para a definição dos planos fotográficos

Dado que não existe uma definição "concreta" dos planos fotográficos, os critérios utilizados para a sua classificação, passaram por uma pesquisa em vários sites (Ver Anexo A) sobre a definição desses planos e foram retirados os pontos em comum.

Em seguida é apresentada a definição utilizada para cada plano.

Comum a todos os planos:

-1 pessoa;

- Necessário conter olhos nariz e boca.

Plano Close up:

- Linha dos ombros tem de estar presente;

- Inclui a cabeça mais espaço para os lados (definido como o espaço suficiente para serem visíveis os ombros);

- Pouco abaixo das axilas ou ao nível, mas sem espaço acima da cabeça.

FIGURA 3.1: Ilustração do plano CloseUp [PlanosFotográficos, 2017]

Plano Big Close up:

- Apenas contém cara ou corte da cara;

- Não tem espaço para os lados nem para cima.

FIGURA 3.2: Ilustração do plano Big CloseUp [PlanosFotográficos, 2017]

Plano Médio:

- Corte acima da cintura;

- Contém ombros e cabeça inteira;

- Pode ter espaço acima da cabeça;

-Tem de existir espaço para os lados dos ombros.

FIGURA 3.3: Ilustração do plano Médio [PlanosFotográficos, 2017]

3.2 Data set utilizado

O data set utilizado para o treino da rede neuronal na identificação dos planos imagens, tem de satisfazer as seguintes características:

- Têm de ser obrigatoriamente constituído por fotografias de pessoas;

- A qualidade das fotografias tem de ser ao nível da qualidade proveniente de um *smartphone* (normalmente a qualidade das fotografias de um *smartphone* é inferior a uma máquina fotográfica DSLR/SLR);

- Ter um grande conjunto de fotografias para cada plano, neste caso para o plano médio, close-up e big close-up (1000 fotografias no mínimo);

Dado que não foi encontrado nenhum data set que satisfizesse as características apresentadas, este teve de ser criado. Para tal foram analisados vários data sets públicos de fotografias de pessoas [Judd et al., 2009] [MIT, 2017] e extraídas as imagens que possuíam as características referidas. Foram também recolhidas fotografias do Instagram, dado que a grande maioria das fotos desta rede social são de pessoas e são tiradas por telemóveis.

Para tal foi utilizado o *script* de código aberto [instagram scraper, 2017] para fazer download de fotografias do Instagram , que permite fazer a seleção das fotografias por "hashtag". Todas as fotos recolhidas pelo *script* são públicas.

Data Set usado na deteção facial e de olhos

Para o teste dos vários algoritmos de deteção facial e do algoritmo de deteção de olhos, foi criado um novo data set utilizando as imagens usadas no treino das redes neuronais consideradas como tendo um bom enquadramento.

Este data set é constituído por um total de 1426 imagens pertencentes aos planos *close up* e médio, com 591 e 835 imagens respetivamente. Cada uma destas imagens apenas contém uma pessoa.

3.3 Análise e Processamento do Data Set

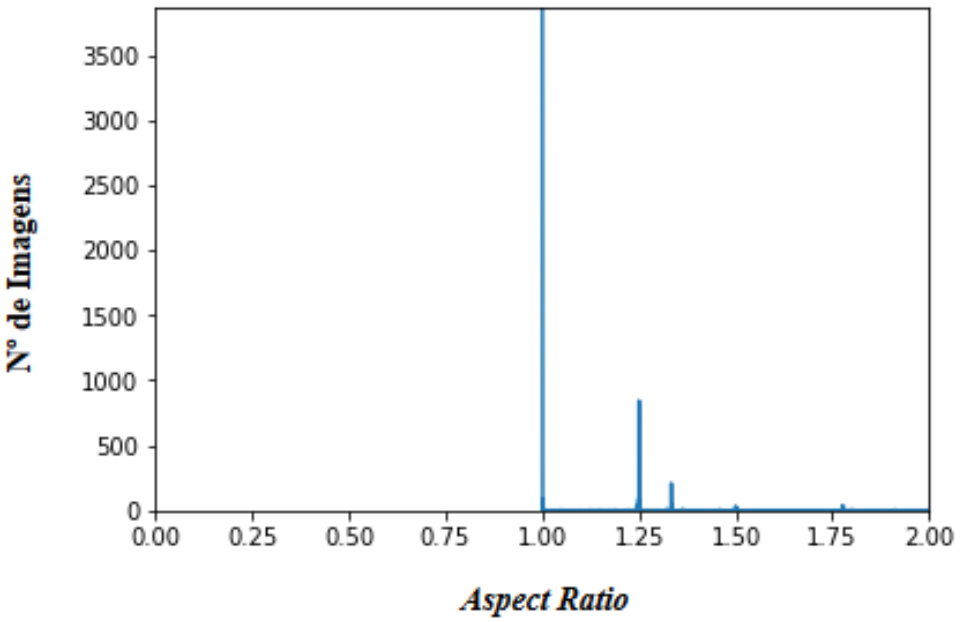
Inicialmente foram recolhidas 30.000 imagens de retratos da internet, as quais foram classificadas individualmente como plano *close-up*, plano médio ou plano não, este último constituído por todas as imagens que não pertencem nem ao plano médio nem ao plano *close-up*.

Após a recolha e classificação manual de imagens por planos, foi realizado um histograma com os *aspect ratios* (formatos) das imagens de cada plano, com o objetivo de determinar qual o *aspect ratio* a ser utilizado no seu processamento para o treino da rede neuronal. A importância da análise do *aspect ratio* deve-se ao facto das CNN's receberem como *input* uma imagem de determinada dimensão, sendo necessário redimensionar todas as imagens para esta dimensão. Ao redimensionar as imagens mantendo o mesmo *aspect ratio* evita-se a distorção da imagem. O *aspect ratio* ou relação de aspeto de uma imagem representa a relação entre a maior e menor dimensão da imagem (comprimento e altura) [Ratio, 2018] [Cedric Demers, 2018] e é dado por $\frac{comprimento}{altura}$ ou $\frac{altura}{comprimento}$ no caso da orientação da imagem ser horizontal ou vertical respetivamente.

Ex: Uma imagem com dimensão 200x150 (comprimento x altura) e uma imagem com dimensão 400x300 têm a mesma relação de aspeto, neste caso igual a 1.333 (formato 4:3).

Plano Não

Nº de imagens para plano não: 6457

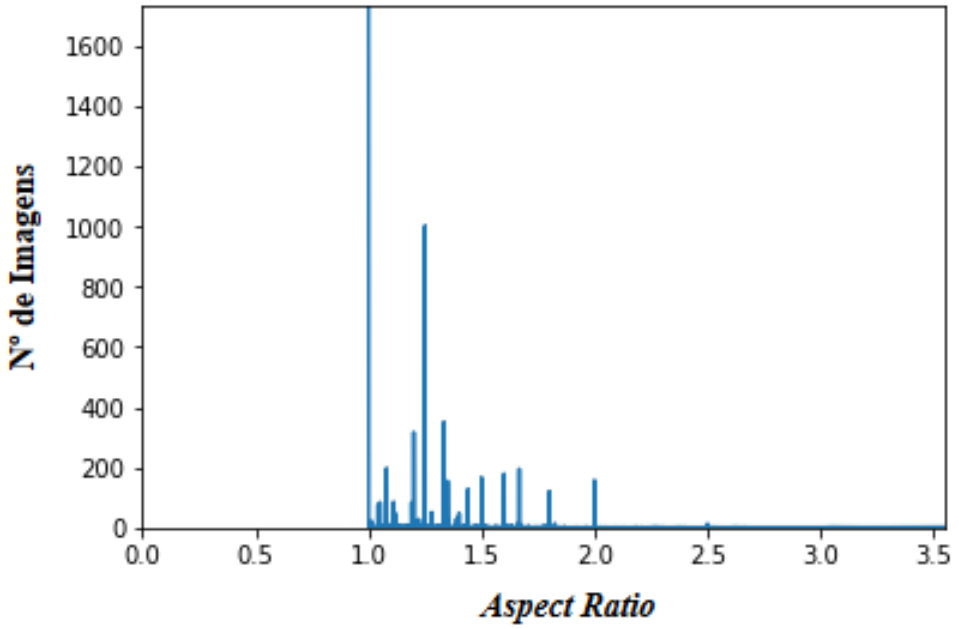


Nº de imagens com o *aspect ratio* mais comum: 3860
Ratio mais elevado: 2.0

FIGURA 3.4: Histograma dos valores do aspect ratio das imagens do Plano Não

Plano Médio

Nº de imagens para plano médio: 8722



Nº de imagens com o *aspect ratio* mais comum: 1731

Ratio mais elevado: 3.553

FIGURA 3.5: Histograma dos valores do aspect ratio das imagens do Plano Médio

Plano Close-Up

Nº de imagens para plano close-up: 6945

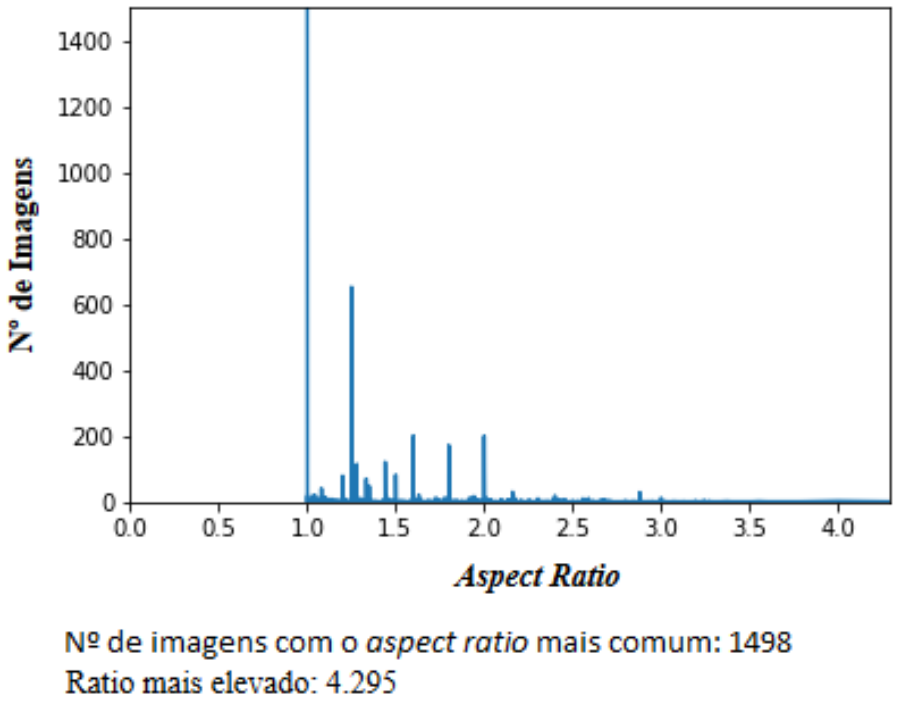


FIGURA 3.6: Histograma dos valores do aspect ratio das imagens do Plano Close-Up

Após a análise verificou-se que o *aspect ratio* mais comum se situava entre o 1:1 (1) e 4:3 (1.333).

Observando as imagens com *aspect ratio* de 1:1 constatou-se que estas possuíam muito espaço para os lados do sujeito e para cima da cabeça deste, não se enquadrando na definição usada para cada um dos planos. Por esta razão e pelo facto de os formatos mais comuns de fotografia serem 4:3 e 16:9, foi escolhido o *aspect ratio* 4:3 para ser usado nas imagens do data set.

Isto implicou o corte manual de cada imagem com o respectivo *aspect ratio*, de forma a se enquadrarem dentro dos critérios de cada um dos planos. Para tal foi criado um *script* que ao clicar num ponto da imagem cria uma janela com *aspect ratio* 4:3 (e dimensões mínimas de 200x150) a qual pode ser aumentada,

mantendo sempre o mesmo *aspect ratio*. O *script* permite também alternar a orientação (horizontal e vertical) da zona a ser cortada.

Verificou-se também que as imagens do plano *close-up* se podiam separar em duas classes distintas, ***close up*** e ***big close up***, dado que existiam fotos onde a cabeça do sujeito fotografado ocupava a imagem toda. (Ver Figuras 3.7 e 3.8)

FIGURA 3.7: Exemplo de uma imagem do plano Close-Up

FIGURA 3.8: Exemplo de uma imagem do plano Big Close-Up

Capítulo 4

Método para sugestões fotográficas

Neste capítulo é apresentado o método utilizado para dar sugestões ao utilizador de como melhorar o enquadramento, explicando a forma de como as CNN's se interligam com os algoritmos de deteção facial e de olhos. O capítulo 4.3 apresenta o estudo efetuado sobre a iluminação e de que forma pode ser controlada através de um *smartphone*.

Para este trabalho é apenas abordado um tipo de fotografia, os retratos. Nestes são apenas diferenciados três dos vários tipos de planos de fotografia, o **plano médio, plano *close-up* e plano *big close-up*** e de forma a melhorar a aquisição da fotografia, são consideradas características como o enquadramento e iluminação.

De forma a identificar que o utilizador está a fotografar uma pessoa (retrato), é utilizado um algoritmo de reconhecimento facial. Em caso afirmativo procede-se à identificação dos planos.

4.1 Processamento de imagem

Para o processamento de imagem é utilizado o OpenCV (*Open Source Computer Vision Library*: [openCV, 2018]) que é uma biblioteca de Código aberto (licença

BSD) que inclui vários algoritmos de visão computacional (*computer vision*) nomeadamente algoritmos de deteção facial. Esta biblioteca suporta Android e está em constante desenvolvimento.

4.2 Método proposto para a melhoria do enquadramento fotográfico

4.2.1 Enquadramento fotográfico

O enquadramento fotográfico, é a forma como estão dispostos os elementos na cena a ser fotografada. Exemplo: Posicionar a câmara na horizontal ou vertical altera o enquadramento.

Existem no entanto algumas "regras" que ajudam a fazer um bom enquadramento, tal como a espiral de ouro e a regra dos terços.

O enquadramento depende do tipo de plano e como dito anteriormente, os planos de fotografia abordados neste trabalho são o plano médio, *close-up* e *big close-up*.

De forma a identificar qual o plano em que o sujeito está a ser fotografado, é utilizada uma CNN (*Convolutional Neural Network* - rede neuronal convolucional) e caso este seja um plano médio ou *close-up* é aplicado um algoritmo de deteção de olhos, que com base na sua posição e na regra dos terços transmite indicações ao utilizador com a direção para onde deve mover a câmara.

Estas sugestões são apenas transmitidas nos planos médio e *close-up*, devido ao plano *big close-up* ser um plano de detalhe e normalmente usado para transmitir as expressões do sujeito a ser fotografado, o que determinou que as mesmas não se adequavam a este tipo de plano.

Regra dos terços

A regra dos terços consiste em dividir a imagem em 9 quadrados iguais, descrevendo 2 linhas horizontais e 2 linhas verticais. Os pontos onde estas linhas se cruzam (pontos de ouro, a vermelho na Figura 4.1) são os pontos de maior interesse da imagem, sendo que o elemento que se quer realçar deve ser colocado num destes pontos. [ThirdsB, 2018]

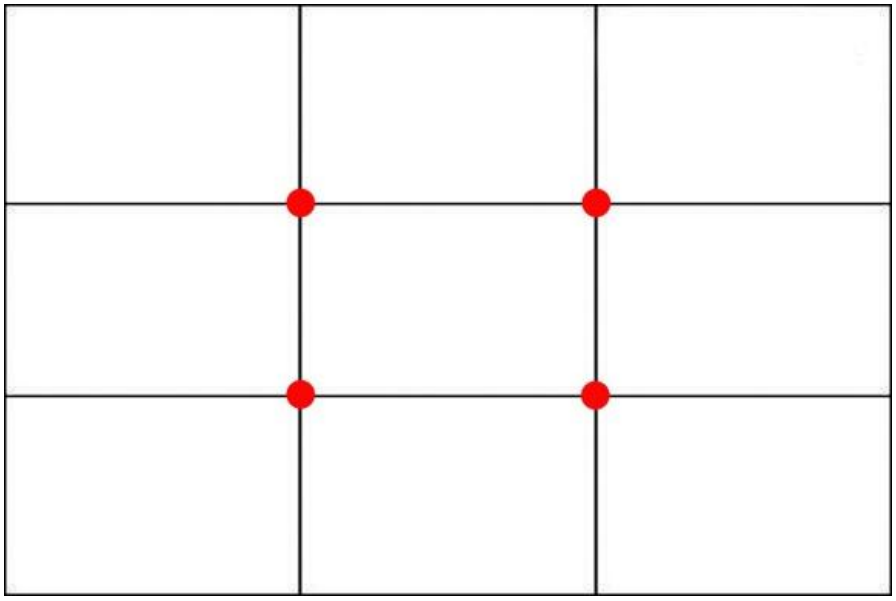


FIGURA 4.1: Pontos de ouro [pOuro, 2017]

4.2.2 Algoritmo Haar-Cascade para detectar olhos

De forma a ser possível dar sugestões ao utilizador sobre se o enquadramento está ou não correto, é necessário detetar os olhos de uma pessoa numa fotografia e fazer coincidir com os pontos de ouro da regra dos terços (regra de fotografia). Para tal são necessários 2 passos, detetar a cara de uma pessoa na fotografia e detetar os olhos da pessoa na região classificada como uma cara.

A biblioteca OpenCV inclui um algoritmo Haar-Cascade para problemas de deteção, dando a possibilidade de treinar o nosso classificador para qualquer objeto. O OpenCV já vem com vários classificadores pré-treinados para caras, olhos, entre outros, não sendo necessário para este caso treinar o classificador.

Após identificar os olhos do sujeito fotografado, assim como a sua posição/localização na fotografia, são dadas indicações ao utilizador, por meio de uma seta, indicando para que direção (esquerda ou direita) deve mover a câmara de modo a que um dos olhos fique situado num dos pontos de ouro da regra dos terços (Ver Imagem 4.2).

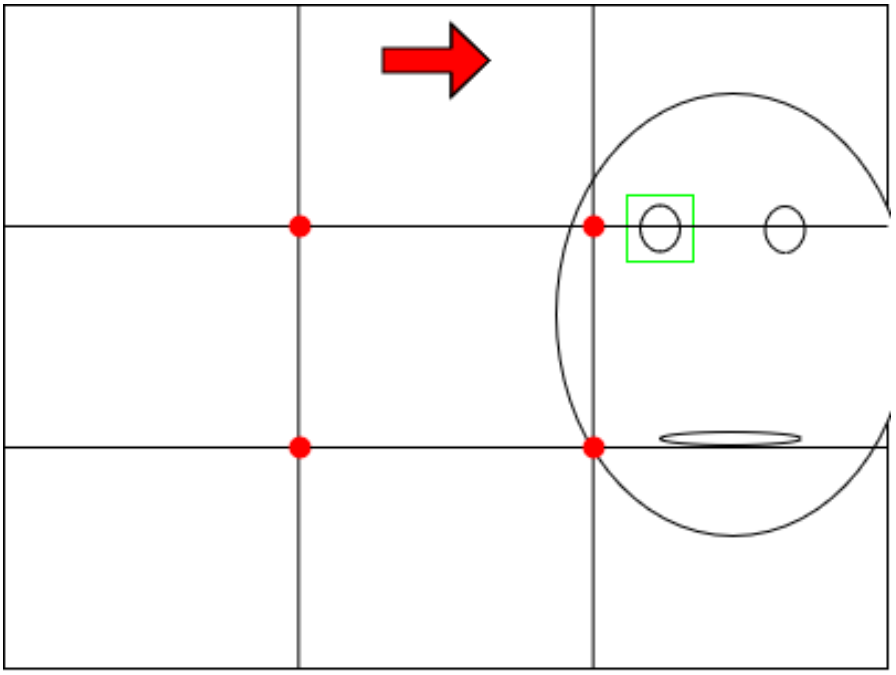


FIGURA 4.2: Esboço Indicações sobre enquadramento com base na posição dos olhos e dos pontos de ouro

De notar que as indicações observadas na Imagem 4.2 são dadas relativamente ao utilizador (o fotógrafo), sendo este aquele que se deve mover e não o "modelo".

4.3 Método para sugestões relativas à iluminação

Neste trabalho foi também previsto dar indicações ao utilizador sobre o nível de iluminação, no entanto apenas foi feito o estudo sobre como efetuar a medição de luz e o levantamento das características presentes num *smartphone*, assim como quais a características da câmara a que permitem aceder de forma a controlar o nível de luminosidade.

4.3.1 Iluminação

A luz é a característica mais importante da fotografia, pois em primeiro lugar sem ela não existe fotografia.

Existindo luz, nem sempre nos deparamos com as condições perfeitas à captação da fotografia, seja por falta ou excesso de luz, o que leva à necessidade de controlar características da câmara como o nível do ISO, abertura e tempo de exposição, para compensar estas condições.

O ISO refere-se ao nível de sensibilidade à luz do filme (fotografia analógica) ou do sensor de imagem (fotografia digital), sendo mais sensível quanto maior for o valor do ISO. Alterar os valores do ISO corresponde a uma alteração no nível de exposição, sendo utilizados valores mais altos em ambientes com baixo nível de luminosidade. No entanto aumentar os valores de ISO implica também uma redução na qualidade da fotografia com o aparecimento de ruído na imagem [Mathies, 2018a].

A abertura controla a quantidade de luz que chega ao sensor e mede-se em f-stops, onde o nível de abertura é tanto maior quanto menor for o valor da f-stop. Ex: f/2 representa uma abertura grande e f/22 uma abertura pequena, diminuindo a quantidade de luz que chega ao sensor [Mansurov, 2018].

O tempo de exposição determina a quantidade de tempo que o sensor fica exposto à luz, a sua unidade de medida é o segundo [Mathies, 2018b].

Uma boa exposição luminosa obtém-se utilizando em conjunto o ISO, abertura e tempo de exposição. A alteração nos valores de um destes componentes obriga à alteração dos outros de forma a preservar o nível de exposição [Mathies, 2018b]. Ex: Num cenário com muita luz, uma grande abertura leva à necessidade de um tempo de exposição mais reduzido ou de um valor de ISO mais baixo.

Existem duas formas de medir o nível de luz presente: medindo a iluminância com base no sensor de luz existente no *smartphone* ou através do cálculo da luminância. [light levels, 2017]

4.3.1.1 Iluminância

A iluminância é o fluxo total de luz que incidente numa superfície, por unidade de área. [light levels, 2017]

Na Tabela 4.1 podemos observar alguns valores comuns de iluminância em certas condições e qual o valor aproximado de abertura da lente (para um tempo de exposição igual a 1/30 segundos e um valor de ISO de 400 [Kodak, 2017]) de modo a captar uma fotografia nessas condições.

TABELA 4.1: Valor de Iluminância e abertura em variadas condições [Levels, 2017] [Levels2, 2017] [Kodak, 2017]

Condições	**Iluminância (Lux)**	**Valor de abertura (f)**
Dia nublado muito escuro	100	2.0 - 2.8
Iluminação de escritório	320-500	5.6
Dia nublado	1000	8.0
	2000	11.0
	4000	16.0
Plena luz do dia	10,000-20,000	32.0

4.3.1.2 Luminância

A luminância, mede a intensidade luminosa refletida por uma superfície, por unidade de área de luz numa determinada direção (cd/m2). [light levels, 2017]

Para calcular a luminância da fotografia é necessário converter os valores de RGB[1] (que estão comprimidos em gama) para RGB linear, utilizando a fórmula 4.1 [RgbToLinear, 2018] (mas primeiro dividem-se por 255 os valores de RGB passando de uma escala de 0 a 255 para uma escala de 0 a 1).

[1]Valores de red, green e blue (vermelho, verde e azul) de cada pixel

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & C_{\text{srgb}} \leq 0.04045 \\ \left(\frac{C_{\text{srgb}}+a}{1+a}\right)^{2.4}, & C_{\text{srgb}} > 0.04045 \end{cases} \tag{4.1}$$

Onde a=0.055 e C é o valor de R, G ou B

Em seguida, é aplicada a fórmula da luminância relativa (que possui os valores de luminância normalizados entre 1 e 100 de acordo com um branco de referência) a cada pixel, obtendo desta forma a luminância. [Luminance, 2017]

$$Y = 0.2126 * R + 0.7152 * G + 0.0722 * B \tag{4.2}$$

4.3.2 Características dos smartphones

De forma a dar sugestões ao utilizador relativamente ao nível de luminosidade, houve a necessidade de fazer um levantamento das características dos *smartphones* em geral e das características da câmara, neles presentes, que são possíveis controlar.

4.3.2.1 Sensor de Luz

Existe um sensor de luz na maioria dos *smartphones* que pode ser utilizado para medir a iluminância (em Lux), mas como este se encontra na parte da frente do *smartphone*[2] só pode ser utilizado com a câmara frontal (ex: selfies).

Desta forma, para medir a intensidade luminosa recorre-se ao cálculo da luminância para cada pixel (Ver equação 4.2) e calcula-se a média.

[2]Normalmente o sensor de luz é utilizado para adaptar o nível do brilho do smartphone às condições de luz presentes. Ex: Na rua com o sol a refletir no ecrã, o nível do brilho aumenta de forma a que o conteúdo do ecrã seja visível.

4.3.2.2 Controlo da câmara

O sistema Android possui uma biblioteca, denominada camera2, que permite aceder às características da câmara tais como exposição, abertura e imagem RAW (sem processamento). [Camera2API, 2018]

Um exemplo de uma aplicação que utiliza esta biblioteca é a Open Camera [openCamera, 2018]. Esta é uma aplicação de código aberto para Android que permite o controlo manual sobre as características da câmara de cada dispositivo, como por exemplo a distância focal, ISO, tempo de exposição, temperatura do balanço de brancos (white balance temperature) e aceder à imagem RAW (formato DNG).

No entanto o acesso a estas características depende de cada câmara, visto que nem todas podem permitir os mesmos controlos, devido a limitações de hardware.

Capítulo 5

Redes neuronais

Para a classificação dos planos fotográficos, foram utilizadas as redes convolucionais LeNet 4 e a rede convolucional VGG16.

Neste capítulo é apresentada a descrição de cada uma das arquiteturas utilizadas, assim como os resultados obtidos no treino, terminando com uma comparação entre cada rede, tendo por base testes efetuados com um conjunto de imagens novo.

Na utilização da rede VGG16, foram efetuados testes com a arquitetura da rede original e testes com uma alteração no bloco *fully-connected* da rede.

5.1 LeNet

A rede neuronal utilizada foi a rede LeNet (LeNet4), que é constituída por duas camadas de convolução com uma função de activação "relu", duas camadas de *Max Pooling*, uma camada *fully-connected* com 500 neurónios com uma função de ativação "relu", seguida de outra camada *fully-connected* com 4 neurónios (que correspondem ao número de classes que queremos classificar) com uma função de ativação "*softmax*".

Nas camadas de convolução presentes nesta arquitetura, a 1ª tem 40 filtros de convolução e a 2ª tem 50 filtros, ambos com um filtro de dimensão 5x5.

A Figura 5.1 representa a arquitetura da rede LeNet.

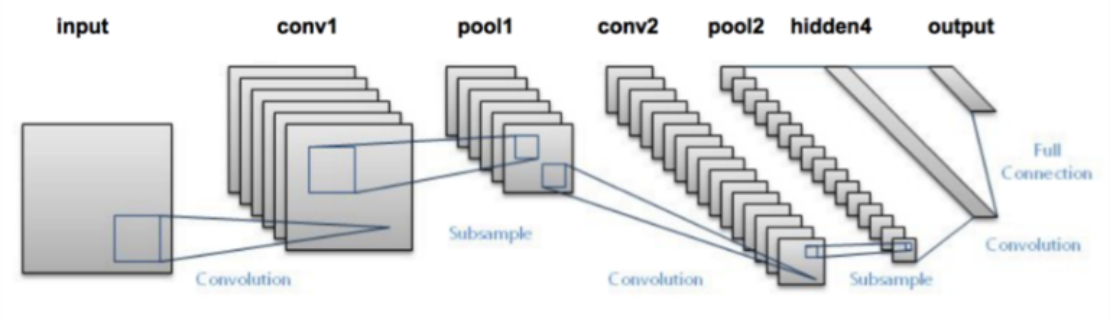


FIGURA 5.1: Arquitectura da rede LeNet [LeNet, 2018]

Originalmente esta rede foi utilizada para detetar dígitos escritos à mão, mas segundo o tutorial "Image classification with Keras and *deep learning*" do site [Rosebrock, 2018b], onde é utilizada a arquitetura desta rede para detetar imagens de pais-natais (detetar se a imagem é ou não um pai natal), foram obtidos resultados bastante precisos. Este tutorial mostra que a rede LeNet pode estender-se a outros tipos diferentes de imagens, tendo sido por esta razão utilizada neste trabalho, para a deteção dos planos fotográficos.

5.2 VGG16

A rede convolucional VGG16 contém 16 camadas (13 camadas de convolução e 3 camadas fully connected) e possui cerca de 138 milhões de parâmetros.

Cada camada de convolução é seguida por uma camada de *Max Pooling* com um fator de 2, i.e. aplica uma redução de dimensão à imagem, diminuindo o seu tamanho para metade.

O número de filtros presentes em cada camada de convolução começa em 64 nas primeiras 2 camadas e vai aumentado até aos 512 filtros nas últimas 6 camadas.

5.3 Modificação da rede VGG16

De forma a usar uma técnica de *transfer learning*, que segundo o livro "Deep Learning with Python - François Chollet" [Chollet, 2017] ajuda a obter melhor precisão, foi implementada uma alteração no bloco *fully-connected* da rede. Esta alteração inclui a utilização de *dropout*.

De forma a perceber até que ponto o *transfer learning* melhora a precisão da rede, esta nova arquitetura foi também treinada sem utilizar *transfer learning*, inicializando todos os seus pesos de forma aleatória.

A tabela 5.1 apresenta a configuração da rede de convolução VGG16 (configuração original) e da rede modificada, sendo vísivel a divisão das camadas que correspondem à base de convolução e ao novo bloco *fully-connected*. Na nova arquitetura, para o caso em que foi utilizado *transfer learning*, as camadas a negrito representam as camadas que foram treinadas (descongeladas), sendo os 4 blocos iniciais de convolução a parte do modelo congelado.

Tabela 5.1: Arquitetura das redes VGG16 utilizadas

VGG16	**VGG16 com novo bloco Dense**	
Conv3-64	Conv3-64	**Bloco de Convolução**
Conv3-64	Conv3-64	
Max Pooling	Max Pooling	
Conv3-128	Conv3-128	
Conv3-128	Conv3-128	
Max Pooling	Max Pooling	
Conv3-256	Conv3-256	
Conv3-256	Conv3-256	
Conv3-256	Conv3-256	
Max Pooling	Max Pooling	
Conv3-512	Conv3-512	
Conv3-512	Conv3-512	
Conv3-512	Conv3-512	
Max Pooling	Max Pooling	
Conv3-512	**Conv3-512**	
Conv3-512	**Conv3-512**	
Conv3-512	**Conv3-512**	
Max Pooling	Max Pooling	
FC-4096	**Dropout(0.5)**	Bloco Fully Connected
FC-4096	**FC-512**	
FC-4	**Dropout(0.5)**	
Soft-Max	**FC-4**	

5.4 Treino das Redes Neuronais

Dado que o data set utilizado era pequeno (4 classes com 1000 imagens cada), foram utilizadas algumas técnicas que permitem obter melhores resultados a nível de desempenho e precisão, tais como data *augmentation*, *dropout* e *transfer*

learning.

Foi utilizada a *Framework* de *deep-learning* Keras que corre por cima de outras bibliotecas de *deep learning* como TensorFlow, Theano ou CNTK [Keras, 2018a].

Foi também utilizada a GPU Titan Xp nos treinos das redes neuronais, o que ajudou a reduzir bastante a duração de cada treino, permitindo desta forma realizar vários testes com diferentes valores de *learning rate*.

Foram efetuados testes com imagens com orientação vertical e horizontal, mas visto que o treino de uma rede neuronal é um processo que demora algum tempo (cerca de 2h cada treino), a grande maioria dos testes incidiu sobre as imagens verticais. Os resultados e análises apresentados em baixo são relativos apenas aos testes com imagens verticais.

Configuração do processo de aprendizagem

Um dos aspetos importantes no treino de uma rede neuronal é a escolha da função de perda e do optimizador.

• Função de perda – Define a distância entre a previsão feita pela rede e o valor real do *output*. Nos problemas de classificação utiliza-se a função "*binary crossentropy*"para um problema de classificação binária (duas classes) e utiliza-se a função "*categorical crossentropy*"para um problema com um número de classes superior a dois.

• Optimizador - Define o método utilizado para a atualização da rede (alteração dos pesos) com base na função de perda. O seu objetivo é encontrar o conjunto de pesos **W** que minimiza a valor da perda [Karpathy, 2018c]. Para tal "implementa uma variante do *stochastic gradient descent* (SGD)", que após calcular o valor da perda procede ao cálculo do seu gradiente (propagando o erro para trás) e atualiza os pesos na direção contrária ao gradiente segundo um valor, denominado *learning rate* [Chollet, 2017]. A Equação 5.1 [Chollet, 2017] representa um exemplo da atualização dos pesos.

$$W = W - (learning\_rate * gradiente) \tag{5.1}$$

Um valor de *learning rate* muito pequeno leva à necessidade de um maior número de iterações para atingir o valor mínimo da função de perda, aumentando também o risco de convergir para mínimos locais (como ilustrado na Figura 5.2 [Raschka, 2015]). Por outro lado, escolhendo um valor muito elevado pode levar a que a rede nunca chegue a convergir, nunca atingido o mínimo [Raschka, 2015]. (Ver Figura 5.2 [Raschka, 2015])

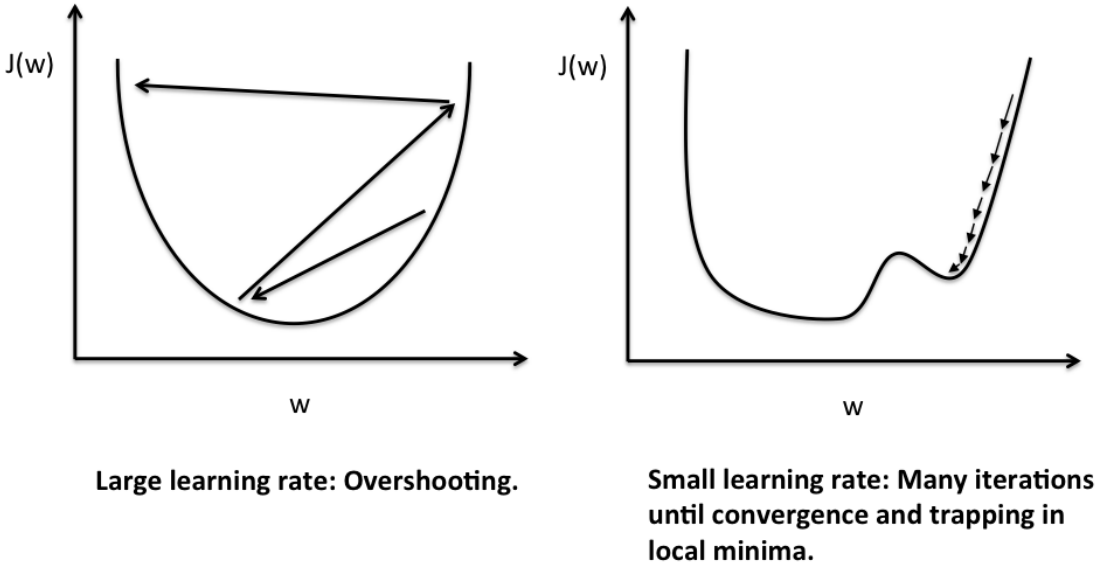


FIGURA 5.2: Gradient descent com um valor de *learning rate* grande (direita) e pequeno (esquerda)

Configuração dos treinos

Para o treino de todas as redes foi utilizado um *learning rate* de valor 10, 8, 4, 2 e 1 os quais foram divididos por 10 cinco vezes, de forma a chegar ao valor que melhor se adequa ao problema em questão, i.e. o valor que permite atingir um maior valor de precisão. Foram utilizados os optimizadores SGD e o ADAM, e foi usada a função "*categorical_ crossentropy*" como função de perda (*Loss Function*) dado que o problema a classificar tem mais que 2 classes [Chollet, 2017]. Cada treino "durou" 25 *epochs*, onde cada *epoch* representa uma iteração sobre todo o data set [Keras, 2018d]. O data set utilizado, composto por 4000 imagens (1000

para cada plano), foi dividido em 70% para o treino das redes e 30% para a validação do modelo (teste).

Em seguida são apresentados os gráficos do treino de cada rede, que mostram a variação do valor de precisão (eixo vertical) da rede com a variação do *learning rate* (eixo horizontal). Onde o valor de precisão da rede, valor entre 0 e 1, corresponde ao rácio entre o número de imagens classificadas corretamente e o número total de imagens classificadas [Keras, 2018b] [Developers, 2018].

A apresentação dos gráficos está estruturada da seguinte forma:

- Resultados LeNet;

- Resultados VGG16;

- Resultados VGG16 com novo bloco dense;

- Resultados VGG16 com novo bloco dense usando transfer learning.

A azul observamos o valor da precisão obtida durante o treino da rede (Acc) e a laranja o valor da precisão obtida no teste (Val_acc), este último é o valor real da precisão do modelo.

Quanto maior for a diferença entre a precisão obtida no treino e a precisão obtida no teste, maior a probabilidade de nos encontrarmos numa situação em que ocorreu *overfit*.

Resultados LeNet

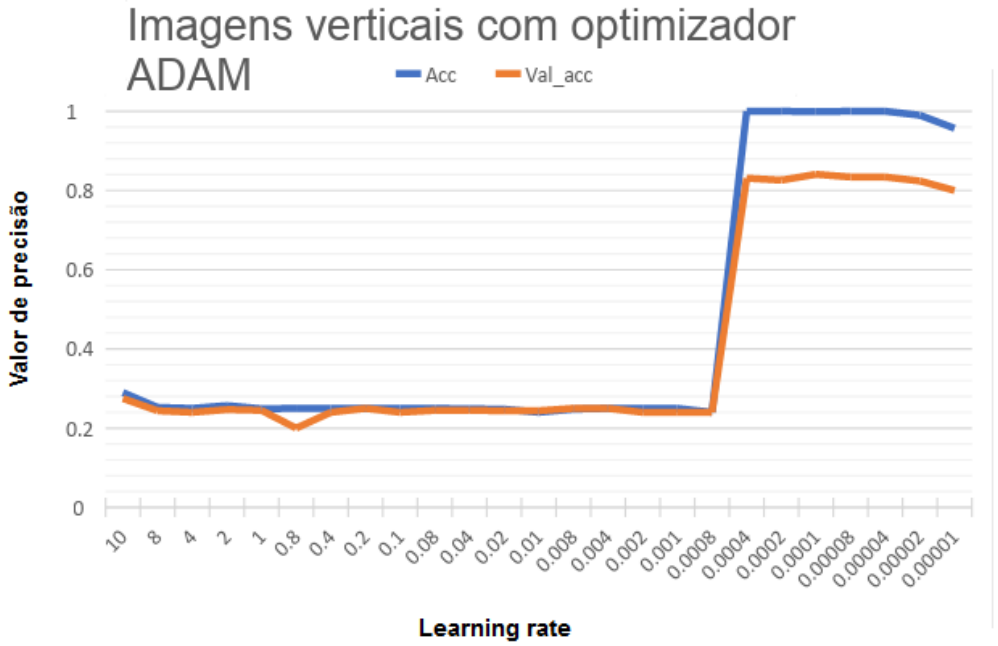


FIGURA 5.3: Evolução do valor de precisão obtida pela rede LeNet no treino com imagens Verticais utilizando o optimizador ADAM

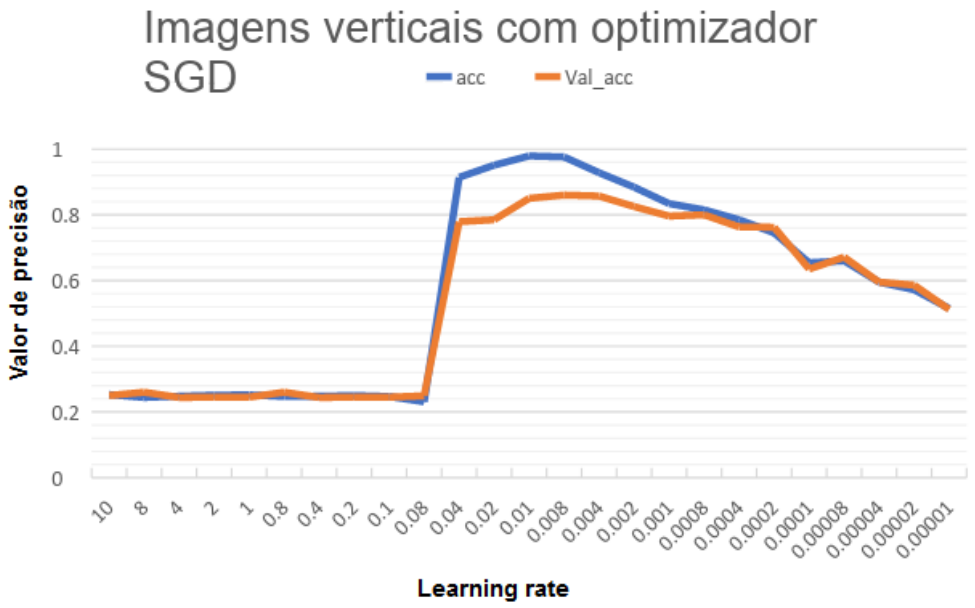


FIGURA 5.4: Evolução do valor de precisão obtida pela rede LeNet no treino com imagens Verticais utilizando o optimizador SGD

Resultados VGG16

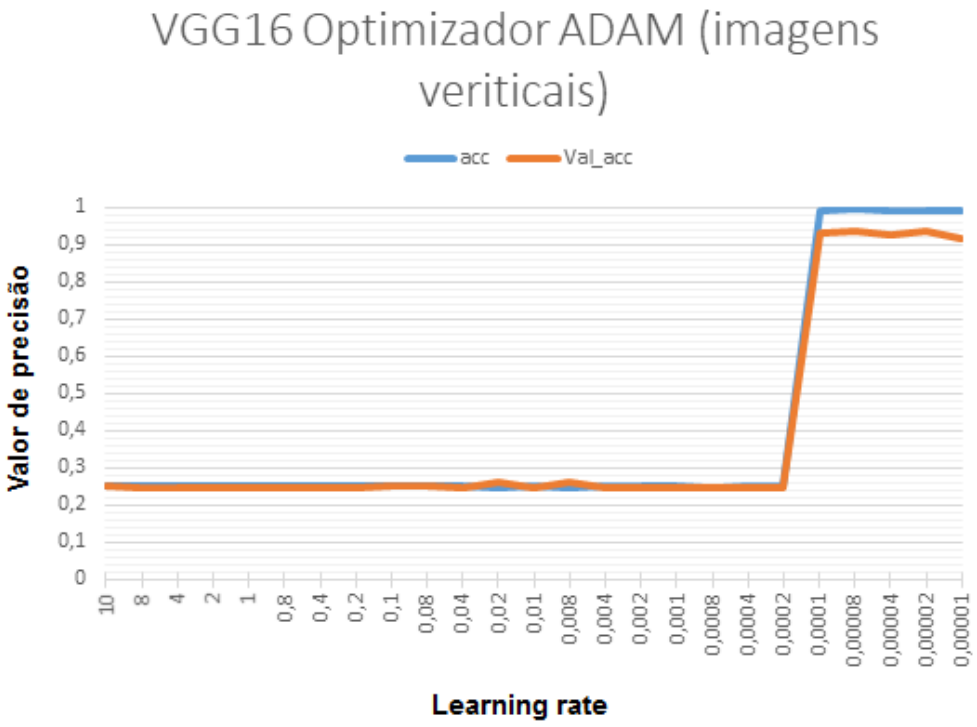


FIGURA 5.5: Evolução do valor de precisão obtida pela rede VGG16 no treino com imagens Verticais utilizando o optimizador ADAM

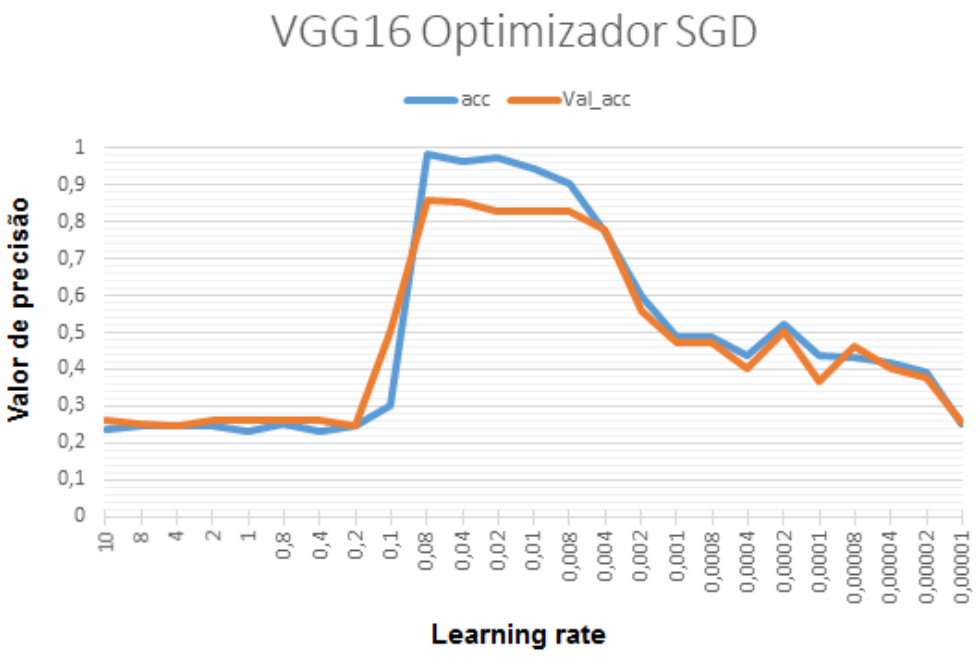


FIGURA 5.6: Evolução do valor de precisão obtida pela rede VGG16 no treino com imagens Verticais utilizando o optimizador SGD

Resultados VGG16 com novo bloco dense

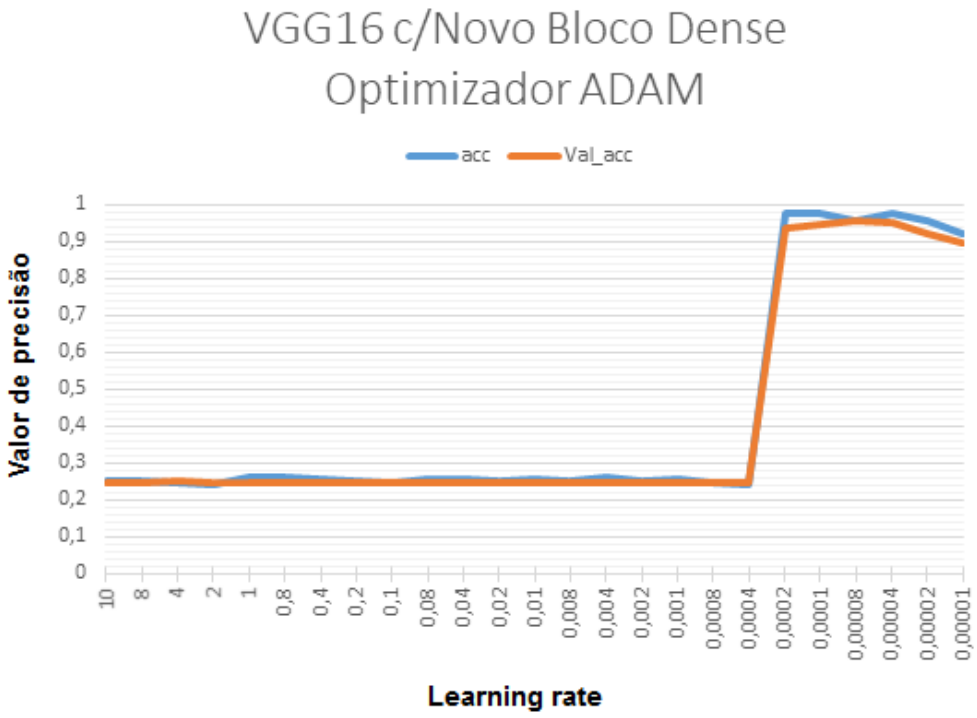


FIGURA 5.7: Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense no treino com imagens Verticais utilizando o optimizador ADAM

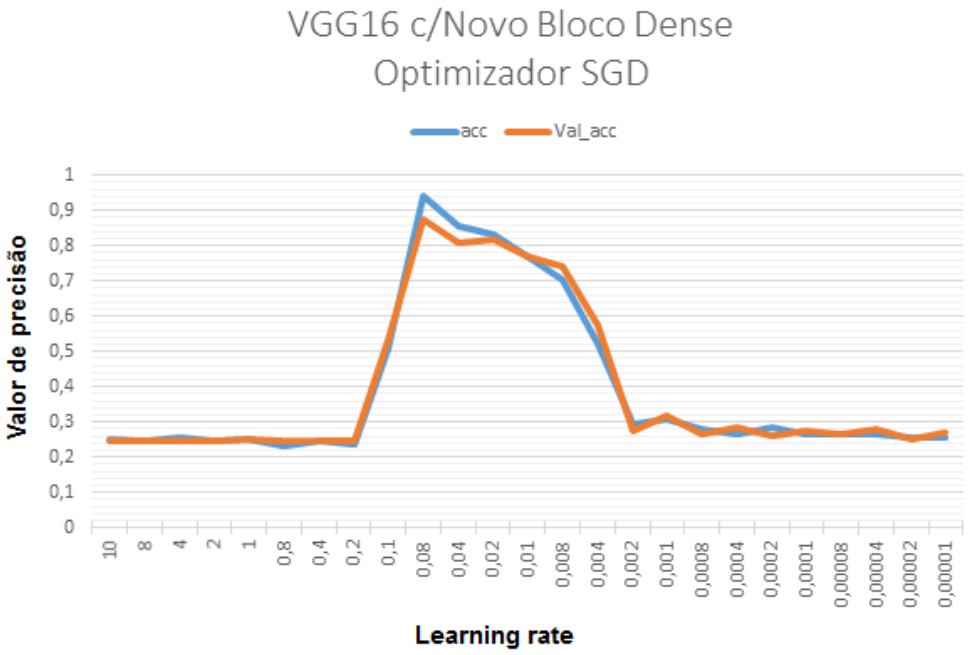


FIGURA 5.8: Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense no treino com imagens Verticais utilizando o optimizador SGD

Resultados VGG16 com novo bloco dense usando transfer learning

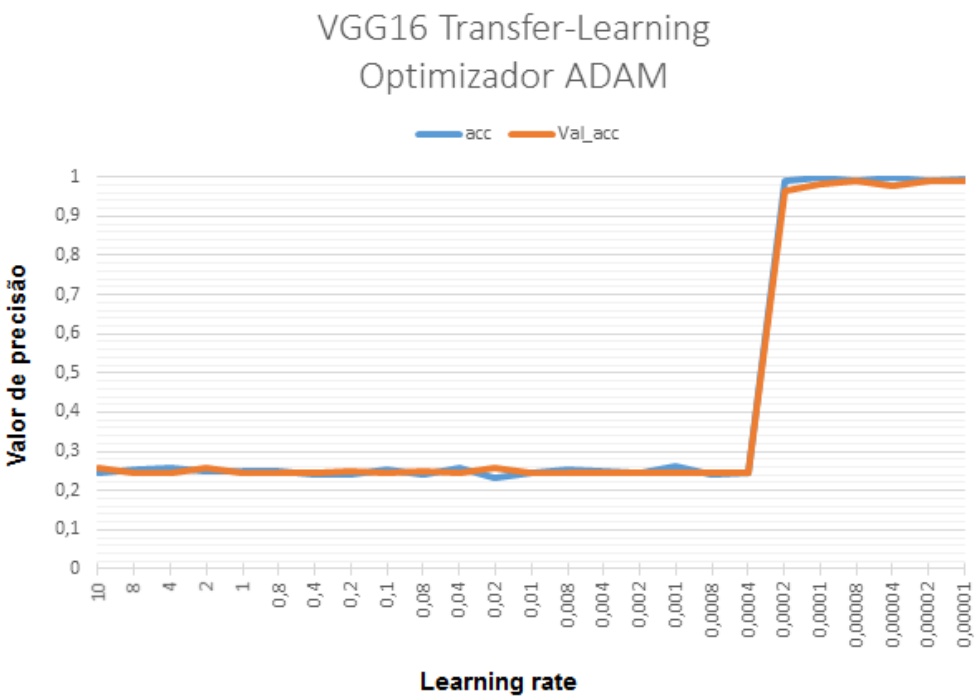


FIGURA 5.9: Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense usando transfer learning no treino com imagens Verticais utilizando o optimizador ADAM

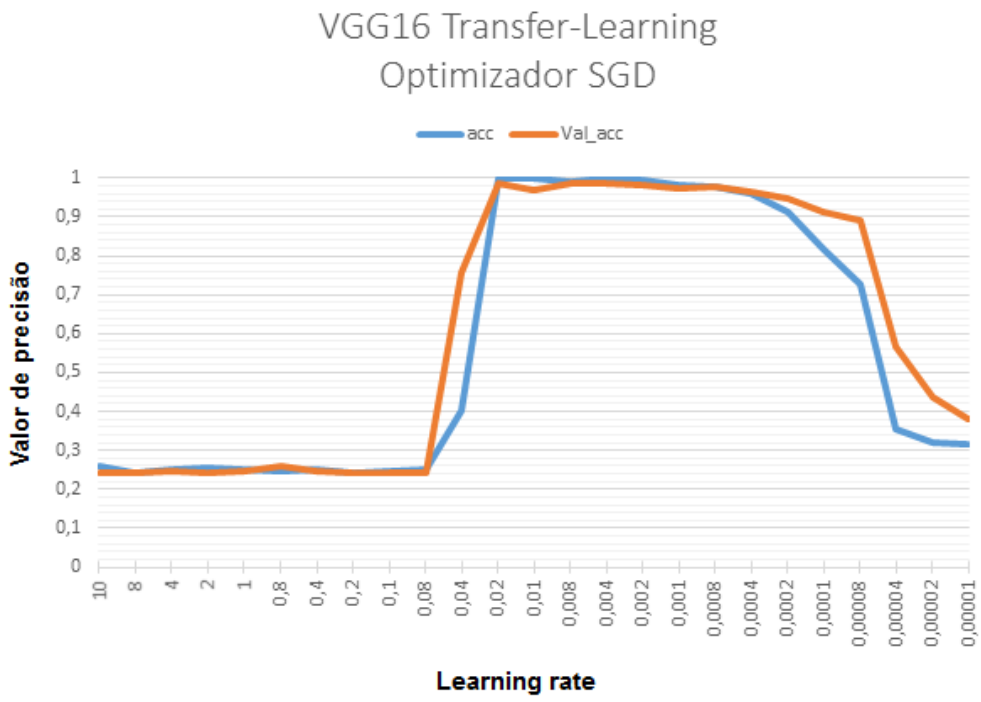


FIGURA 5.10: Evolução do valor de precisão obtida pela rede VGG16 c/Novo Bloco Dense usando transfer learning no treino com imagens Verticais utilizando o optimizador SGD

5.4.1 Discussão dos resultados

Observando os gráficos verifica-se que para um certo valor de *learning rate* existe um "salto"positivo no valor de precisão. Isto significa que este é o valor máximo de *learning rate* que se pode utilizar, dado que para valores mais elevados o treino nunca chega a convergir [Surmenok, 2017].

Observa-se também que utilizando o optimizador ADAM, o treino da rede só começa a convergir com valores de *learning rate* muito baixos, sendo que a utilização do mesmo nas redes VGG16 resultou em valores de precisão mais elevados.

Apesar disto, os melhores resultados obtidos com a rede LeNet foram com o optimizador SGD, onde o valor de precisão mais elevado com esta rede foi de 86% (precisão no teste).

5.5 Comparação das redes

Tabela 5.2: Comparação das CNN's pelo número de parâmetros

Redes	**Parâmetros Treináveis**	**Parâmetros Não-Treináveis**	**Nº Total de Parâmetros**
LeNet	187.555.594	0	187.555.594
VGG16	258.008.900	0	258.008.900
VGG16 com novo bloco dense	43.028.804	0	43.028.804
VGG16 com novo bloco dense (usando transfer learning)	35.393.540	7.635.264	43.028.804

Dado que a diferença entre o número de parâmetros das redes VGG16 com novo *bloco dense*, a rede VGG16 e a rede LeNet, é muito grande, foram efectuados treinos mais prolongados (50 epochs) nas redes VGG16 e LeNet, de forma a verificar se o valor de precisão aumentava.

Para estes treinos foram apenas utilizados os valores de *learning rate* que obtiveram maior valor de precisão nos treinos com 25 epochs.

Na rede VGG16 verificou-se um aumento de até 3,6% com o optimizador SGD, passando de 0,859 para 0,895 e um aumento de até 2% com o optimizador ADAM, passando de 0,928 para 0,945. A Tabela 5.3 apresenta a maior precisão de teste obtida em cada uma das redes, sendo que alguns valores foram apenas obtidos nos treinos mais prolongados, com 50 epochs.

Tabela 5.3: Precisão obtida por cada rede

Redes	**Optimizador**	
	SGD	**ADAM**
LeNet	86%	84%
VGG16	89,5%	94,5%
VGG16 com novo bloco dense	87,7%	95,6%
VGG16 com novo bloco dense (usando transfer learning)	98,8%	99%

5.5.1 Testes Reais

De forma a testar verdadeiramente a eficácia dos modelos treinados, foram efetuados testes, apenas para os modelos que obtiveram maior precisão em cada rede, com várias imagens estáticas de cada plano. Este data set inclui as imagens utilizadas no treino da rede assim como novas imagens de cada plano fotográfico que nunca foram "vistas"pelas redes, contabilizando em média o dobro das imagens usadas no treino das redes.

Em seguida são apresentados os resultados obtidos para as imagens de cada plano, onde o campo "Modelo" descreve a rede, o valor de *learning rate* (Lr) e o optimizador utilizados, e o campo "Previsões falhadas" é dado pelo número de imagens que o modelo classificou de forma errada (i.e. classificou a imagem como pertencente a outro plano).

Os modelos "VGG16 NovaDense"e "VGG16 NovaDense c/Transfer Learning", correspondem às redes "VGG16 com novo bloco dense"e "VGG16 com novo bloco dense (usando transfer learning)"respetivamente.

5.5.1.1 Resultados obtidos com Imagens Verticais

Plano Médio

Tabela 5.4: Resultados obtidos na deteção de imagens com orientação vertical do plano médio

Nº Imagens Processadas: 2443		
Modelo	**Previsões falhadas**	**% Precisão**
LeNet_Lr0.008_SGD	363	85,1%
LeNet_Lr0.004_SGD	352	85,6%
VGG16 Lr0.00008 ADAM	154	93,7%
VGG16 Lr0.00004 ADAM	150	93,9%
VGG16 NovaDense Lr0.00008 ADAM	88	96,4%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	12	99,5%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	10	99,6%

Plano Close-Up

Tabela 5.5: Resultados obtidos na deteção de imagens com orientação vertical do plano close-up

Nº Imagens Processadas: 1547		
Modelo	**Previsões falhadas**	**% Precisão**
LeNet_Lr0.008_SGD	99	93,6%
LeNet_Lr0.004_SGD	100	93,5%
VGG16 Lr0.00008 ADAM	35	97,7%
VGG16 Lr0.00004 ADAM	46	97%
VGG16 NovaDense Lr0.00008 ADAM	55	96,4%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	19	98,8%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	19	98,8%

Plano Big Close-Up

Tabela 5.6: Resultados obtidos na deteção de imagens com orientação vertical do plano big close-up

Nº Imagens Processadas: 1272		
Modelo	**Previsões falhadas**	**% Precisão**
LeNet_Lr0.008_SGD	61	95,2%
LeNet_Lr0.004_SGD	55	95,7%
VGG16 Lr0.00008 ADAM	44	96,5%
VGG16 Lr0.00004 ADAM	34	97,3%
VGG16 NovaDense Lr0.00008 ADAM	39	96,9%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	8	99,4%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	5	99,6%

Plano Não

Tabela 5.7: Resultados obtidos na deteção de imagens com orientação vertical do plano não

Nº Imagens Processadas: 2507		
Modelo	**Previsões falhadas**	**% Precisão**
LeNet_Lr0.008_SGD	332	86,8%
LeNet_Lr0.004_SGD	446	82,2%
VGG16 Lr0.00008 ADAM	184	92,7%
VGG16 Lr0.00004 ADAM	181	92,8%
VGG16 NovaDense Lr0.00008 ADAM	214	91,5%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	52	97,9%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	57	97,7%

No geral verifica-se que a rede VGG16 NovaDense usando Transfer Learning obteve os melhores resultados, atingindo uma precisão muito próxima dos 100% em todos os planos.

Podemos observar também que apesar de no treino a rede VGG16 NovaDense, com o optimizador ADAM, ter obtido um valor de precisão ligeiramente superior ao da rede VGG16, nestes testes verificou-se o contrário.

5.5.1.2 Percentagem de confiança com Imagens Verticais

As Tabelas 5.8 a 5.11 apresentam as percentagens de confiança das imagens de cada plano que foram corretamente classificadas pelos modelos, sendo que o campo "Confiança Mínima"representa a menor percentagem com que uma imagem foi corretamente classificada pelo modelo.

Plano Médio

TABELA 5.8: Percentagem de confiança obtida nas imagens do plano médio

	Nº de Imagens com confiança >80%	**Nº de Imagens com confiança entre 80% e 50%**	**Nº de Imagens com confiança <50%**	**Confiança Mínima**
LeNet_Lr0.008_SGD	1894	178	8	40,9%
LeNet_Lr0.004_SGD	1824	237	30	31,1%
VGG16 Lr0.00008 ADAM	2254	34	1	49,7%
VGG16 Lr0.00004 ADAM	2270	23	0	50%
VGG16 NovaDense Lr0.00008 ADAM	1218	505	632	0,21%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	1163	335	933	0,00365%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	1210	352	871	0,02%

Plano Close-Up

Tabela 5.9: Percentagem de confiança das imagens do plano close-up

	Nº de Imagens com confiança >80%	**Nº de Imagens com confiança entre 80% e 50%**	**Nº de Imagens com confiança <50%**	**Confiança Mínima**
LeNet_Lr0.008_SGD	1378	64	6	37,26%
LeNet_Lr0.004_SGD	1350	85	12	37%
VGG16 Lr0.00008 ADAM	1502	10	0	50%
VGG16 Lr0.00004 ADAM	1492	9	0	50%
VGG16 NovaDense Lr0.00008 ADAM	724	296	472	0,057%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	851	181	496	0,0045%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	756	253	519	0,029%

Plano Big Close-Up

Tabela 5.10: Percentagem de confiança das imagens do plano big close-up

	Nº de Imagens com confiança >80%	**Nº de Imagens com confiança entre 80% e 50%**	**Nº de Imagens com confiança <50%**	**Confiança Mínima**
LeNet_Lr0.008_SGD	1167	38	6	32,2%
LeNet_Lr0.004_SGD	1166	46	5	42,8%
VGG16 Lr0.00008 ADAM	1223	5	0	50%
VGG16 Lr0.00004 ADAM	1230	8	0	50%
VGG16 NovaDense Lr0.00008 ADAM	944	106	183	0,09%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	1015	93	156	0,00212%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	1096	72	99	0,012%

Plano Não

Tabela 5.11: Percentagem de confiança das imagens do plano não

	Nº de Imagens com confiança >80%	**Nº de Imagens com confiança entre 80% e 50%**	**Nº de Imagens com confiança <50%**	**Confiança Mínima**
LeNet_Lr0.008_SGD	1965	186	24	40,4%
LeNet_Lr0.004_SGD	1749	290	22	37%
VGG16 Lr0.00008 ADAM	2264	56	3	44,1%
VGG16 Lr0.00004 ADAM	2291	34	1	48,05%
VGG16 NovaDense Lr0.00008 ADAM	273	485	1535	0,05%
VGG16 NovaDense c/Transfer Learning Lr0.00002 ADAM	1874	253	328	0,00395%
VGG16 NovaDense c/Transfer Learning Lr0.00001 ADAM	1856	254	340	0,002%

Observa-se que os modelos VGG16 e LeNet classificaram as imagens com maior confiança do que os modelos VGG16 NovaDense e VGG16 NovaDense c/Transfer Learning, além de que o valor de confiança mínimo com que classificaram corretamente as imagens foi de 32%, muito superior aos modelos VGG16 NovaDense e VGG16 NovaDense c/Transfer Learning que classificaram com 0.003% e 0.002%.

5.5.1.3 Imagens Horizontais

Com base nos modelos treinados com imagens verticais que obtiveram maior precisão, foram efetuados treinos das CNN's com imagens horizontais e efetuada a mesma análise para essas redes. A Tabela 5.12 mostra os modelos usados, assim como a sua configuração.

Tabela 5.12: Modelos Horizontais Utilizados

Modelo
LeNet Lr0.008 Epochs50 SGD
VGG16 Lr0.00004 Adam
VGG16 NovaDense Lr0.00008 Adam
VGG16 NovaDense c/Transfer Learning Lr0.0001 Adam
VGG16 NovaDense c/Transfer Learning Lr0.0002 Adam

O modelo que obteve melhores resultados foi o modelo VGG16 NovaDense c/Transfer Learning Lr0.0002 Adam, com uma média de 99% de precisão em todos os planos (97% no Plano Não e 99% de precisão nos restantes).

5.6 Discussão

Apesar das imagens classificadas pelo modelo LeNet terem sido classificadas com grande confiança (sendo 32% a menor confiança com que uma imagem foi classificada), a diferença de precisão entre este modelo e os modelos VGG16 é bastante grande, visto que este modelo tem apenas 86% de precisão comparado com 94.5%, 95.6% e 99% dos modelos VGG16 e como se pode ver pela análise feita, foi o modelo que mais falhou ao classificar as imagens. Uma razão para este sucedido deve-se ao facto do modelo possuir poucas camadas de convolução o que é um fator importante para atingir melhores resultados [Cireşan et al., 2012].

Comparando as redes VGG16 observamos que a rede original classifica as imagens com grande confiança situando-se em média sempre perto dos 50%. Em

termos de precisão esta rede e a rede VGG16 NovaDense são bastante semelhantes, sendo que a grande diferença foca-se no número de parâmetros de cada uma, onde a rede original perde por possuir quase 6 vezes mais parâmetros, o que se resume numa maior necessidade de processamento e de tempo gasto tanto no treino como na execução do modelo.

Apesar disso a rede que obteve mais sucesso é a rede VGG16 NovaDense que utiliza *transfer learning* pois é a rede que classifica corretamente mais imagens.

Capítulo 6

Algoritmos de deteção facial e deteção de olhos

Neste capítulo são apresentados os resultados dos algoritmos de deteção facial utilizados e os resultados obtidos com o algoritmo Haar Cascades na deteção de olhos.

No capítulo 6.2 é também apresentada a análise feita à localização dos olhos nas imagens, de forma a encontrar uma tolerância entre a posição dos olhos e os pontos de ouro.

Para os testes dos algoritmos foi elaborado um *script* em *Python*, o qual deteta as caras das pessoas, assim como os olhos e a sua posição/localização na fotografia.

6.1 Resultados da deteção facial

Foram testados 3 algoritmos de deteção facial diferentes, Haar Cascades, LBP Cascades e HOG (histograma de gradientes orientados), apenas com imagens dos planos médio e *close-up*, com o objetivo de descobrir qual o mais preciso.

São utilizados os algoritmos pré-treinados haarcascade_frontalface_default e lbpcascade_frontalface_improved disponibilizados pela biblioteca OpenCV, passando a ser denominados no seguimento deste trabalho apenas por Haar cascades default e LBP Improved respectivamente. O algoritmo HOG pré-treinado utilizado, foi disponibilizado pela biblioteca DLIB.

As Tabelas 6.1 e 6.2 apresentam os resultados obtidos (dados por número de imagens) para as imagens de cada plano, onde o campo "Algoritmo/Scale" representa o algoritmo usado e a percentagem com que a imagem é reduzida, o campo "Caras detetadas mas mal localizadas"é dado pelo número de imagens onde foram detetadas caras em zonas que não continham faces e o campo "TC", correspondente à taxa de caras corretamente detetadas e localizadas, é dado pela percentagem do número de imagens onde foram corretamente detetadas e localizadas caras em relação ao número total de imagens.

Plano Close-Up

Tabela 6.1: Resultados dos algoritmos de deteção facial para as imagens do plano close-up utilizando 591 imagens

Nº de imagens: 591				
Algoritmo/Scale	**Caras detectadas e bem localizadas**	**Caras detectadas mas mal localizadas**	**Caras não detectadas**	**TC**
Haar cascades Default 1.1	462	57	72	78%
Haar cascades Default 1.3	382	31	178	65%
LBP Improved 1.1	538	2	51	91%
LBP Improved 1.3	468	0	123	79%
HOG	553	1	37	94%

Plano Médio

Tabela 6.2: Resultados dos algoritmos de deteção facial para as imagens do plano médio utilizando 835 imagens

Nº de imagens: 835				
Algoritmo/Scale	**Caras detectadas e bem localizadas**	**Caras detectadas mas mal localizadas**	**Caras não detectadas**	**TC**
Haar cascades Default 1.1	722	88	25	86%
Haar cascades Default 1.3	677	22	136	81%
LBP Improved 1.1	687	0	148	82%
LBP Improved 1.3	528	1	306	63%
HOG	809	4	22	97%

Observa-se que o algoritmo que obteve maior taxa na deteção facial foi o HOG atingindo uma taxa de deteção 94% e 97%. Em segundo lugar observa-se o algoritmo LBP Improved 1.1 e o Haar cascades Default 1.1 com uma taxa de deteção no plano *close-up* de 91% e 78% respetivamente e com uma taxa de deteção no plano médio de 82% e 86%, respetivamente.

De salientar que de entre estes 3 algoritmos o Haar cascades Default 1.1 é o que possui um número mais elevado de "caras detetadas mas mal localizadas".

Foi também feita uma análise ao tempo de processamento de cada um destes 3 algoritmos, descrito na Tabela 6.3, onde cada cada tempo de processamento corresponde ao tempo que cada algoritmo necessitou para processar todo o data set das imagens do plano médio. Esta análise foi efetuada num computador com as seguintes características:

- CPU: Intel(R) Core(TM) i5-2430M 2.40GHz

- RAM: 4GB
- Disco SSD

Tabela 6.3: Tempo de processamento dos algoritmos de deteção facial no data set do plano médio utilizando 835 imagens

Nº de imagens: 835	
Algoritmo/Scale	**Tempo de Processamento**
Haar cascades Default 1.1	150ms
LBP Improved 1.1	102ms
HOG	446ms

Observa-se que o algoritmo LBP Improved 1.1 é o mais rápido a detetar caras, sendo que o algoritmo HOG tem um tempo de processamento 4 vezes mais lento.

6.2 Resultados do algoritmo de deteção de olhos

Para a deteção dos olhos foi utilizado um algoritmo Haar Cascades pré-treinado disponibilizado na biblioteca OpenCV denominado haarcascade_eye.

Foram efetuados testes com este algoritmo tanto nas imagens do plano *close-up* como no plano médio. As tabelas 6.4 e 6.5 apresentam os resultados obtidos (por número de imagens), onde o campo "Olhos detetados mas mal localizados"é dado pelo número de imagens onde foram detetados olhos em zonas que não correspondiam a olhos e o campo "TO", que corresponde à taxa de imagens onde foram detetados e localizados olhos corretamente, é dado pela percentagem do número de imagens onde foram corretamente detetados e localizados olhos face ao número total de imagens. Considera-se a deteção dos olhos correta se todos os olhos presentes na fotografia forem detetados.

Plano Close-Up

Tabela 6.4: Resultados do algoritmo haarcascade_eye na deteção de olhos nas imagens do plano close-up utilizando 133 imagens

Nº Imagens: 133	**Olhos detetados e bem localizados**	**Olhos detetados mas mal localizados**	**Olhos não detetados**	**TO**
haarcascade_eye	103	20	10	77%

Plano Médio

Tabela 6.5: Resultados do algoritmo haarcascade_eye na deteção de olhos nas imagens do plano médio utilizando 170 imagens

Nº Imagens: 170	**Olhos detetados e bem localizados**	**Olhos detetados mas mal localizados**	**Olhos não detetados**	**TO**
haarcascade_eye	92	17	61	54%

Através da deteção dos olhos foi possível chegar a um valor de tolerância entre a posição dos olhos e os pontos de ouro.

Para isso foi desenvolvido um *script* que determinava o ponto central da região onde tinha sido detetado o olho e em seguida era feita a comparação da sua localização com os pontos de ouro.

Até à data este teste foi apenas efetuado para o plano *close up*.

Ao longo do eixo horizontal de forma a calcular a coordenada dos pontos de ouro à esquerda da imagem, divide-se o comprimento desta por 3, para calcular os pontos de ouro à direita divide-se o comprimento por 1.5.

Desta forma sabendo a localização dos olhos na imagem e dividindo o comprimento da imagem pela coordenada dos olhos no eixo horizontal obtém-se a fração da imagem onde se localiza o olho (sendo que o ideal seria esta fração ser igual a 3 ou a 1.5)

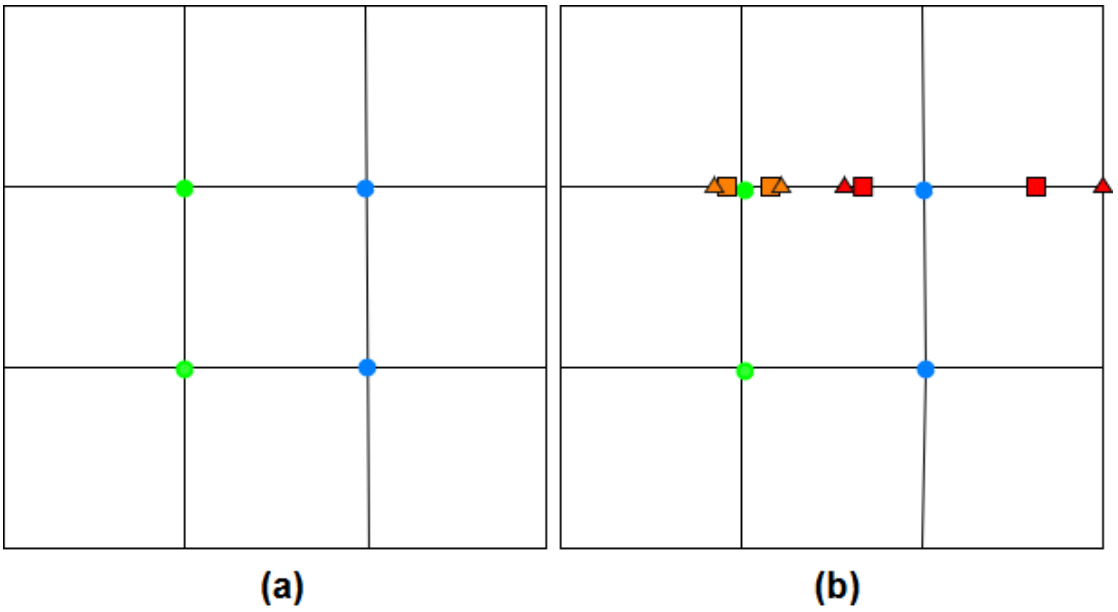


FIGURA 6.1: Representação dos pontos de ouro em (a) e do valor de tolerância de 0.5 e 0.35 aos pontos de ouro em (b), representados pelos pontos vermelhos e laranja respetivamente.

Os pontos verdes e azuis representam os pontos de ouro à esquerda e à direita da imagem respetivamente, sendo os pontos verdes calculados no eixo horizontal por $\frac{1}{3} \times comprimento$ ou $\frac{comprimento}{3}$ e os pontos azuis calculados por $\frac{2}{3} \times comprimento$ ou $\frac{comprimento}{1.5}$. Os triângulos representam o valor de tolerância de 0.5 até ao qual se verificava a maior ocorrência dos olhos nas imagens, os quadrados representam o valor de tolerância de 0.35. As cores laranja e vermelho na imagem representam os valores de tolerância ao ponto de ouro esquerdo e direito, respetivamente.

A Figura 6.2 representa a ocorrência das frações da imagem onde se localizam os olhos, apenas ao longo do eixo horizontal.

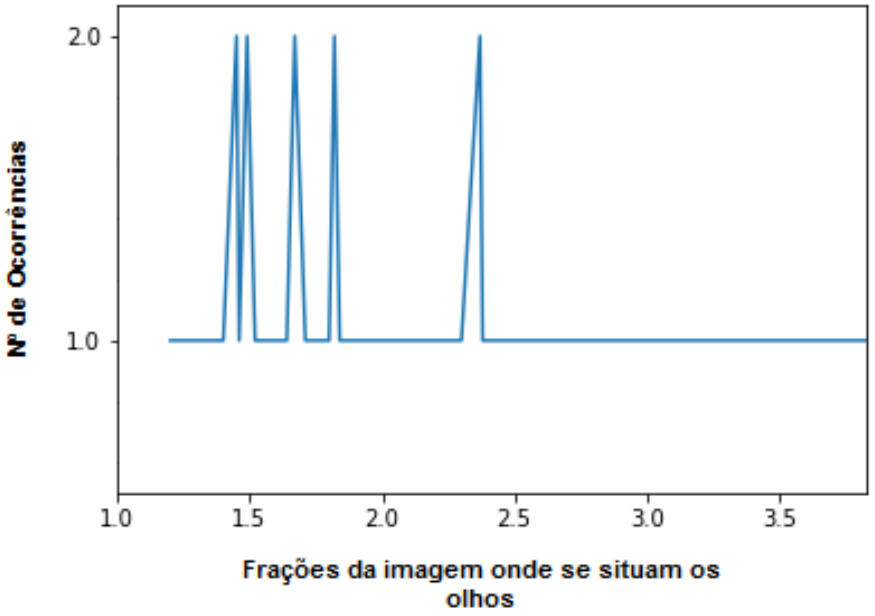


FIGURA 6.2: Histograma das frações da imagem ao longo do eixo horizontal onde se localizam os olhos

O eixo horizontal representa o valor pela qual é dividido o comprimento da imagem e o eixo vertical representa a sua ocorrência.

Por análise ao gráfico observa-se que os olhos se localizam a uma fração mínima de 1.2 e a uma fração máxima de 3.83 do comprimento da imagem.

Observa-se que no data set utilizado a maioria dos olhos estão localizados perto do ponto de ouro mais à direita (a sua posição é calculada por $\frac{comprimento}{1.5}$), no entanto para o cálculo do valor da tolerância, não é tida em conta esta diferenciação. Ex: Uma imagem contém um olho localizado na posição $\frac{comprimento}{1.15}$ e noutra imagem um dos olhos localiza-se na posição $\frac{comprimento}{2.65}$, ambos se situam a 0.35 de um ponto de ouro. Foi observado que a localização dos olhos situada a uma tolerância dos pontos de ouro de:

- 0.2 englobava 34% do data set;
- 0.35 englobava 54.5% do data set;
- 0.5 englobava 77.3% do data set;
- 0.65 englobava 90.9% do data set;

Na Figura 6.3 é apresentado o gráfico da ocorrência das frações da imagem onde se localizam os olhos ao longo do eixo vertical.

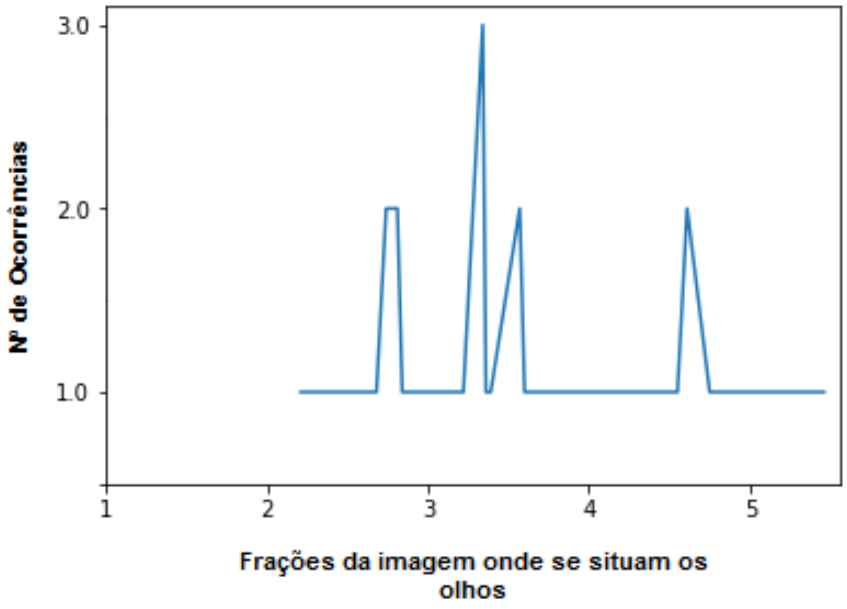


FIGURA 6.3: Histograma das frações da imagem ao longo do eixo vertical onde se localizam os olhos

Analisando o gráfico observa-se que os olhos se localizam a uma fração mínima de 2.21 e a uma fração máxima de 5.46 da altura da imagem.

Observa-se que a distância vertical entre a posição dos olhos e os pontos de ouro é bastante maior que a distância horizontal. Isto deve-se ao facto das fotografias presentes no data set variarem no sentido em que umas têm o corte ao nível da testa e outras possuem o corte mais acima, contendo a cabeça toda.

Observa-se também que a localização dos olhos situada a uma tolerância dos pontos de ouro de:

- 0.2 inclui 20.5% do data set;
- 0.35 inclui 43.2% do data set;
- 0.5 inclui 50% do data set;
- 0.65 inclui 59.1% do data set;

6.3 Testes Reais na aplicação Android

Foram efectuados testes com os algoritmos de deteção facial e deteção de olhos na aplicação Android.

Com base na regra dos terços e na localização dos olhos do sujeito a ser fotografado o algoritmo apresenta uma seta a vermelho indicando para que lado o fotógrafo deve mover a câmara fotográfica, neste caso o smartphone.(Ver Figuras 6.4, 6.5, 6.6 e 6.7)

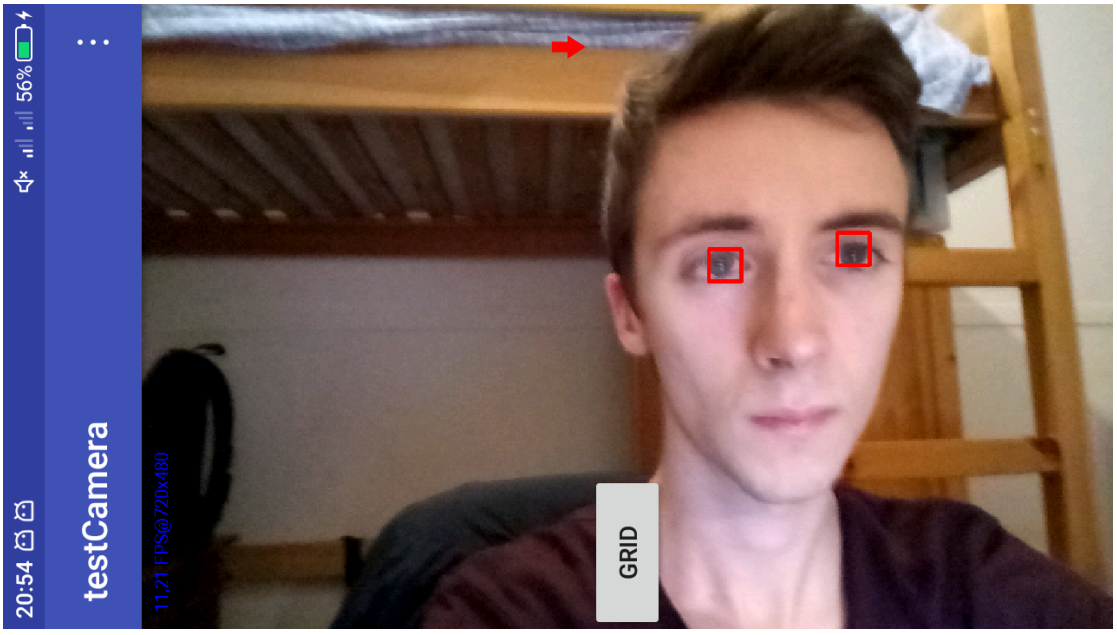


FIGURA 6.4: Teste do algoritmo que fornece sugestões relativo ao enquandramento

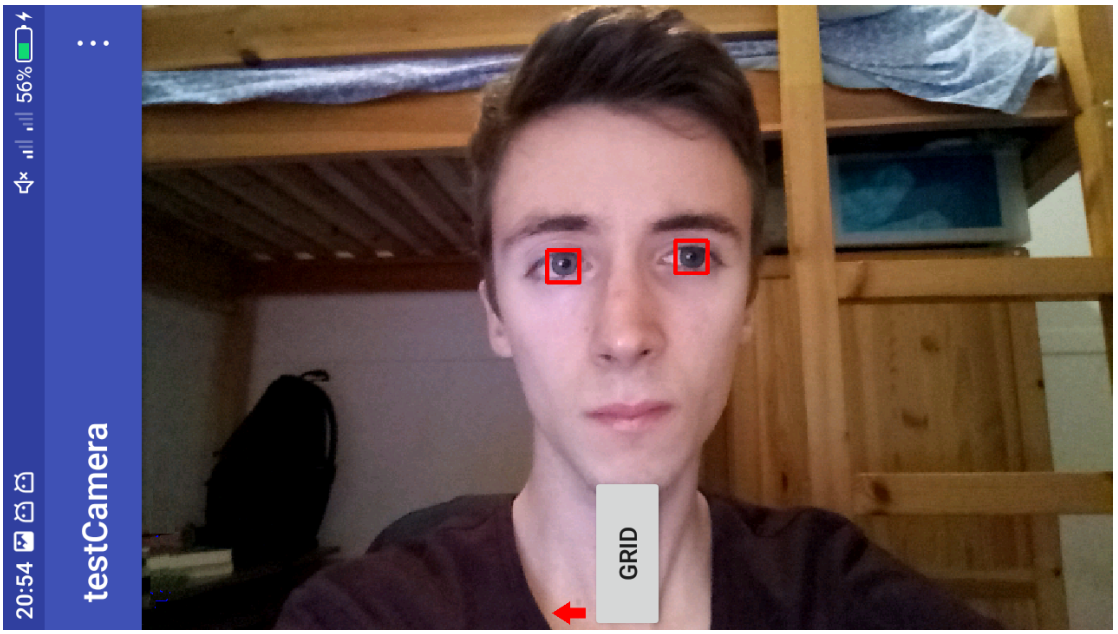


FIGURA 6.5: Teste do algoritmo que fornece sugestões relativo ao enquandramento

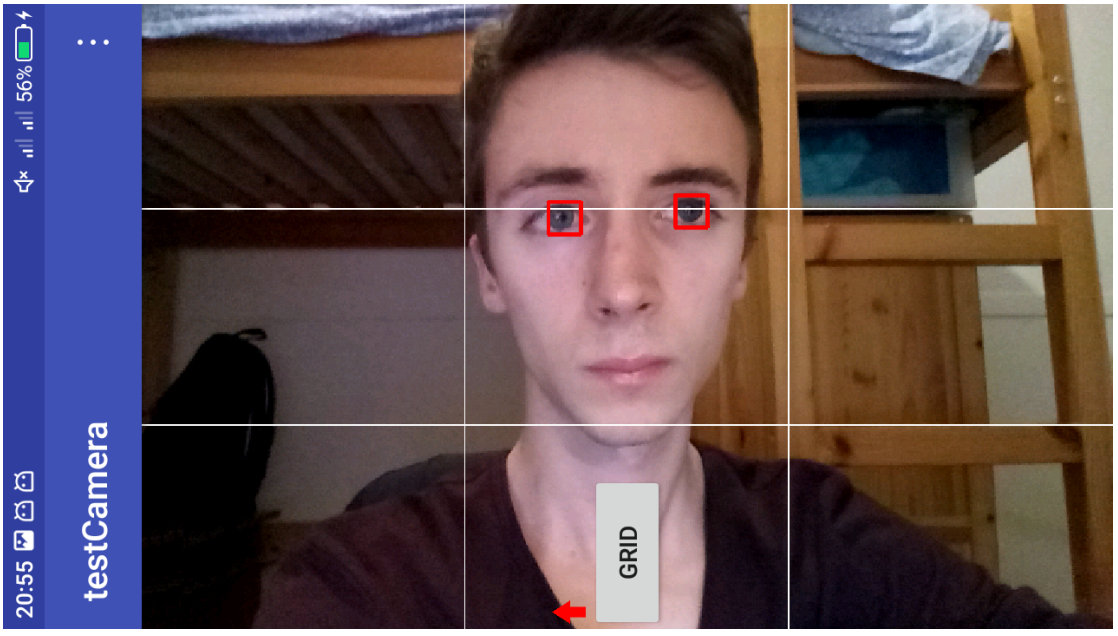


FIGURA 6.6: Teste do algoritmo visualizando a grelha

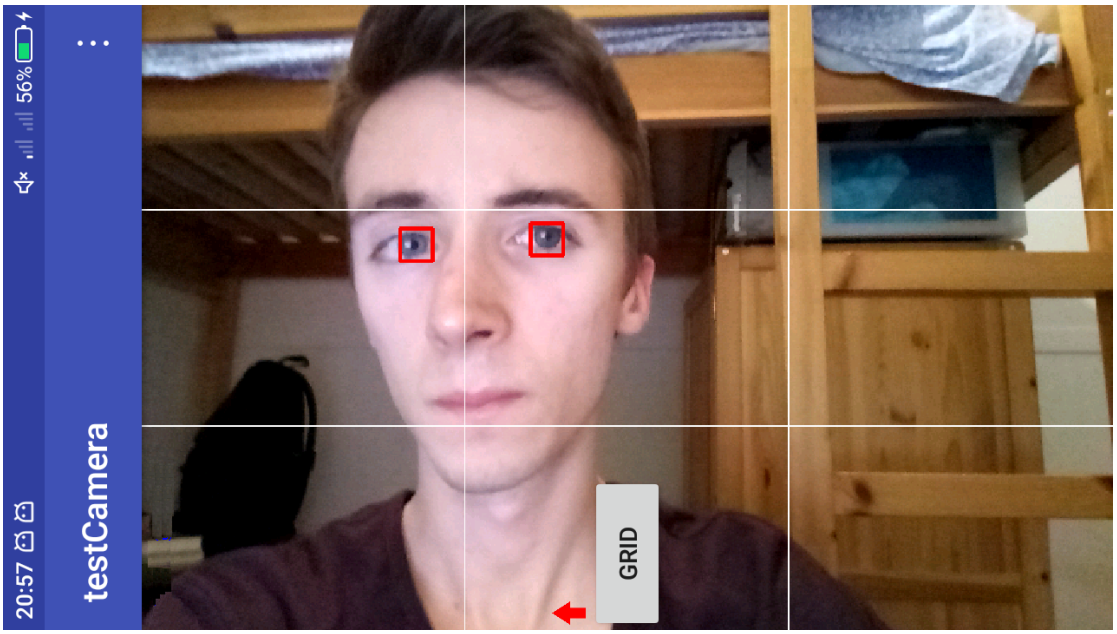


FIGURA 6.7: Teste nº2 do algoritmo visualizando a grelha

6.4 Discussão

Para os algoritmos de deteção facial, apesar da diferença da taxa de deteção destes algoritmos para o algoritmo HOG ser bastante significativa, o algoritmo optado para ser utilizado na aplicação móvel foi o LBP Improved 1.1, devido ao facto de ser o que consome menos processamento e por sua vez ser mais rápido nos cálculos, o que é bastante importante se pensarmos nas limitações de um *smartphone* (tanto a nível de processamento como de bateria). Outra razão para ter sido escolhido o LBP é o facto deste ser o algoritmo que possui um número mais baixo de "caras detetadas mas mal localizadas".

Para as sugestões do enquadramento, o valor da tolerância escolhido foi entre 0.35 e 0.5 de forma a obter um valor não muito distante dos pontos de ouro.

Capítulo 7

Conclusões

A capacidade de captação de imagens de grande qualidade das câmaras fotográficas presentes nos dispositivos móveis, originou um grande número de potenciais "fotógrafos amadores", contudo a maioria destes utilizadores possui pouco ou quase nenhum conhecimento sobre fotografia.

Neste trabalho é apresentado um método que dá indicações ao utilizador sobre como melhorar o enquadramento fotográfico em tempo real, utilizando para tal algoritmos de deteção facial e de olhos, usados em conjunto de forma a maximizar a taxa de sucesso na deteção dos olhos, com o objetivo de fazer coincidir a posição dos olhos entre uma tolerância a cada ponto de ouro.

Para tal foram comparados 3 algoritmos de deteção facial de forma a encontrar um que permitisse atingir uma boa eficácia de deteção com o menor custo computacional, chegando a atingir uma taxa de deteção de 91% com uma implementação do algoritmo LBP. E de maneira a determinar um valor de tolerância a cada ponto de ouro foi efetuada uma análise à posição dos olhos presentes num conjunto de imagens consideradas como tendo um bom enquadramento, observando que a grande maioria se encontravam entre uma tolerância de 0.35 a 0.5 de um ponto de ouro.

Devido ao enquadramento depender do tipo de plano fotográfico, foram utilizadas redes neuronais convolucionais para identificar o tipo de plano presente. Foram

treinadas várias redes, as quais foram testadas e comparadas para determinar qual a que melhor se adequa ao problema.

A rede final atinge 99% de precisão com uma implementação da rede VGG16 usando uma técnica de *transfer learning* e utilizando o optimizador ADAM, com um *learning rate* de 0.00001, treinado com imagens de orientação vertical.

Por fim, a aplicação desenvolvida no sistema Android, indica ao utilizador como melhorar o enquadramento, apenas com base na posição dos olhos e na regra dos terços, não tendo sido incluída a classificação dos planos fotográficos.

7.1 Trabalho Futuro

Dado que numa fase onde foram necessários muitos testes a vários algoritmos, tanto a nível de deteção facial como na deteção dos planos fotográficos com as redes neuronais, de modo a facilitar, controlar e analisar os resultados obtidos a aplicação foi desenvolvida em *Python*.

A aplicação móvel desenvolvida em Android até ao momento apenas implementa o método de sugestões relativas ao enquadramento, com base na posição dos olhos. Estando em falta a implementação do módulo de sugestões responsável por determinar as regras de enquadramento específicas para cada plano fotográfico, após a identificação do mesmo, assim como outras sugestões:

- Alertar o utilizador para não utilizar o flash, caso este se encontre no plano fotográfico *big close-up*;

- No caso do plano fotográfico *big close-up*, indicar ao utilizador qual o propósito daquele plano e as sensações que aquele tipo de fotografia transmite (ex: Este tipo de plano foca-se nas expressões do sujeito).

É também necessário efetuar testes de forma a verificar que valores de iluminância (lux) e de luminância relativa (entre 1 e 100) correspondem a uma fotografia com pouca/muita luz e qual o valor "ideal" de uma boa iluminação.

Anexos

Anexo A

Planos Fotográficos

Definições close up e big close-up segundo as fontes:

- "Figura humana é enquadrada dos ombros para cima. É considerado um plano expressivo e simbólico. " [Santana, 2016]

Primeiro plano ou close-up:

- "Quando se mostra os ombros e rosto da pessoa." [Medya, 2017]

• Enquadra o rosto;

• Plano é cortado pouco abaixo das axilas. Permite por exemplo imagens de alguém a fumar, cortando totalmente o ambiente em redor. Este tipo de planos privilegia o que é transmitido pela expressão facial; [X6, 2018]

• Essencial para se alcançar a máxima intensidade dramática;

• Apresenta nitidamente a expressão do rosto e projeta as características do personagem;

• Pode revelar pensamentos e o momento interior do personagem;

• Corresponde à invasão do campo da consciência;

• Desempenha função mais emocional. [Duarte, 2012]

Plano Close-up – Cobre o sujeito fotografado desde a linha dos ombros para cima. [PlanosFotográficos, 2017]

Plano Médio

"Cortamos pela cintura e a personagem. Permitem-nos identificar-nos com o sujeito. Interessa a reação da personagem, a sua expressividade, sua ação... Valor dramático, mas conservando verdadeiro valor narrativo. (Planos)".

Plano médio longo: "Um pouco mais abaixo da cintura, mas sem chegar aos joelhos."

Plano médio curto: "Entre peito e cintura."[tvlata, 2017]

Plano médio – Este plano cobre o sujeito fotografado da cintura para cima, sendo a linha de corte desde a virilha até meio do peito (plano médio curto). Caso este esteja sentado a linha de corte pode situar-se a meio da coxa. [Illescas, 2017]

Bibliografia

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2037–2041.

[Brownlee, 2018] Brownlee, J. (2018). `https://machinelearningmastery.com/transfer-learning-for-deep-learning/`. Accessed: 2018-09-19.

[CalcLBP, 2018] CalcLBP (2018). Cálculo de lbp. `https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html`. Accessed: 2018-09-19.

[Camera2API, 2018] Camera2API, A. (2018). `https://developer.android.com/reference/android/hardware/camera2/package-summary.html`. Accessed: 2018-09-19.

[Cedric Demers, 2018] Cedric Demers, M. A. (2018). `https://www.rtings.com/tv/learn/what-is-the-aspect-ratio-4-3-16-9-21-9`. Accessed: 2018-10-21.

[Chang-Yeon, 2008a] Chang-Yeon, J. (2008a). `http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf`. Accessed: 2018-09-19.

[Chang-Yeon, 2008b] Chang-Yeon, J. (2008b). Face detection using lbp features. *Final Project Report*.

[Chang-Yeon, 2008c] Chang-Yeon, J. (2008c). Lbp padrões uniformes. `http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf`. Accessed: 2018-09-19.

[Chollet, 2017] Chollet, F. (2017). *Deep learning with python*. Manning Publications Co.

[Cireşan et al., 2012] Cireşan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.

[Classifier, 2018] Classifier, C. (2018). `https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQTMG6DEdyOk7nTgJGp8U-tIVLGA5WCxvvzQdimhWifdUvE9HuF`. Accessed: 2018-09-19.

[Cruz et al., 2015] Cruz, C., Shiguemori, E. H., and GUIMARÃES, F. (2015). A comparison of haar-like, lbp and hog approaches to concrete and asphalt runway detection in high resolution imagery. *Journal of Computational Interdisciplinary Sciences*, 6(3):121–136.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[Deng and Yu, 2014] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. Technical report.

[Developers, 2018] Developers, G. (2018). Accuracy calculation. `https://developers.google.com/machine-learning/crash-course/classification/accuracy`. Accessed: 2018-10-10.

[Duarte, 2012] Duarte, M. (2012). Planos e ângulos. `https://pt.slideshare.net/mduart/planos-e-ngulos`. Accessed: 2017-12-21.

[HaarFeactures, 2018] HaarFeactures (2018). Exemplo de caracaterísticas haar. `https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html`. Accessed: 2018-09-19.

[HaarFeacturesB, 2018] HaarFeacturesB (2018). https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html. Accessed: 2018-09-19.

[hog, 2018] hog (2018). https://cdn-images-1.medium.com/max/800/1*WF54tQnH1Hgpoqk-Vtf9Lg.gif. Accessed: 2018-09-19.

[HuaweiMate10, 2018] HuaweiMate10 (2018). http://www.zdnet.com/article/huawei-mate-10-pro-camera-enhancing-auto-mode-through-artificial-intelligence Accessed: 2018-09-19.

[Illescas, 2017] Illescas, S. (2017). https://www.dzoom.org.es/el-retrato-fotografico-tipos-de-plano/. Accessed: 2017-12-21.

[ImageNet, 2018a] ImageNet (2018a). http://image-net.org/challenges/LSVRC/2014/. Accessed: 2018-09-24.

[ImageNet, 2018b] ImageNet (2018b). http://image-net.org/. Accessed: 2018-09-24.

[instagram scraper, 2017] instagram scraper (2017). https://github.com/rarcega/instagram-scraper. Accessed: 2017-12-24.

[IntegralImage, 2018] IntegralImage (2018). Integral image. http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-1.pdf. Accessed: 2018-09-19.

[Judd et al., 2009] Judd, T., Ehinger, K., Durand, F., and Torralba, A. (2009). Learning to predict where humans look. http://people.csail.mit.edu/tjudd/WherePeopleLook/ALLSTIMULI.zip.

[Kamencay et al., 2017] Kamencay, P., Benčo, M., Mižduš, T., and Radil, R. (2017). A new method for face recognition using convolutional neural network.

[Karpathy, 2018a] Karpathy, A. (2018a). http://cs231n.github.io/convolutional-networks/. Accessed: 2018-10-10.

[Karpathy, 2018b] Karpathy, A. (2018b). http://cs231n.github.io/convolutional-networks/. Accessed: 2018-10-10.

[Karpathy, 2018c] Karpathy, A. (2018c). http://cs231n.github.io/optimization-1/. Accessed: 2018-10-10.

[Keras, 2018a] Keras (2018a). https://keras.io/. Accessed: 2018-02-15.

[Keras, 2018b] Keras (2018b). Keras categorical accuracy calculation. https://keras.io/metrics/. Accessed: 2018-10-10.

[Keras, 2018c] Keras (2018c). Keras image generator. https://keras.io/preprocessing/image/. Accessed: 2018-10-10.

[Keras, 2018d] Keras (2018d). Keras sequential model. https://keras.io/models/sequential/. Accessed: 2018-02-15.

[Kodak, 2017] Kodak (2017). Estimating luminance and illuminance with reflection-type exposure meters and an 18% neutral test card. https://web.archive.org/web/20070709163424/http://www.kodak.com/cluster/global/en/consumer/products/techInfo/am105/am105kic.pdf. Accessed: 2017-12-24.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[LeNet, 2018] LeNet, A. R. (2018). https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/. Accessed: 2018-02-19.

[Levels, 2017] Levels, L. (2017). https://www.noao.edu/education/QLTkit/ACTIVITY_Documents/Safety/LightLevels_outdoor+indoor.pdf. Accessed: 2017-12-24.

[Levels2, 2017] Levels2, L. (2017). http://www.scopecalc.com/. Accessed: 2017-12-24.

[light levels, 2017] light levels, M. (2017). https://web.archive.org/web/20150112024752/http://academy.autodesk.com:80/library/building-science/measuring-light-levels. Accessed: 2017-12-24.

[Luminance, 2017] Luminance (2017). luminancia relativa. `https://www.w3.org/Graphics/Color/sRGB`. Accessed: 2017-12-24.

[Mansurov, 2018] Mansurov, N. (2018). `https://photographylife.com/what-is-aperture-in-photography`. Accessed: 2018-10-10.

[Mathies, 2018a] Mathies, D. (2018a). `https://www.digitaltrends.com/photography/what-is-iso/`. Accessed: 2018-10-10.

[Mathies, 2018b] Mathies, D. (2018b). `https://www.digitaltrends.com/photography/understanding-exposure-settings/`. Accessed: 2018-10-10.

[Medya, 2017] Medya (2017). Tipos de plano. `http://cursodefoto-madrid.com/curso-de-fotografia-los-diferentes-tipos-de-plano`. Accessed: 2017-12-21.

[MIT, 2017] MIT, D. S. (2017). `http://saliency.mit.edu/datasets.html`. Accessed: 2017-12-31.

[Ojala et al., 2002] Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987.

[openCamera, 2018] openCamera (2018). `http://opencamera.org.uk/`. Accessed: 2018-09-19.

[openCV, 2018] openCV (2018). `http://opencv.org`. Accessed: 2018-09-19.

[PlanosFotográficos, 2017] PlanosFotográficos (2017). Planosfotograficos. `https://ppunipar.files.wordpress.com/2016/06/planos.png`. Accessed: 2017-12-24.

[pOuro, 2017] pOuro (2017). `http://eaibeleza.com/wp-content/uploads/2014/10/ruleofthirds-850x566.jpg`. Accessed: 2017-12-24.

[Raschka, 2015] Raschka, S. (2015). `https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html`. Accessed: 2018-10-10.

[Ratio, 2018] Ratio, A. (2018). `https://en.wikipedia.org/wiki/Aspect_ratio`. Accessed: 2018-10-21.

[RgbToLinear, 2018] RgbToLinear (2018). `https://en.wikipedia.org/wiki/SRGB#The_reverse_transformation`. Accessed: 2018-09-19.

[Rodriguez, 2006] Rodriguez, Y. (2006). Face detection and verification using local binary patterns. Technical report, Ecole Polytechnique Fédérale de Lausanne.

[Rojas et al., 2011] Rojas, M., Masip, D., Todorov, A., and Vitria, J. (2011). Automatic prediction of facial trait judgments: Appearance vs. structural models. *PloS one*, 6(8):e23323.

[Rosebrock, 2018a] Rosebrock, A. (2018a). `https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/`. Accessed: 2018-09-19.

[Rosebrock, 2018b] Rosebrock, A. (2018b). `https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/`. Accessed: 2018-02-19.

[Santana, 2016] Santana, L. (2016). Tipos de plano. `https://roteiristaleo.wordpress.com/artigos-publicados/sobre-roteiros-para-hqs/sobre-enquadramentos-para-hqs/`. Accessed: 2017-12-21.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Singh, 2018] Singh, C. D. (2018). Image classification: Cifar-10 neural networks vs support vector machines. Accessed: 2018-10-10.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Surmenok, 2017] Surmenok, P. (2017). `https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0`. Accessed: 2018-10-10.

[Thirds, 2018] Thirds, R. (2018). `https://www.digital-photography-tips.net/digital-photography-tutor-thirds.html`. Accessed: 2018-09-19.

[ThirdsB, 2018] ThirdsB, R. (2018). `http://omeuolhar.com/artigos/que-regra-tercos`. Accessed: 2018-09-19.

[tvlata, 2017] tvlata (2017). `http://www.tvlata.org/node/306`. Accessed: 2017-12-21.

[Viola and Jones, 2001a] Viola, P. and Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1. IEEE.

[Viola and Jones, 2001b] Viola, P. and Jones, M. (2001b). Robust real-time face detection. In *Second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing, and Sampling*. IEEE.

[Wang, 2018] Wang, K. K. (2018). Image classification with pyramid representation and rotated data augmentation on torch 7. Accessed: 2018-10-10.

[X6, 2018] X6, S. M. (2018). Planos e ângulos. `https://community.serif.com/appresources/MPX6/Tutorials/en-us/tutorials/basics_shottypes.htm`. Accessed: 2018-04-16.