

# Age Guesser

Universidade de Aveiro  
DETI

1<sup>st</sup> Rui Santos  
Estudante  
DETI  
Universidade de Aveiro  
Aveiro, Portugal  
ruipedro99@ua.pt

2<sup>nd</sup> Tomás Martins  
Estudante  
DETI  
Universidade de Aveiro  
Aveiro, Portugal  
tomasfilipe7@ua.pt

3<sup>rd</sup> Pétiá Georgieva  
Docente  
DETI  
Universidade de Aveiro  
Aveiro, Portugal  
petia@ua.pt

**Abstract**—O presente documento trata-se de um relatório elaborado no âmbito da Unidade Curricular de Aprendizagem Automática, no contexto do segundo projeto. Neste, irá ser referida uma análise ao estado da arte do tema em questão, assim como uma descrição dos dados, o seu pré-processamento e os algoritmos de aprendizagem automática em utilização, e ainda uma apresentação dos resultados e conclusões tiradas.

**Index Terms**—Aprendizagem Automática; Neural Networks; Deep Learning; Convolutional Neural Network; Processamento de imagem; Dados; Python; Classificação; Dataset; Overfitting; Underfitting; Epoch.

## I. INTRODUÇÃO

### A. Motivação

As análises faciais têm sido um grande objeto de estudo por muitos investigadores no decorrer dos últimos anos. Não só pode permitir detetar situações menos desejadas como doenças, como também permite uma grande evolução no que toca à interação humano-computador. Posto isto, um dos grandes desafios encontrados no que toca a este tema, é conseguir uma estimativa de identificação da idade de uma pessoa pelo seu rosto.

### B. Objetivo

Este trabalho tem como objetivo analisar um conjunto de imagens de pessoas, fazer o seu processamento através da aplicação de diversos filtros entre outras técnicas e classificá-las de acordo com a sua faixa etária. Pretende-se deste modo, verificar se a aplicação de algoritmos de processamento de imagem podem resultar numa melhoria da precisão dos modelos de ML.

Para tal, foi decidido utilizar uma classificação de 5 níveis:

- **young** - abaixo dos 23 anos
- **middle1** - dos 24 aos 35 anos
- **middle2** - dos 36 aos 45 anos
- **middle3** - dos 46 aos 60 anos
- **old** - acima dos 61 anos

## II. ANÁLISE DO ESTADO DA ARTE

Inicialmente, foi efetuado um estudo afim de entender que tipo de abordagens eram tomadas para tratar o problema em questão. Nesta secção serão discutidos os projetos estudados e as respetivas conclusões que modelaram o rumo do projeto.

### A. Estudo 1

O primeiro trabalho estudado foi *Age Prediction From Images (CNN Regression)* por Gabriel Atkin [5]. Este projeto visa prever a idade precisa de pessoas entre os 20 e os 50 anos através do treino de um modelo desenvolvido com TensorFlow/Keras CNN que utiliza o dataset *Age prediction* [6].

Numa primeira análise, os resultados apresentados no artigo pareceram bastante positivos, uma vez que o modelo apresenta uma distância média da idade prevista à real na ordem dos 9 anos, o que é uma média aceitável.

Assim, foi decidido treinar o modelo apresentado no artigo com outro dataset composto por imagens de pessoas de todas as idades. Após correr o programa, notou-se que os resultados não foram satisfatórios, visto que o *root-mean-square error (RMSE)* esteve na ordem dos 17, e os valores previstos pelo modelo estiveram constantemente compreendidos entre os 33 e os 41 anos. Os resultados podem ser constatados pela figura 1.

```
Test RMSE: 17.09560
Test R^2 Score: 0.01383
Null/Baseline Model Test RMSE: 17.21508
```

Fig. 1. Output do estudo 1 adaptado

Assim, concluiu-se que o modelo construído não foi satisfatório, uma vez que se trata de um algoritmo de regressão e que foi desenvolvido para obter uma média de distâncias da idade prevista à real o mais pequena possível. Deste modo, o modelo acabou por atribuir idades medianas, não prevendo assim, jovens ou idosos. Os resultados apresentados no artigo correspondem ao que foi anteriormente referido, uma vez que o intervalo de idades do seu dataset é de 30 anos, ou seja, é muito menos do que o intervalo do dataset que foi testado, o que resulta, naturalmente, num inferior RMSE.

### B. Estudo 2

O segundo artigo estudado foi *Using artificial neural network for human age estimation based on facial images* por Sarah N. Kohail [7] cujo objetivo é prever a idade precisa de pessoas através da análise das suas fotografias, tal como o projeto referido no último parágrafo.

No entanto, este trabalho apresenta bastantes diferenças, nomeadamente as idades das pessoas nas fotografias que se encontram desde os 0 aos 89 anos, o número de *epochs* é bastante superior, sendo igual a 2000 num modelo e 4000 para outro e existe uma análise das imagens muito mais profunda, detectando 94 pontos de referência e obtendo assim múltiplos *features* como a grossura e o comprimento da sobrancelha, comprimento do nariz e da boca, contorno e distância dos vários elementos da face, etc. Estes *features* obtidos a partir das imagens são bastante relevantes visto que oferecem informação clara e importante para a previsão das idades. Na figura 2 está representado um exemplo do processamento e deteção de pontos efetuado.

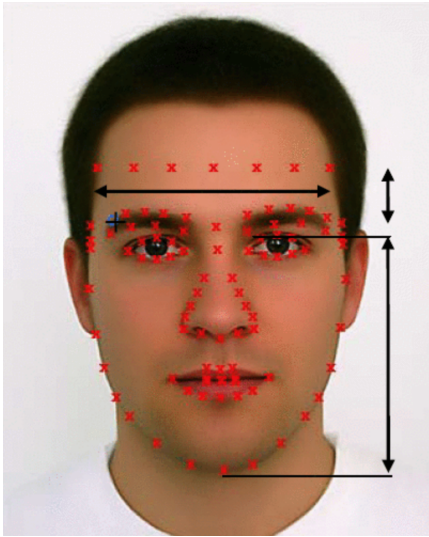


Fig. 2. Exemplo do processamento feito no estudo 2

Foi também aplicado *Artificial Neural Networks (ANN)* e *Multi-layer perceptron network (MLP)* como ferramenta de regressão e de classificação. Assim, obtiveram-se valores bastante positivos, na ordem dos 88% de *accuracy* o que mostrou que a utilização de redes *MLP* é uma boa solução para o problema em questão.

Contudo, estes resultados podem certamente dever-se ao processamento das imagens e obtenção de *features* que requerem que as faces das pessoas estejam alinhadas com a fotografia, o que restringe a aplicação dos modelos desenvolvidos a ambientes mais específicos e resulta numa análise menos prática.

### C. Estudo 3

O terceiro projeto estudado foi *Facial-Demographics-using-CNN* [11] que usa o *dataset UTKFace* [10] que contém 20 000

imagens de pessoas de todas as idades com *labels* relativas ao género, idade e etnia. Este projeto, ao contrário dos projetos mencionados anteriormente, não visa prever a idade precisa das pessoas, mas sim identificar a classe de idade a que as pessoas pertencem bem como o seu género. Trata-se, portanto, de um problema de classificação e não de regressão.

O projeto usou, essencialmente, *Convolutional Neural Networks* no desenvolvimento do modelo bem como a tecnologia *Flask* para desenvolver uma aplicação *user friendly* para permitir o upload de fotografias de pessoas e devolver a classe de idades prevista pelo modelo. A figura 3 representa as *layers* utilizadas na elaboração do modelo.

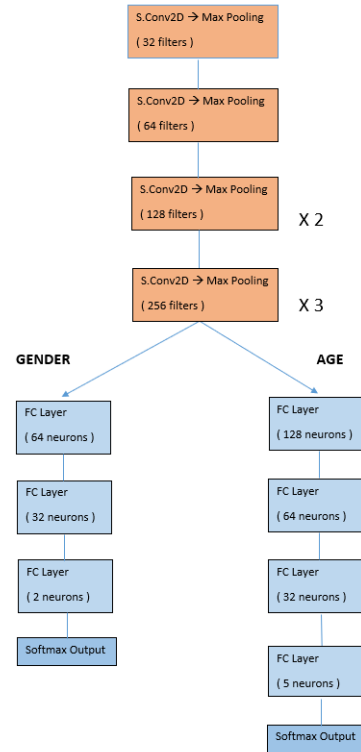


Fig. 3. Layers do modelo do estudo 3

Os resultados obtidos foram bastante positivos, sendo que o *val\_accuracy* alcançou os 82% na previsão da classe de idades e 95% na previsão do género, o que mostra que a abordagem de classificação por idades é uma boa forma de adotar o problema, uma vez que na maioria dos casos de aplicação, não é necessário saber a idade precisa das pessoas mas antes a sua faixa etária.

Apesar dos resultados terem sido bastante positivos, acredita-se que se devem bastante aos *labels* de género e etnia usados na previsão da idade. Este método não é prático numa situação real, uma vez que, ao analisar as fotografias das pessoas, não se terá qualquer informação ou *label* das mesmas.

### D. Estudo 4

Foi também estudado o artigo *Deep Convolutional Neural Network for Age Estimation based on VGG-Face Model* que

usa *Convolutional Neural Networks* para a classificação de pessoas por faixas etárias.

Foi usada uma abordagem bastante parecida com a referida anteriormente, tendo sido atingida uma *val\_accuracy* na ordem dos 50% e 90% quando se usa 1 faixa etária de intervalo.

Concluiu-se, mais uma vez, que a utilização de *CNN* na classificação de faixas etárias é uma boa abordagem para a presente questão.

#### E. Estudo 5

Por último, foi estudado o projeto *Designing an Age Classification Model with Deep Learning* por *Pranjal Saxena* [13] que aborda o presente problema da mesma forma que o artigo anterior, através da classificação das pessoas por faixa etária.

O *dataset* usado apenas contém 3 faixas etárias:

- YOUNG
- MIDDLE
- OLD

Apesar do número reduzido de classes usadas, o modelo não obteve excelentes resultados, apenas 38%. No entanto, o código é bastante simples, perceptível e os passos seguidos foram bem documentados. Pareceu ser um bom ponto de partida para o presente projeto e foi com base nele que se elaborou o código e se obteve os resultados que irão ser apresentados nos capítulos seguintes.

### III. DESCRIÇÃO DOS DADOS, VISUALIZAÇÃO E ANÁLISE ESTATÍSTICA

Para a realização deste estudo, utilizou-se um *dataset* composto por 60383 imagens organizadas em pastas de acordo com a idade da pessoa presente na fotografia, disponibilizadas pela *Wikipédia*. Estas imagens contêm a face de celebridades de todas as idades, em diversas poses, com diversas expressões faciais.

No entanto, a quantidade de fotos para cada idade estava desbalanceada, como se pode observar na figura 4.

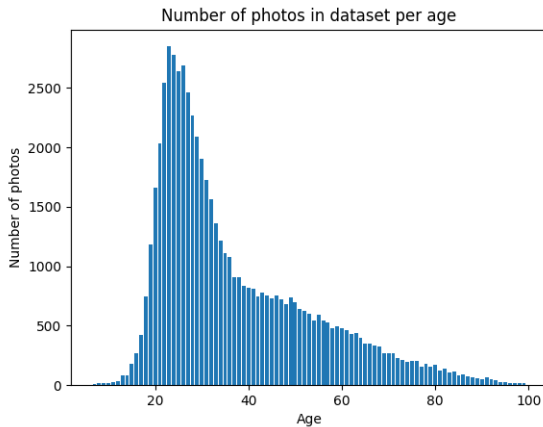


Fig. 4. Número de fotografias por idade no dataset

### IV. PRÉ-PROCESSAMENTO DOS DADOS

#### A. Organização hierárquica do dataset inicial

O *dataset* de imagens inicial não continha qualquer tipo de organização ou hierarquia, somente um diretório com o conjunto de todas as 60383 fotografias aleatoriamente dispostas, com a idade da pessoa em questão como parte do nome do ficheiro. Decidiu-se portanto, automatizar um processo que percorresse todos estes ficheiros e os distribuisse por diversas pastas consoante a idade de modo a facilitar a filtragem e utilização dos dados.

O código utilizado para este processo está apresentado em seguida:

```
import os
import glob
import shutil

for i in glob.glob("/Users/ruisantos/Desktop/ano4/aa/project2/wiki_crop/**/*.jpg"):
    filename = i.split("/")[-1]
    year1 = (int)(filename.split("_")[1].split("-")[0])
    year2 = (int)(filename.split("_")[-1].split(".jpg")[0])
    age = year2 - year1
    if not os.path.exists("/Users/ruisantos/Desktop/ano4/aa/project2/wiki_crop/photos/"+str(age)+"/"):
        os.makedirs("/Users/ruisantos/Desktop/ano4/aa/project2/wiki_crop/photos/"+str(age)+"/")
    shutil.move(i, "/Users/ruisantos/Desktop/ano4/aa/project2/wiki_crop/photos/"+str(age)+"/"+filename)
```

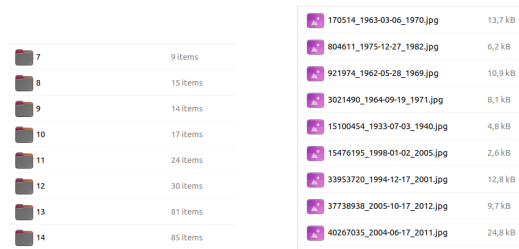


Fig. 5. Esquerda - Organização das pastas por idades. Direita - Ficheiros dentro de cada pasta.

#### B. Divisão das idades por classes

Estando tudo organizado por as idades correspondentes, voltou-se a organizar os dados, mas desta vez por classes, agrupando diferentes faixas etárias, com recurso à hierarquia criada anteriormente, em diferentes conjuntos, que seriam estes:

- **Young** - Pessoas dos 7 aos 22 anos.
- **Middle1** - Pessoas dos 23 aos 34 anos.
- **Middle2** - Pessoas dos 35 aos 44 anos.
- **Middle3** - Pessoas dos 45 aos 59 anos.
- **Old** - Pessoas dos 60 aos 100 anos.

Esta organização, iria permitir uma classificação mais precisa e resultados mais satisfatórios no resultado final.

```
import shutil
import os
import cv2

source_dir = '/home/tomasfilipe7/Desktop/
Universidade/4_Ano/2_Semestre/AA/age_guesser
/data/data/photos/{ }'
target_dir = '/home/tomasfilipe7/Desktop
/Universidade/4_Ano/2_Semestre/AA/age_guesser
/data/data/photos2/{ }'

image_id = 0
for i in range(7, 100):
    this_src_dir = source_dir.format(str(i))
    this_dest_dir = ""
    if(i < 23):
        this_dest_dir = target_dir.format('young')
    elif(i < 35):
        this_dest_dir = target_dir.format('middle1')
    elif(i < 45):
        this_dest_dir = target_dir.format('middle2')
    elif(i < 60):
        this_dest_dir = target_dir.format('middle3')
    else:
        this_dest_dir = target_dir.format('old')

file_names = os.listdir(this_src_dir)
for file_name in file_names:
    if (os.path.getsize(source_dir
        .format(str(i) + "/" + file_name)) >
        334) & (file_name != '.DS_Store'):
        # shutil.copy(os.path.join(this_
        src_dir, file_name), this_dest_
        dir)
        image = cv2.imread(os.path.join
        (this_src_dir, file_name), 1)
        imgray = cv2.cvtColor(image,
        cv2.COLOR_BGR2GRAY)
        print(os.path.join(this_dest_
        dir, file_name))
        cv2.imwrite(os.path.join(this_
        dest_dir, file_name), imgray)
```






 middle1	21913 items
 middle2	8308 items
 middle3	8772 items
 old	6448 items
 young	7450 items

Fig. 6. Images divided by class

Por último, quando as imagens estiverem corretamente organizadas, o conteúdo de cada diretório será dividido em imagens de treino e imagens de teste. Optou-se por utilizar 90% (corresponde a 5760 fotografias) das mesmas para treinar o modelo e os restantes 10%(corresponde a 640) para o testar.

```
import shutil
import os
import glob

for i in glob.glob("/Users/ruisantos/Desktop
/ano4/aa/project2/heartbeat/photos2/*"):
    filename = i.split("/")[-1]
    if not os.path.exists("/Users/ruisantos/
    Desktop/ano4/aa/project2/heartbeat/train
    /"+filename+"/"):
        os.makedirs("/Users/ruisantos/Desktop
        /ano4/aa/project2/heartbeat/train
        /"+filename+"/")
    os.makedirs("/Users/ruisantos/Desktop
    /ano4/aa/project2/heartbeat/test
    /"+filename+"/")

cont = 1
for i in glob.glob(i+"/*"):
    if cont < 5760: # 90% para treino
        shutil.copy(i,"/Users/ruisantos
        /Desktop/ano4/aa/project2
        /heartbeat/train/"+filename+"/")
    elif cont < 6400: # 10% para treino
        shutil.copy(i,"/Users/ruisantos
        /Desktop/ano4/aa/project2
        /heartbeat/test/"+filename+"/")
    else:
        break
    cont+=1
```

### C. Filtragem das imagens

De modo a melhorar os resultados, utilizou-se uma abordagem que consistia em editar as imagens com diversos filtros que iriam normalizar as cores das imagens ou abstrair um pouco os elementos menos relevantes das mesmas, como a cor, o fundo, ou qualquer parte do corpo que não fosse a cara, com recurso à biblioteca *OpenCV*.

Para tal, começou-se por transformar as cores de todas as imagens numa escala de cinzento (Fig. 3), pois isto permitiria que as imagens tivessem todas o mesmo esquema de cores e o modelo não estaria inclinado a usar esse fator como um elemento de comparação.

```
image = cv2.imread(os.path.join(this_src_
dir, file_name), 1)
grayimage = cv2.cvtColor(image, cv2.COLOR
_BGR2GRAY)
cv2.imwrite(os.path.join(this_dest_dir ,
file_name), grayimage)
```



Fig. 7. Imagem transformada em tons de cinzento.

Outro filtro que se utilizou com o intuito de obter uma melhor solução, foi ofuscar o fundo das imagens (*blur*). Para tal, recorreu-se à utilização de um classificador em cascata baseado num *Haar-like feature*, que consiste numa abordagem de aprendizagem automática para identificação de objetos, que utiliza uma função em cascata para treinar um conjunto de imagens positivas (imagens que contêm o que se pretende identificar) e negativas (imagens que não contêm o que se pretende identificar) de modo a reconhecer um determinado objeto. Recorreu-se à utilização de um modelo já pré-treinado ([9]) que reconhece faces frontais e, deste modo, foi possível, recorrendo à utilização da funcionalidade *blur* da biblioteca *OpenCV*, esbater o fundo da imagem, realçando a cara das pessoas.

```
def blur_background( grayscale , image):

    face = cascade_face.detectMultiScale(
        (grayscale , 1.3 , 5)

    for(x_face , y_face , w_face , h_face)in face:
        #storing face
        roi_face= image[y_face: y_face+
            h_face , x_face: x_face+ w_face]
        image = cv2.blur(image,(10,10))
        image[y_face: y_face+ h_face , x_face:
            x_face+ w_face]=roi_face
    return image

image = cv2.imread(os.path.join(this_src_dir ,
    file_name), 1)
cascade_face = cv2.CascadeClassifier(
    (os.path.join(current_dir ,
        'haarcascade_frontalface_default.xml'))
grayscale = cv2.cvtColor(image ,
    cv2.COLOR_BGR2GRAY)
image_result = blur_background(grayscale , image)
print(os.path.join(this_dest_dir , file_name))
cv2.imwrite(os.path.join(this_dest_dir , file_name),
    image_result)
```



Fig. 8. Imagem com o fundo esbatido, realçando a cara.

Após serem obtidas as imagens com *blur*, ponderou-se utilizar a técnica de processamento de imagem *Canny edge detector*, de maneira a realçar os traços faciais de cada cara, sem se ter o fundo da imagem para perturbar esta técnica, com o propósito de eliminar informação desnecessária para o modelo, de maneira a obter-se uma classificação mais precisa. No entanto, após o processamento das imagens, verificou-se que muitas delas ficaram imperceptíveis, de modo a que optou-se por não utilizar estes dados.

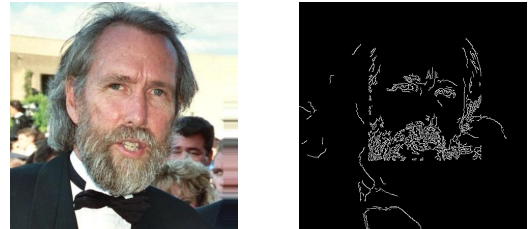


Fig. 9. Exemplo de imagem imperceptível utilizando o *Canny edge detector*

Ainda utilizando o método referenciado no filtro do *blur* (classificador em cascata baseado num *Haar-like feature*), tentou-se identificar a cara nas imagens e recortá-la do resto da fotografia. Isto permitiu isolar completamente a maior parte das caras, mas em contrapartida perdeu-se algumas imagens por falta de identificação da cara ou perdeu-se certos detalhes na fotografia que poderiam ser relevantes para a classificação da mesma.

```
def cut_face( grayscale , image):
    face = cascade_face.detectMultiScale(
        (grayscale , 1.3 , 5)
    if (len(face) > 1):
        print("Too long")
        return None
    elif(len(face) == 0):
        print("No faces")
        return image
    else:
        x_face , y_face , w_face , h_face
        = face[0]
        roi_face= image[y_face: y_face +
            h_face ,
            x_face: x_face+ w_face]
        image = roi_face

    return image

image = cv2.imread(os.path.join(this_src_dir ,
    file_name), 1)
cascade_face = cv2.CascadeClassifier(os.path
    .join
        (current_dir ,
            'haarcascade_frontalface_default.xml'))
grayscale = cv2.cvtColor(image
    , cv2.COLOR_BGR2GRAY)
image_result = cut_face( grayscale
```



```

        , image)
if not image_result is None:
    print(os.path.join(this_dest_dir
                        , file_name))
    cv2.imwrite(os.path.join(this_dest_dir
                              , file_name), image_result)

```



Fig. 10. Imagem cortada segundo o método do classificador em cascata

## V. DESCRIÇÃO DOS ALGORITMOS DE APRENDIZAGEM AUTOMÁTICA APLICADOS

### A. Convolutional Neural Networks

Dado que se está a trabalhar com análise de imagens em grande quantidade, optou-se por utilizar uma solução baseada em *Convolutional Neural Networks* (CNN). *Neural Networks* são sistemas de computação de entradas inspirados nas redes neurais dos seres vivos. Estas são compostas por diferentes *nodes* interligados que partilham sinais entre si, distribuídos por camadas.

Uma CNN é uma classe das *Neural Networks*, muito comum na utilização de análise de imagem, pois utiliza um tipo de camadas específicas, as *Convolutional Layers*. Estas camadas utilizam como entrada, informação de diversos nós anteriores, o que permite uma proximidade e criação do que é chamado, um bairro. Esta abordagem permite reduzir o número de nós na rede, pois o seu sistema funciona de acordo com a associação de "muitos para um", o que resulta numa probabilidade mais reduzida de ocorrer *overfitting*. Simultaneamente, dado que a informação é partilhada dentro dos "bairros", a sua utilização é bastante útil na área da análise de imagem, pois a capacidade de comparar pixels vizinhos é muito importante e fornece uma melhor compreensão da imagem a analisar.

### B. Modelos

Para a elaboração dos modelos, foram utilizadas as bibliotecas *Keras/TensorFlow* que fornecem importantes ferramentas na área de *Machine Learning* e *Neural Networks*. Foi elaborado um modelo base sequencial com 3 grupos de *convolutional*, *RELU* e *pooling layers* com *batch normalization* para que o modelo não sofra de *overfitting*. Finalmente, foi utilizada uma *Dense layer* para que o modelo possa então classificar as imagens em 5 grupos.

Relativamente ao *optimizer*, foi utilizado *Adadelta* com um *learning\_rate* igual a 0.0001 e *decay* igual a  $10^{-6}$ . A *loss function* utilizada foi *categorical\_crossentropy*.

O código seguinte apresenta a estrutura do modelo base:

```

train_path = '/Users/ruisantos/Desktop/ano4/
aa/project2/heartbeat/train/'
train_batches = datagen.flow_from_directory(
train_path, target_size=(200,200), classes=
classes_required, batch_size=batch_size_train)

```

```

test_path = '/Users/ruisantos/Desktop/ano4/
aa/project2/heartbeat/test/'
test_batches = datagen.flow_from_directory(
test_path, target_size=(200,200), classes=
classes_required, batch_size=
batch_size_train)

```

```

model=Sequential()
model.add(Conv2D(16, kernel_size=(3,3),
activation="relu", input_shape=IMAGE_SIZE
+ [3], padding='same'))

```

```

model.add(Conv2D(32, kernel_size=(3,3),
activation="relu", padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.30))

```

```

model.add(Conv2D(64, kernel_size=(3,3),
activation="relu", padding='same'))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.35))

```

```

model.add(Conv2D(128, kernel_size=(3,3),
activation="relu", padding='same'))
#model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.45))

```

```

model.add(Flatten())
model.add(Dense(64, activation="relu"))
model.add(Dense(num_classes, activation=
"softmax"))

```

```

opt = Adadelta(learning_rate=0.0001,
decay=1e-6)
model.compile(optimizer=opt, loss=
'categorical_crossentropy', metrics=
['accuracy'])

```

## VI. MODELO 1

O primeiro modelo implementado tem a estrutura igual ao código do modelo base, apresentado no capítulo V.

## A. Discussão de resultados

```

Epoch 87/100 - 123s 2s/step - loss: 1.8187 - accuracy: 0.2872 - val_loss: 1.3623 - val_accuracy: 0.3313
Epoch 88/100 - 123s 2s/step - loss: 1.7833 - accuracy: 0.2888 - val_loss: 1.3620 - val_accuracy: 0.3316
Epoch 89/100 - 123s 2s/step - loss: 1.8030 - accuracy: 0.3112 - val_loss: 1.3615 - val_accuracy: 0.3381
Epoch 90/100 - 123s 2s/step - loss: 1.7742 - accuracy: 0.3224 - val_loss: 1.3612 - val_accuracy: 0.3324
Epoch 91/100 - 123s 2s/step - loss: 1.8076 - accuracy: 0.2824 - val_loss: 1.3595 - val_accuracy: 0.3324
Epoch 92/100 - 124s 2s/step - loss: 1.8545 - accuracy: 0.2952 - val_loss: 1.3594 - val_accuracy: 0.3344
Epoch 93/100 - 123s 2s/step - loss: 1.7632 - accuracy: 0.3088 - val_loss: 1.3587 - val_accuracy: 0.3367
Epoch 94/100 - 123s 2s/step - loss: 1.7105 - accuracy: 0.3160 - val_loss: 1.3581 - val_accuracy: 0.3363
Epoch 95/100 - 124s 2s/step - loss: 1.7143 - accuracy: 0.3248 - val_loss: 1.3569 - val_accuracy: 0.3398
Epoch 96/100 - 122s 2s/step - loss: 1.7625 - accuracy: 0.3232 - val_loss: 1.3562 - val_accuracy: 0.3414
Epoch 97/100 - 123s 2s/step - loss: 1.7456 - accuracy: 0.3368 - val_loss: 1.3555 - val_accuracy: 0.3422
Epoch 98/100 - 123s 2s/step - loss: 1.7928 - accuracy: 0.2856 - val_loss: 1.3552 - val_accuracy: 0.3422
Epoch 99/100 - 122s 2s/step - loss: 1.7459 - accuracy: 0.3850 - val_loss: 1.3548 - val_accuracy: 0.3426
Epoch 100/100 - 123s 2s/step - loss: 1.7793 - accuracy: 0.2888 - val_loss: 1.3547 - val_accuracy: 0.3426

```

Fig. 11. Resultados do modelo 1

Tal como está representado na figura 11, o valor máximo que a *val\_accuracy* atingiu foi 34.26% na *epoch* 100. Da mesma forma, o valor mínimo da *val\_loss* foi 1.3547, atingido também na *epoch* 100.

## B. Conclusão

Através dos resultados obtidos pode-se verificar que não existe qualquer *overfitting*, uma vez que o *val\_accuracy* subiu constantemente.

Concluiu-se assim, que o valor da *val\_accuracy* é algo insatisfatório, no entanto, acredita-se que este poderia ter sido mais elevado com o aumento do número de *epoch*, o que não foi possível, visto que o treino demorou +/- 4 horas a ser executado.

## VII. MODELO 2

No segundo modelo, foi alterado o parâmetro relativo ao *learning\_rate* do *optimizer* para 0.001.

## A. Discussão de resultados

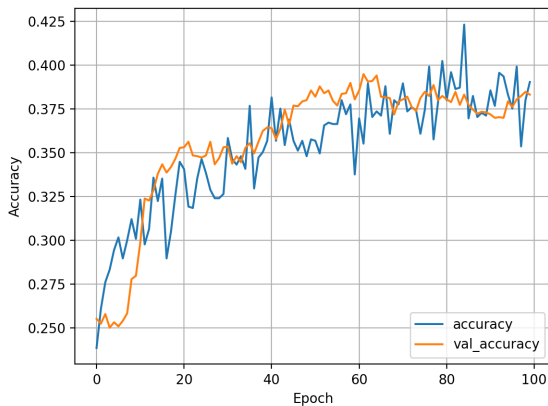


Fig. 12. Accuracy do modelo 2 em função do número de epochs

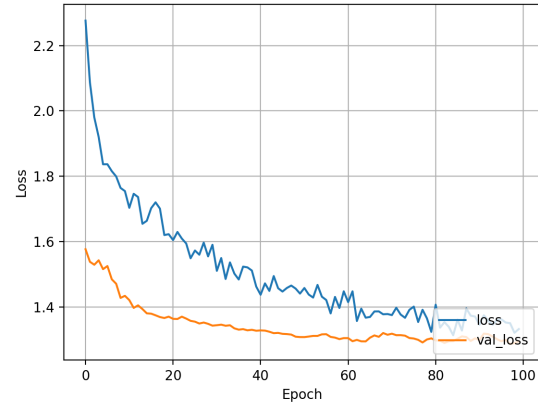


Fig. 13. Loss do modelo 2 em função do número de epochs

Observando os resultados relativos à *val\_accuracy* do modelo 2 é possível verificar uma grande subida da probabilidade entre as *epoch* 1 e 20 que depois atingiu o valor máximo de 39.49% na *epoch* 62. Nas *epochs* seguintes, os valores mantiveram-se relativamente constantes.

Relativamente ao gráfico de *val\_loss*, reparou-se que houve uma descida contante da mesma à medida que o número de *epochs* aumenta. O valor mais baixo foi de 1.2875 obtido na *epoch* 98.

## B. Conclusão

Através dos resultados obtidos, pode-se concluir que houve *underfitting* nas primeiras *epochs*, uma vez que o modelo ainda não teve *runs* suficientes para melhorar. Também se pode verificar que não existiu *overfitting*, visto que os valores da *val\_accuracy* e *val\_loss* se mantiveram relativamente constantes após a *epoch* 60.

Deste modo, verificou-se que a alteração do valor *learning\_rate* resultou num aumento de 5.23%, o que foi bastante positivo.

## VIII. MODELO 3

Neste modelo, o *dataset* utilizado foi gerado a partir das fotografias originais, mas com a transformação das mesmas em preto e branco utilizando a biblioteca *OpenCV*, tal como foi referido na secção IV. Foi mantida a configuração do Modelo 2, descrito na secção VII, com a exceção do *input\_shape* da primeira *layer* que, em vez de ter uma dimensão para cada cor, passa a ter apenas 1. Foram também adicionadas *color\_mode="grayscale"* nas *batches*.

### A. Discussão de resultados

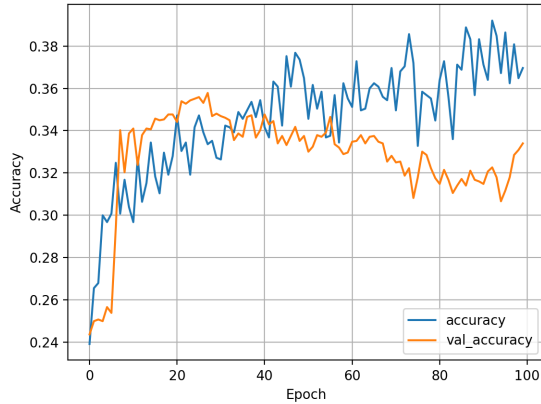


Fig. 14. Accuracy do modelo 3 em função do número de epochs

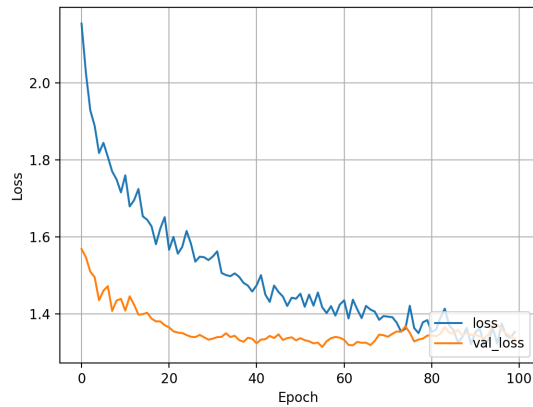


Fig. 15. Loss do modelo 3 em função do número de epochs

Através do gráfico relativo à *val\_accuracy*, pode-se verificar uma subida bastante acentuada nas primeiras *epochs*, um máximo igual a 35.78% que é atingido na *epoch* 28 e uma descida a partir desta *epoch*. O comportamento da *accuracy* é semelhante, porém, este valor sobe depois da *epoch* 28, ao contrário do que aconteceu com os valores da *val\_accuracy*.

Tal como aconteceu no modelo anterior, os valores relativos ao *val\_loss*, desceram contantemente ao longo das *epochs*, tendo adquirido um mínimo de 1.3143 na *epoch* 57.

### B. Conclusão

Analisando os resultados, é possível concluir que, no geral, tem um comportamento semelhante ao observado no modelo 2, no entanto, existe um *overfitting* a partir da *epoch* 28, uma vez que os valores da *val\_accuracy* começam a descer.

Visto que a percentagem da *val\_accuracy* diminuiu 3.17% relativamente ao modelo anterior, conclui-se que a utilização de imagens com tonalidades cinzentas não tem um impacto

positivo no treino dos modelos, possivelmente por estes terem menos informação relativamente à fotografias a cores.

## IX. MODELO 4

Tal como no modelo anterior VIII, no modelo 4 foi aplicado outro filtro ao *dataset* inicial que destaca as faces das pessoas e aplica um efeito de *blur* em torno das mesmas. Este processamento foi explicado mais detalhadamente na secção IV.

A configuração usada no modelo 4 foi a mesma que foi utilizada no modelo 2.

### A. Discussão de resultados

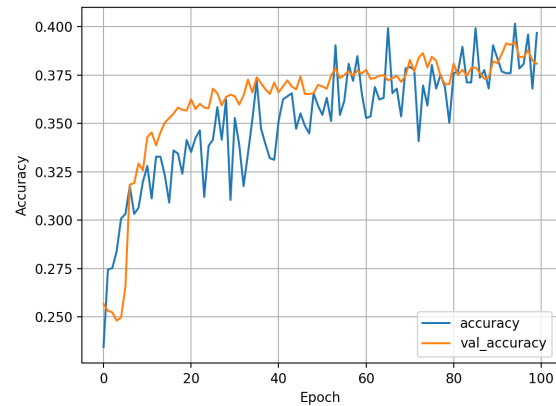


Fig. 16. Accuracy do modelo 4 em função do número de epochs

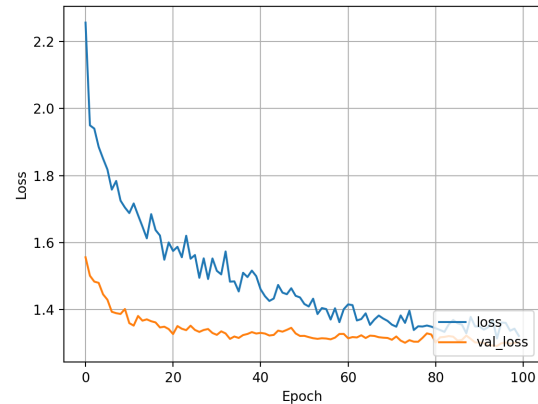


Fig. 17. Loss do modelo 4 em função do número de epochs

Através do gráfico relativo aos valores do *val\_accuracy*, é possível verificar um subida acentuada dos valores da *epoch* 1 à 20 que depois continuam a subir mas de uma forma mais gradual, acompanhando sempre os valores da *accuracy*. Assim, o valor máximo foi de 39.22% atingido na *epoch* 95.

Relativamente aos valores de *val\_loss*, houve uma descida mais acentuada nas primeiras *epochs* mas depois manteve-se sempre constante.



## B. Conclusão

Através dos resultados obtidos, é possível verificar que o modelo teve um comportamento muito semelhante com o modelo 2, sendo que os valores máximos de *val\_accuracy* foram praticamente iguais e em ambos os casos não houve *overfitting*.

Assim, pode-se concluir que a utilização do efeito de *blur* não resultou numa melhoria notável do modelo. No entanto, acredita-se que, se houvessem mais *epochs*, o modelo pederia ter atingido uma *val\_accuracy* superior, dado que não se observou *overfitting* com 100 *epochs*.

## X. MODELO 5

No último modelo foi decidido aplicar um processamento às imagens semelhante ao modelo anterior IX destacando a face das pessoas mas, invés de aplicar o efeito de *blur* no resto da imagem, cortar apenas a face, ignorando assim tudo o resto por completo. Este processamento foi explicado com mais detalhe na secção IV.

### A. Discussão de resultados

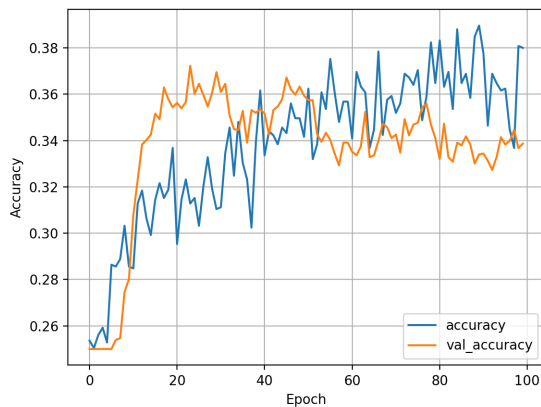


Fig. 18. Accuracy do modelo 5 em função do número de epochs

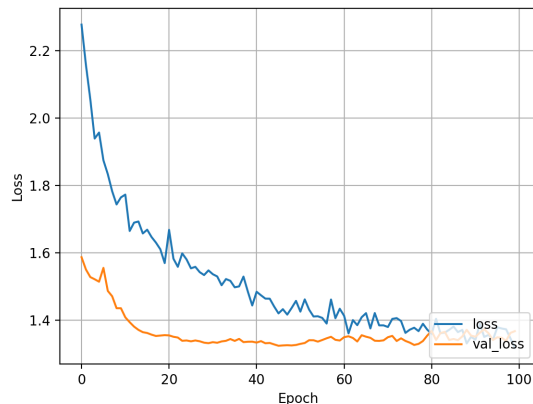


Fig. 19. Loss do modelo 5 em função do número de epochs

Através da figura 18 é possível verificar uma subida repentina dos valores de *val\_accuracy* nas primeiras *epochs*, atingido o seu máximo de 37.23% na *epoch* 23. De seguida, os valores foram descendo constantemente, ao contrário da *accuracy* que continuou a subir.

O valores relativos ao *val\_loss*, desceu mais acentuadamente nas primeiras *epochs*, tendo ficado relativamente constante a partir da *epoch* 20.

## B. Conclusão

Através dos resultados obtidos, é possível verificar que este modelo teve um comportamento semelhante ao modelo 3 VIII, em que se aplicou um filtro de cinzentos. Tal como o modelo 3, houve *overfitting* depois de se ter atingido o valor máximo de *val\_accuracy*, uma vez que este começou a descer constantemente, ao contrário do *accuracy* que continuou a subir, como seria de esperar.

Concluiu-se que o método de focar apenas na face das pessoas para prever a sua faixa etária não resulta num melhor modelo, uma vez que muita da informação como o cabelo, barba e outros atributos físicos são perdidos. Assim, o valor máximo da *val\_accuracy* diminuiu 2.26% relativamente ao modelo 2.

## XI. APLICAÇÃO DOS MODELOS

Tendo analisado todos os modelos, concluiu-se que o modelo mais bem sucedido foi o modelo 2, que não continha qualquer tipo de filtros para a alteração das imagens do *dataset*. Posto isto, decidiu-se utilizar este modelo treinado para classificar imagens individuais e verificar a veracidade dos resultados. Para tal, utilizou-se a função *predict*, da biblioteca *Tensorflow* para, utilizando o modelo, avaliar uma determinada imagem. Esta função irá retornar um *array* com o valor de previsão de cada classe, para esta fotografia.

```
from tensorflow.keras.models import load_model
import tensorflow as tf
import numpy as np
import cv2
import os
```

```
dir = '/home/tomasfilipe7/Desktop/Universidade/4_Ano/2_Semestre/AA/age_guesser/'
model_loaded = load_model(os.path.join(
    dir, 'best_model.hdf5'))
test_image=cv2.imread(os.path.join(dir,
    "test5.jpg"))
test_image = tf.image.resize(test_image,
    [200,200])
test_image = np.expand_dims(test_image,
    axis=0)
print(model_loaded.predict(test_image))
result=np.argmax(model_loaded.predict(
    test_image))
if(result == 0):
    print("Result: Young")
elif(result == 1):
```

```

    print("Result: Middle1")
elif(result == 2):
    print("Result: Middle2")
elif(result == 3):
    print("Result: Middle3")
else:
    print("Result: Old")

```

Da execução deste programa, o resultado foi o esperado, com algumas imagens a obterem o resultado positivo e outras um resultado negativo.

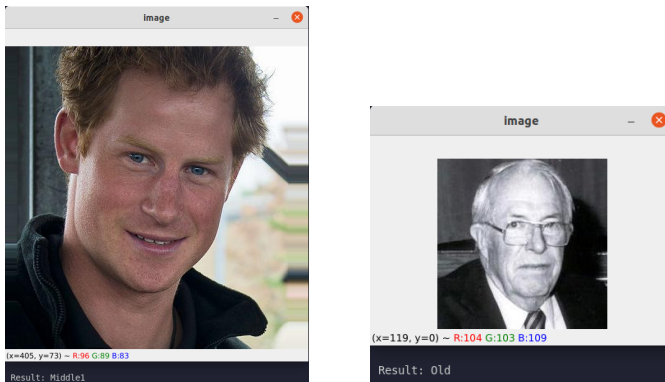


Fig. 20. Exemplo de imagens com resultado de previsão positivo, seguindo o nosso modelo 2.

## XII. CONCLUSÕES

O objetivo do presente trabalho foi a análise das diversas abordagens para o problema da previsão da idade de pessoas através das suas imagens, estudadas por terceiros, bem como a elaboração de diversos modelos de *machine learning* que utilizariam diversos efeitos e processamento de imagem para alterar os *dataset* e fazer conclusões consoante os resultados obtidos. Este objetivo foi concluído, uma vez que se analisaram e foram elaboradas conclusões de diversos projetos estudados que moldaram o rumo tomado na elaboração dos modelos. Foram aplicados diversos efeitos e processamento de imagem estudados não só na disciplina de *Aprendizagem Automática* como também em *Computação Visual* e aplicados aos *datasets* utilizados nos diversos modelos.

Após a análise feita aos resultados obtidos nos diversos modelos, é possível concluir que houve uma grande subida na precisão do modelo quando se alterou o parâmetro *learning\_rate* do *optimizer*, resultando num aumento de 5.23%, o que foi bastante positivo e que mostrou a importância da variação dos parâmetros do modelo, neste caso, o *optimizer*.

É também possível concluir que a utilização dos filtros abordados não resultou, necessariamente, num aumento da precisão dos modelos, uma vez que este processamento elimina informação e foca as imagens em aspetos específicos. Observou-se ainda que, ao restringir a informação ao modelo, houveram casos em que este começou a sofrer de *overfitting* nas *epochs* mais altas, o que não aconteceu com os modelos que utilizaram o *dataset* original.

Acredita-se que foi especialmente difícil o treino dos modelos com o *dataset* utilizado, uma vez que este era uma compilação das fotografias de celebridades da *WIKI* que continha imensas fotografias indevidas, onde não se encontravam quaisquer pessoas ou, em muitos mais casos, as pessoas encontravam-se distantes ou numa posição lateral, que dificultava a percepção da sua idade mesmo para um ser humano. No entanto, o *dataset* utilizado era composto por milhares de imagens o que poderá ter resultado numa resistência ao *overfitting* nos diversos modelos.

Contudo, conclui-se que a abordagem de classificação das pessoas pelas faixas etárias resultou melhor do que as técnicas de regressão utilizadas por alguns trabalhos estudados que obtiveram uma previsão de idades muito limitada nas idades médias.

Este projeto mostrou que o presente problema é complexo, desafiante, pertinente e com imensas potenciais aplicações e concluiu-se que a precisão alcançada foi algo positiva.

## XIII. TRABALHO FUTURO

Como trabalho futuro, considerou-se que seria interessante observar o comportamento dos modelos desenvolvidos que não apresentaram *overfitting* com mais *epochs*, tal como foi feito no estudo 2 II-B onde se utilizaram até 4000 *epochs*. Infelizmente, isto não foi possível ser feito no presente projeto, visto que cada modelo demorou cerca de 4 horas a ser treinado, e houve uma limitação computacional por ser apenas possível utilizar 1 computador para o treino dos modelos. Esta limitação é um problema clássico das *Neural Networks* que poderá ser atenuado com a evolução da tecnologia computacional.

Seria também interessante aplicar algoritmos de processamento de imagem para obter pontos faciais nas fotografias, resultando em vários *features* que poderiam ser bastante relevantes. Isto foi feito no estudo 2 II-B.

Acredita-se que os resultados podem também ser melhorado através da utilização de outro *dataset* que contenha imagens mais claras das faces das pessoas, sem que estas estejam muito longe ou em posições laterais.

A utilização de mais *layers* e diferentes *optimizers* pode, claramente, resultar em modelos mais capazes.

## XIV. CONTRIBUIÇÕES

Em termos das contribuições para o projeto, ambos os elementos contribuíram igualmente para o estudo, desenvolvimento e análise dos resultados.

## XV. REFERENCES

### REFERENCES

- [1] <https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>.
- [2] Pétia Georgieva, Machine Learning lecture 4: Neural Networks.
- [3] <https://machinelearningmastery.com/check-point-deep-learning-models-keras/>.
- [4] <https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/>.
- [5] Gabriel Atkin, Age prediction from images, <https://www.kaggle.com/gcdatkin/age-prediction-from-images-cnn-regression>

- [6] Maria Fretescu, Age prediction dataset, <https://www.kaggle.com/mariafrenti/age-prediction>.
- [7] Sarah N. Kohail, Using artificial neural network for human age estimation based on facial images, <https://ieeexplore.ieee.org/document/6207735>.
- [8] Bor-Chun Chen, Chu-Song Chen, Winston Hsu, Cross-Age Reference Coding for Age-Invariant
- [9] Vadim Pisarevsky, [github.com/vpisarev](https://github.com/vpisarev), 2013 [github.com/vpisarev/opencv/blob/master/data/haarcascades/haarcascade\\_frontalface\\_default.xml](https://github.com/vpisarev/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml)
- [10] UTKFace dataset, <https://susanqq.github.io/UTKFace/>
- [11] Shreyansh Joshi, Facial-Demographics-using-CNN, <https://github.com/ShreyanshJoshi/Facial-Demographics-using-CNN>
- [12] Zakariya Qawaqneh, Arafat Abu Mallouh, Buket D. Barkana, Deep Convolutional Neural Network for Age Estimation based on VGG-Face Model
- [13] Pranzal Saxena, Designing an Age Classification Model with Deep Learning, <https://heartbeat.fritz.ai/designing-age-classifier-model-with-deep-learning-a346028d0530>
- [14] <https://stackoverflow.com/questions/63248277/how-to-use-black-and-white-images-in-keras-cnn>
- [15] <https://stackoverflow.com/questions/55572325/how-can-i-use-neutral-network-with-gray-scale-images-in-keras>
- [16] <https://www.tensorflow.org/tutorials/images/cnn>
- [17] <https://www.bmc.com/blogs/keras-neural-network-classification/>
- [18] <https://keras.io/api/preprocessing/image/>