Together for Tomorrow!
Enabling People
Education for Future Generations

AI Course

Team Project Final Report

For students (instructor review required)

Samsung Innovation Campus

Top 5 Brands Similarities

Date: 30/09/2021

CLIck

Catarina Bento
Inês Costa
Luisa Trigo da Roza

Content

1. Introduction

O curso de Inteligência Artificial da NOVA School of Science and Technology patrocinado pela Samsung deu-nos a oportunidade de mergulhar no Projeto Capstone em grupo. Este curso oferece u-nos uma aprendizagem profunda em Inteligência Artificial, através das suas próprias implementações em Machine Learning e Deep Learning. Mais tarde, na fase em que nos encontramos, os professores desafiaram-nos para um projecto final com o objectivo de nos fazer compreender e obter experiência prática com o processo e conceção, implementação e comunicação dos resultados.

Sendo que a identidade de uma marca se baseia na sua imagem, esta representa uma oportunidade para o seu crescimento e distinção no mercado. Vivendo num mundo em que todos os dias surgem muitas ideias e projetos novos, a quantidade de marcas existentes em Portugal foi algo que nos chamou à atenção. De facto, só em Portugal existem 3 070 031 marcas diferentes, e muitas delas com nomes semelhantes. Para além da semelhança de nome, existe também uma enorme semelhança entre as imagens destas mesmas marcas, desde a forma, desenho à cor. Consequentemente, existem vários efeitos negativos para estas marcas, não só para a sua identidade e a sua diferenciação como juridicamente. Deste modo, pensámos que seria interessante elaborar algum tipo de solução para este problema com recurso a Inteligência Artificial.

O nosso objetivo era elaborar um algoritmo que, através de uma base de dados, mostrasse um top 5 de marcas mais semelhantes à que se pretende registar.
Por esta razão, pretendemos expor o método mais eficiente para o reconhecimento de semelhanças entre uma imagem e/ou respetivo nome: em primeiro lugar apresentando um Top 5 dos nomes mais parecidos e, em segundo lugar, um Top 5 das imagens mais parecidas com a da marca inserida.

Inicialmente, fizemos uma previsão de agenda para o mês de setembro da seguinte forma:

- de 2 a 7 setembro: preparar o *Dataset*
- de 8 a 9 setembro: Processamento dos dados
- de 10 a 23 setembro: *Coding* (Treino e Teste)
- de 24 a 27 setembro: escrever o *Report*
- de 27 a 29 setembro: fazer a *Final Presentation*

No entanto, devido a obstáculos e desafios na elaboração do nosso código alguns destes agendamentos foram prolongados, nomeadamente para o *Coding*, e outros obrigatoriamente encurtados. Tendo em conta as dificuldades que fomos encontrando na realização de algumas fases do código para as imagens, vimo-nos obrigadas a não realizar o Top 5 de nomes mais semelhantes.

Em relação à distribuição de tarefas, todos os membros do grupo tiveram participação em todas as secções do projecto.

2. Project Execution

2.1 Data Acquisition

2.2 Training Methodology

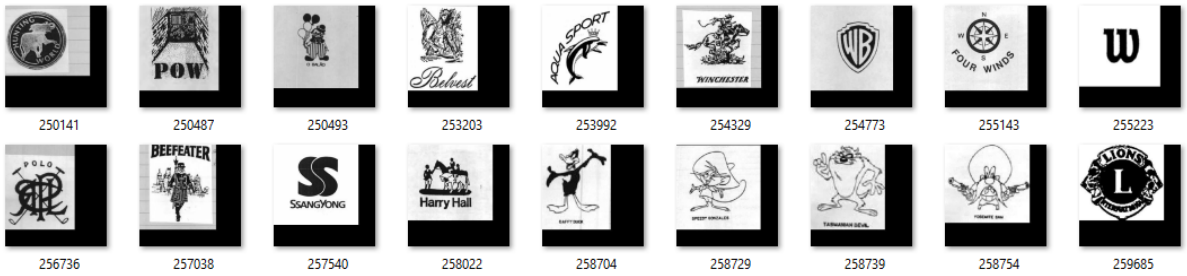
2.3 Workflow

2.4 System Diagram

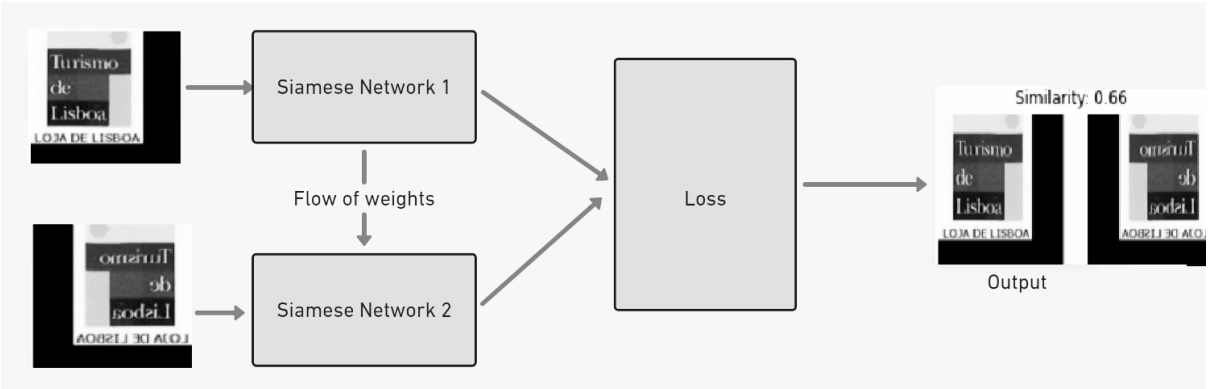
A execução do nosso projecto desenrolou-se durante todo o mês de setembro, com vários ob stáculos pelo percurso.

O primeiro passo foi a recolha de dados. Retirado da plataforma TMView, uma base de dados que contém todas as marcas registadas na União Europeia, o nosso *dataset* reúne 2000 mar cas distintas, registadas em Portugal, da classe 25 - Vestuário, Calçado e Chapelaria.
Para a elaboração do mesmo, foi criado um ficheiro CSV onde constam o nome das marcas e número da respetiva imagem e, ainda, uma pasta com as mesmas.

De seguida, passamos ao *Pre-Processing* das imagens. Colocamo-las todas com a mesma di mensão (299x299) através da técnica de Padding e convertemo-las para Grayscale (não perd endo qualquer informação).



Após essa fase e depois de uma profunda pesquisa, estabelecemos que a *Siamese Neural N etwork* era a mais adequada para o nosso projeto. Esta rede é uma arquitetura de redes neur onais que contém duas ou mais sub-redes idênticas com a mesma configuração, parâmetros e pesos. Ao alterar os pesos de uma, os pesos da outra também são modificados.



System Diagram

Em relação à arquitetura de cada rede, utilizamos as seguintes layers:

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 299, 299, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 299, 299, 64)      640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 149, 149, 128)     73856
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 149, 149, 128)     147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 74, 74, 128)       147584
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 74, 74, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 37, 128)       0
_________________________________________________________________
global_average_pooling2d (Gl (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 48)                6192
=================================================================
Total params: 523,440
Trainable params: 523,440
Non-trainable params: 0
```

A camada *Convolutional*, abreviada para **conv2D**, percorre a imagem recebida como input e e stipula valores para cada pixel da mesma.

A camada *Max Pooling* reduz a dimensão da imagem, escolhendo apenas o maior valor dentr o do pool_size definido.

Por fim, utilizamos uma camada *Lambda* para calcular a distância euclidiana entre os outputs das duas redes e, logo de seguida, uma camada *Dense* com a função de ativação *Sigmoid* qu e coloca o valor de similaridade, dado pela camada anterior, entre 0 e 1.

```
Model: "model_3"
__________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
==========================================================================================
input_5 (InputLayer)         [(None, 299, 299, 1)      0
__________________________________________________________________________________________
input_6 (InputLayer)         [(None, 299, 299, 1)      0
__________________________________________________________________________________________
model_2 (Functional)         (None, 48)                523440      input_5[0][0]
                                                                   input_6[0][0]
__________________________________________________________________________________________
lambda (Lambda)              (None, 1)                 0           model_2[0][0]
                                                                   model_2[1][0]
__________________________________________________________________________________________
dense_3 (Dense)              (None, 1)                 2           lambda[0][0]
==========================================================================================
Total params: 523,442
Trainable params: 523,442
Non-trainable params: 0
__________________________________________________________________________________________
```

Para compilar o nosso modelo, utilizamos como *Loss Function* a Binary Cross-Entropy, uma v ez que já se encontra implementada. Esta função calcula a diferença entre o valor previsto e o valor real. Apesar de esta não ser a mais indicada para o nosso problema, não foi possível ex ecutar de raiz a Triplet Loss, tendo em conta o curto prazo de entrega.

Para treinar a rede, fizemos uma função que constrói pares de imagens que são semelhantes e diferentes, isto é, pares positivos e negativos, respetivamente. Para isso, passamos à próxi ma etapa que consistiu em *Data Augmentation.* Realizámos uma série de operações (Stretchi ng, Flipping e Rotation 45º e 90º) às nossas imagens para podermos formar os pares positivo s.

Para além disso, para a rede aprender se as duas imagens são semelhantes ou não, fizemos um array com as respetivas *labels* dos pares, onde '0' representa um par negativo e '1' um par positivo.

```
In [31]: pairs.shape #[[imgA, imgB], ...]
Out[31]: (20000, 2, 299, 299, 1)

In [32]: labels.shape #[[0], [1], ...]
Out[32]: (20000, 1)
```

WORKFLOW:

- ➔ Read Data
- ➔ Pre-Processing Data (Padding & Grayscale)
- ➔ Data Augmentation (Stretching, Flipping & Rotation 45º/90º)
- ➔ Normalizing Data (/255)
- ➔ Add Dimension (from 2D to 3D)
- ➔ Make Pairs
- ➔ Split Dataset (Train & Test)
- ➔ Build Network (Siamese)
- ➔ Compile & Train (fit)
- ➔ Test
- ➔ Predict Top5

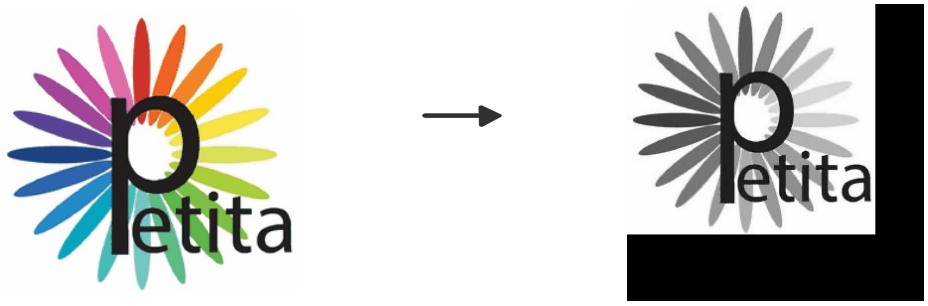
3. Results

3.1. Data Preprocessing

No pré-processamento, tal como mencionado anteriormente, as imagens foram colocadas todas à mes ma dimensão (padding) e convertidas para o modo grayscale.

```
#grayscaling
image = Image.open(img).convert('L')
```

```
#add padding
image = cv2.copyMakeBorder(image_arr, 0, maxHeight - image_arr.shape[0], 0,
                           maxWidth - image_arr.shape[1], cv2.BORDER_CONSTANT, None, value = 0)
```

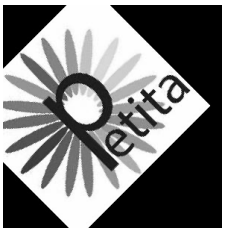


De seguida, realizamos algumas operações para fazer *Data Augmentation*:

➔ Rotate 90º & 45º:

```
#rotate image
image_rot = Image.open(img).rotate(90)
```

```
#rotate image
image_rot = Image.open(img).rotate(45)
```



➔ Stretch:

```
#stretching image
new_height = 200
dim_size = (new_height, np.array(image_gray).shape[1]) #new dimensions
image_stretch = cv2.resize(np.array(image_gray), dim_size)
```

➔ Flip:

```
#flip image
image_flip = Image.open(img).transpose(Image.FLIP_LEFT_RIGHT)
```

Para normalizar as imagens:

```
images = np.array([np.array(Image.open(fname)) /255 for fname in filelist])
```

Isto permitiu-nos representar os pixeis em valores num intervalo de [0,1], assegurando que todos tinha m uma distribuição semelhante.

Também alterámos a dimensão de cada imagem antes de as passar para a SNN (Siamese Neural Netwo rk)

```
In [33]: images[0].shape
Out[33]: (299, 299)

In [34]: images3Dim[0].shape
Out[34]: (299, 299, 1)
```

3.2 Exploratory Data Analysis (EDA)

O nosso *dataset* original é composto por 2000 linhas e 2 colunas que representam o nome da marca e o número correspondente à sua imagem, respetivamente.

```
df = pd.read_csv('marcas.csv', names=['Number','Name'])
```

```
df.head(5)
```

	Number	Name
0	250141	HUNTING WORLD;
1	250487	POW;
2	250493	O BALAO;
3	253203	BELVEST;
4	253992	AQUA SPORT;

```
df.shape
```

```
(2000, 2)
```

Para além disso, podemos ainda observar que não existe qualquer marca repetida.

```
df[df.duplicated()]
```

Number	Name

```
df[df.duplicated()].shape
```

```
(0, 2)
```

Uma vez que não temos quaisquer valores numéricos, não faz sentido avaliarmos o *dataset* estatistica mente.

3.3 Modeling

Para treinar a nossa Rede Siamesa, utilizámos dois callbacks:

- ➔ Early Stopping, para parar o treino quando já não há mais evolução ao fim de um número estipulado de épocas (3, no nosso caso);
- ➔ Model Checkpoint, para salvar o melhor modelo tendo em conta o respetivo *monitor* (' val_loss', no nosso caso), podendo ser usado mais tarde.

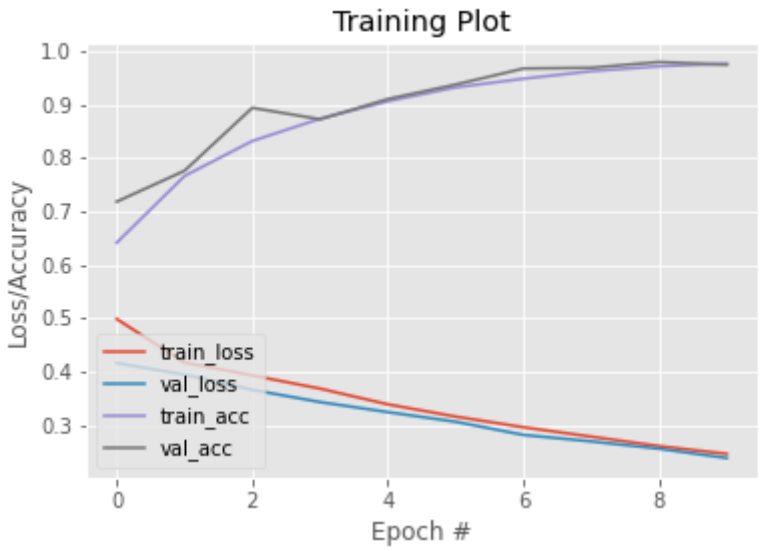
```
# callbacks
early = EarlyStopping(monitor='val_loss', patience=3)
path = 'siamese.h5'
checkpoint = ModelCheckpoint(filepath= path, monitor= 'val_loss', save_best_only= True)
```

O nosso modelo foi treinado com:

- ➔ Batch Size = 128, uma vez que tínhamos muitos exemplos para treino;
- ➔ Número de Épocas = 10, porque a *accuracy* já estava suficientemente alta e um valor mais alto implicaria um maior tempo dispendido neste processo.

Para referência, estes parâmetros levaram a uma duração de cerca de 24 horas de treino.

A evolução dos resultados obtidos neste processo foi a seguinte:



3.4 User Interface (Interface).

O nosso projeto foi elaborado no *Jupyter Notebook* e não foi implementada outra interface.
O input a ser inserido pelo utilizador é obtido através da mudança da diretoria.

3.5. Testing and Improvements.

Após o treinar o modelo, estes foram os resultados que obtivemos:

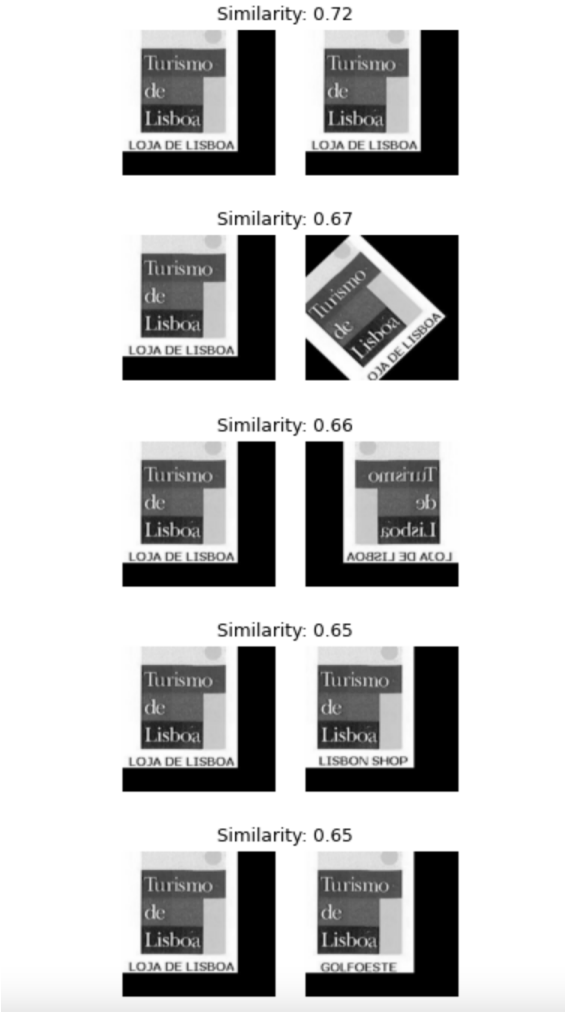
```
# evaluate model
score = model.evaluate([testA, testB], testLabels, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
188/188 [==============================] - 981s 5s/step - loss: 0.2427 - accuracy: 0.9768
Test loss: 0.2427113652229309
Test accuracy: 0.9768333435058594
```

Para chegar a estes resultados, alterámos:

- ➔ O número de camadas *Convolutional* + *Max Pooling*, aprofundando o nosso modelo. Fo mos colocando séries de duas *Convolutional* mais uma *Max Pooling*, para que a rede não ficasse demasiado simples, tendo em conta a grandeza do nosso problema;
- ➔ O pool_size, passando de (3,3) para (2,2);
- ➔ O batch_size, passando de 64 para 128, pois tínhamos bastantes exemplos para trein o;
- ➔ O número de épocas, porque estava a demorar muito tempo a correr e os resultados j á estavam bastante bons ao fim de 10 épocas.

Por fim, implementámos uma função que nos mostrass e o top 5 das marcas mais parecidas com a inserida pel o utilizador.
Nesta função fizemos o predict entre o input e todas as outras imagens do nosso dataset e depois através do va lor de similaridade dado, encontrámos as 5 imagens ma is parecidas.



4. Projected Impact

4.1. Accomplishments and Benefits

Os benefícios deste projeto centram-se no facto da operação ser automatizada, o que conduz à diminui ção de erros que poderiam ser cometidos ao verificar manualmente todas as marcas já registadas. Para além disso, os resultados são obtidos mais rapidamente.

Se fosse futuramente implementado, este programa aumentaria a produtividade e a eficiência dos func ionários e, assim, estes poderiam focar-se em tarefas que só podem ser realizadas manualmente.

4.2 Future Improvements

Devido ao curto prazo, vimo-nos obrigadas a eliminar parte do nosso objetivo inicial: o top 5 de nomes de marcas mais semelhantes.

Em relação à *Loss Function,* poderíamos implementar de raiz a *Triplet Loss*, possivelmente melhorando os resultados do nosso projeto.

Do mesmo modo, poderíamos expandir o nosso programa a uma aplicação mobile ou um website que permitisse ao utilizador inserir a ideia da sua marca a registar e, assim, obter o resultado diretamente.

Relativamente ao nosso *dataset*, poderíamos evidentemente aumentá-lo. Assim, não se trataria apenas da classe 25 nem apenas de marcas registadas em Portugal.

Outro aprimoramento do nosso código seria reduzir a dimensão das imagens para o processo de treino ser mais rápido.