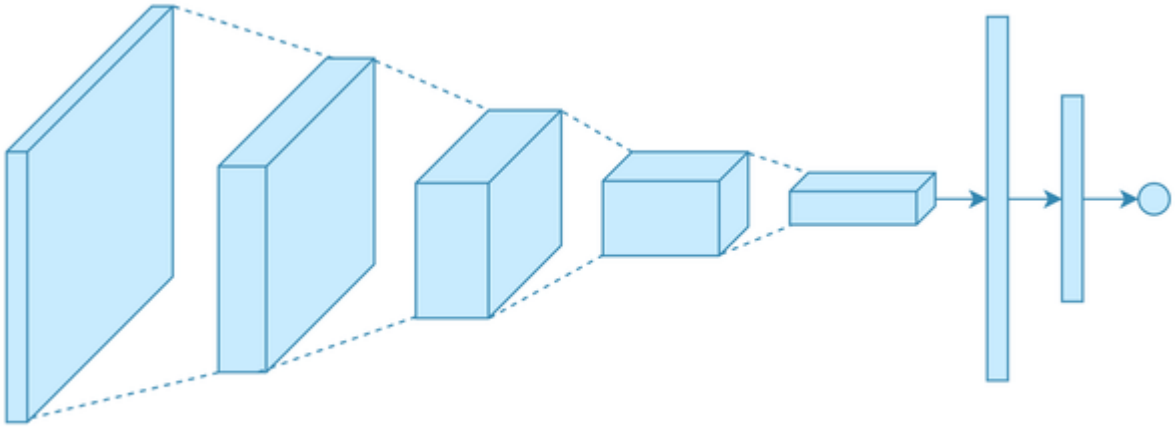
TÉCNICO LISBOA

Homework 2

Final Report



Deep Learning

Instituto Superior Técnico

Beatriz Cavaleiro de Ferreira | 73802

Pedro Zenário | 102348

Ricardo Fraga Simões | 93674

Vicente Sobral | 102134

31 de janeiro de 2022

Table of Contents

Introduction

The purpose of this project is to develop classification tasks, delving further into the topics of Convolutional Neural Networks (CNN) and the procedures of image captioning. These exercises allowed us to get in touch with new datasets and to apply the concepts and models learnt in class fully by ourselves.

Question 1

Exercise 1.1

Input Image: 28x28x3

Convolutional Layer:

- $Output\ width = \frac{input\ width - kernel\ width + 2 \times padding\ width}{stride} + 1 = \frac{28-5+2\times 0}{1} + 1 = 24 = Output\ height$ (input, filter and padding are symmetric)

- $Output\ depth = \#\ filters = 8$

- $\#Parameters\ =$ inputs $\times$ outputs + biases $= inputs \times (\#filters) \times (filter\ size) + (\#\ filters) = 3 \times 8 \times 5 \times 5 = 608$

Max Pooling Layer:

- $Output\ width = \frac{input\ width - kernel\ width + 2 \times padding\ width}{stride} + 1 = \frac{24-4+2\times 0}{2} + 1 = 11 = Output\ height$ (input, filter and padding are symmetric)

Linear Layer:

- Flatten before the linear transformation, $shape = 11 \times 11 \times 8 = 968$

- $\#Parameters = Ax + b\ (affine\ operation) = 10 * 968 + 10 = 9\,690$

Total $\#Parameters = 608 + 9\,690 = 10\,298$

We can confirm this result using functions from Keras package

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 24, 24, 8)	608
max_pooling2d_4 (MaxPooling 2D)	(None, 11, 11, 8)	0
flatten_4 (Flatten)	(None, 968)	0
dense_4 (Dense)	(None, 10)	9690

Total params: 10,298
Trainable params: 10,298
Non-trainable params: 0

Table 1

Exercise 1.2

1st Layer:

- Flatten before the linear transformation, $shape = 28 \times 28 \times 3 = 2\,352$
- $\#Parameters = Ax + b\ (affine\ operation) = 100 * 2\,352 + 100 = 23\,530$

2nd Layer:

- $\#Parameters = Ax + b\ (affine\ operation) = 10 * 100 + 10 = 1\,010$

Total $\#Parameters = 23\,530 + 1\,010 = 24\,540$

$$increase = \frac{24\,540 - 10\,298}{10\,298} \times 100\% = 138.3\%$$

.

This shows one of the many advantages of using CNNs (with convolution and pooling layers) instead of fully connected layers. While in a fully connected all activations depend on all inputs, in a CNN this does not happen due to the filters used. For that reason, there are a many more parameters to be learned in the fully connected network, so a CNN is a much more viable solution, especially from the computational point of view.

Exercise 1.3

Exercise 1.3.1

Let's consider the self-attention mechanism applied for a sequence of length L. In addition, let $X \in \mathbb{R}^{L \times n}$ be the input matrix for this sequence, where n is the embedding size.

We can start by initializing a set of weights for the **keys**, a set of weights for the **queries** and a set of weights for the **values.**

$$W_K{}^{(h)} \in \mathbb{R}^{n \times d}, W_Q{}^{(h)} \in \mathbb{R}^{n \times d} \; and \; W_V{}^{(h)} \in \mathbb{R}^{n \times d}, h \in \{1,2\}$$

We derive the **key, query** and **value** by considering the dot product between the input matrix X by the respective weight matrix,

$$K^{(h)} = X.W_K{}^{(h)}, K^{(h)} \in \mathbb{R}^{L \times d}, h \in \{1,2\}$$

$$Q^{(h)} = X.W_Q{}^{(h)}, Q^{(h)} \in \mathbb{R}^{L \times d}, h \in \{1,2\}$$

$$V^{(h)} = X.W_V{}^{(h)}, V^{(h)} \in \mathbb{R}^{L \times d}, h \in \{1,2\}$$

Now we can compute the attention scores S and then convert these scores to probabilities P via softmax transformation (row-wise),

$$S^{(h)} = Q^{(h)}\left(K^{(h)}\right)^T, S^{(h)} \in \mathbb{R}^{L \times L}, h \in \{1,2\}$$

$$P^{(h)} = softmax\left(S^{(h)}\right), P^{(h)} \in \mathbb{R}^{L \times L}, h \in \{1,2\}$$

In fact,

$$P^{(h)} = softmax\left(S^{(h)}\right) = softmax\left(Q^{(h)}\left(K^{(h)}\right)^T\right) = softmax\left(XW_Q{}^{(h)}\left(XW_K{}^{(h)}\right)^T\right) =$$

$$= softmax\left(XW_Q{}^{(h)}(W_K{}^{(h)})^T X^T\right) = \; softmax(XAX^T), h \in \{1,2\}$$

After showing that the self-attention probabilities for each head could be written in the required form, we conclude that:

$$A = {W_Q}^{(h)}({W_K}^{(h)})^T, A \in \mathbb{R}^{n\times n}, h \in \{1,2\}$$

We can find in the Literature[1] that:

1. Rank of a matrix $A_{m\times n}$, $\rho(A_{m\times n}) \leq Min(m,n)$

2. If $\rho(A) = m$ and $\rho(B) = n$ then $\rho(AB) \leq min(m,n)$

By hypothesis,

- ${W_Q}^{(h)}, {W_Q}^{(h)}, {W_Q}^{(h)} \in \mathbb{R}^{n\times n}$ for $h \in \{1,2\}$, with $d \leq n$

So we conclude that,

$$\begin{cases} \rho\left({W_Q}^{(h)}\right) \leq d \\ \rho\left({W_K}^{(h)^T}\right) \leq d \end{cases} \Rightarrow \rho(A) \leq d$$

Exercise 1.3.2

From the previous question, we found that

$$P^{(h)} = softmax\left(X{W_Q}^{(h)}({W_K}^{(h)})^T X^T\right), h \in \{1,2\}$$

Starting with the first attention head,

$$P^{(1)} = softmax\left(X{W_Q}^{(1)}({W_K}^{(1)})^T X^T\right)$$

For the second attention head,

$$P^{(2)} = softmax\left(X{W_Q}^{(2)}\left({W_K}^{(2)}\right)^T X^T\right)$$

[1] https://blogmedia.testbook.com/blog/wp-content/uploads/2016/09/Rank-of-a-Matrix-and-Its-Properties-GATE-Study-Material-in-PDF.pdf

But since

$$W_Q{}^{(2)} = W_Q{}^{(1)} B$$

$$W_K{}^{(2)} = W_K{}^{(1)} (B^{-1})^T$$

$$P^{(2)} = softmax\left(XW_Q{}^{(1)} B\left(W_K{}^{(1)} (B^{-1})^T\right)^T X^T\right) = softmax\left(XW_Q{}^{(1)} BB^{-1}\left(W_K{}^{(1)}\right)^T X^T\right) =$$

$$= softmax\left(XW_Q{}^{(1)} I\left(W_K{}^{(1)}\right)^T X^T\right) = softmax\left(XW_Q{}^{(1)}\left(W_K{}^{(1)}\right)^T X^T\right) = P^{(1)}$$

So the probabilities are the same, as we wanted to prove.

Question 2

Exercise 2.1

They convolutional layers[2] – a key component of Convolutional Neural Networks (CNNs), can fancy a property called parameter sharing. This is a method of regularization where only a subset of parameters needs to be learned (and stored in memory). This particular property causes the layers to have another property called equivariance (to translation). In simple terms, to say a function is equivariant means that if the input changes, then the output changes in the same way. In mathematical terms:

$f(x)$ is equivariant to a function g if $f\big(g(x)\big) = g(f(x))$

Considering a convolutional layer, if g is a function that applies a translation to an object/input, the convolutional layer is equivariant to g. This is not true if the operation is rotation or scale changes; this particular kind of operations require other mechanisms.

Exemplifying, if g shifts each pixel of the image one pixel to right and then we apply convolution, the result will be the same as applying convolution first and then the shift operation g.

In image processing this property is extremely useful. Since there is parameter sharing, an object will be detected independently of its position in the image, which is it's something that will, most likely, happen in image classification problems of large datasets.

2 Goodfellow, I., Bengio, Y. and Courville, A., 2017. Deep learning. Cambridge, Mass: The MIT Press.

Exercise 2.2

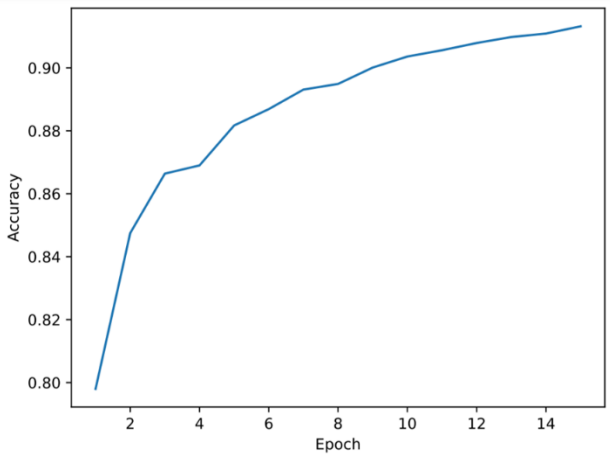


Figure 1

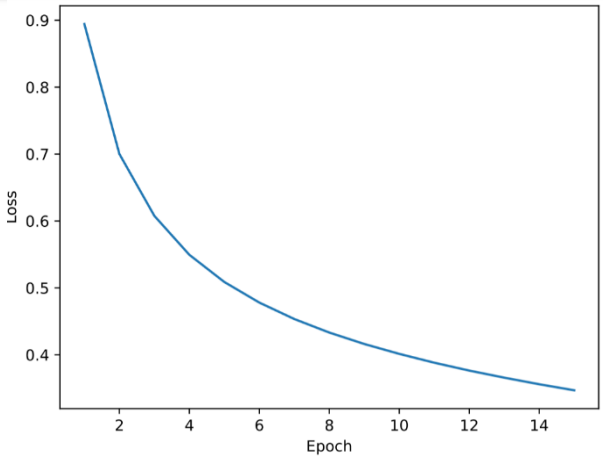


Figure 2

2.2	dropout prob	learning_rate	Training loss	Valid Acc	Final Test Acc	Ranking
	0.5	0.01	0.3469	0.9132	0.9073	1
	0.4	0.01	0.3349	0.9121	0.9080	2
	0.2	0.01	0.3195	0.9113	0.9051	3
	0.3	0.01	0.3268	0.9111	0.9060	4
	0.6	0.01	0.3602	0.9098	0.9032	5
	0.1	0.01	0.3107	0.9095	0.9076	6
	0.8	0.01	0.4086	0.9064	0.9020	7
	0.3	0.1	0.2542	0.9016	0.8986	8
	0.5	0.1	0.3002	0.8906	0.8876	9
	0.8	0.1	0.4494	0.8648	0.8635	10
	0.3	0.001	0.6465	0.8572	0.8517	11
	0.5	0.001	0.6773	0.8526	0.8475	12
	0.8	0.001	0.7809	0.8313	0.8237	13

Table 2

Exercise 2.3

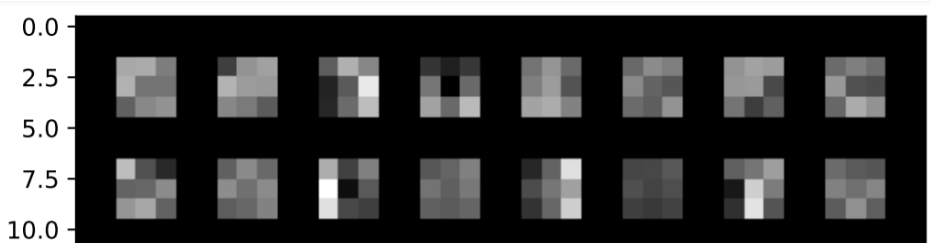


Figure 3

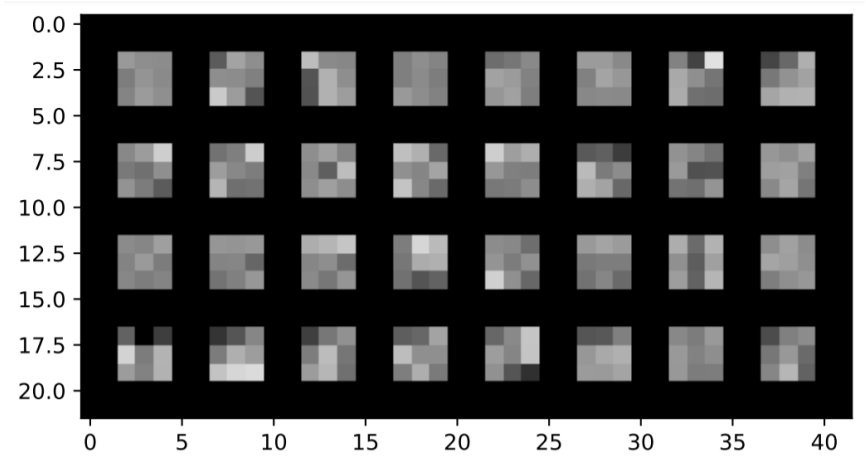


Figure 4

Each filter acts as a detector for a particular feature. The first layer filters mostly detect directions, colors, edges and simple shapes, so the results are not very detailed, as we can observe in Figure 5. These direction and color filters then get combined into basic grid and spot textures, so a little bit more detail becomes visible, like is shown in Figure 6.

As we go deeper into the network, the filters build on top of each other, and learn to encode more complex patterns. For example, in Figure 7, we can see a bird detector, where we observe multiple heads in different orientations, because the location of the bird in the image is not important, as long as it appears somewhere in the filter will activate.

That's why the filter tries to detect the bird head in several positions by encoding it in multiple locations in the filter. This clearly illustrates that the model learns very detailed patterns near the output, i.e., in the final layers of the model, whereas more global and abstract ones are learnt in the early layers.

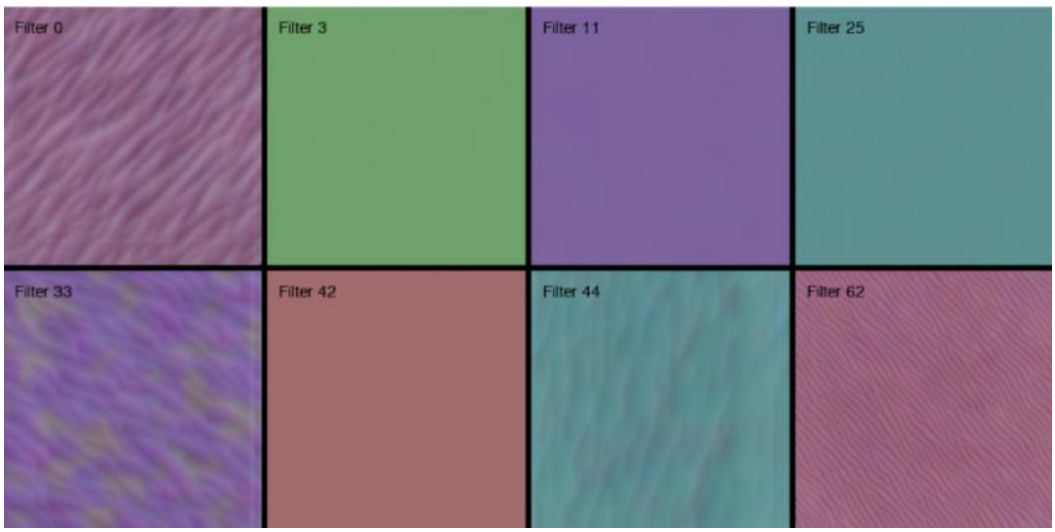


Figure 5

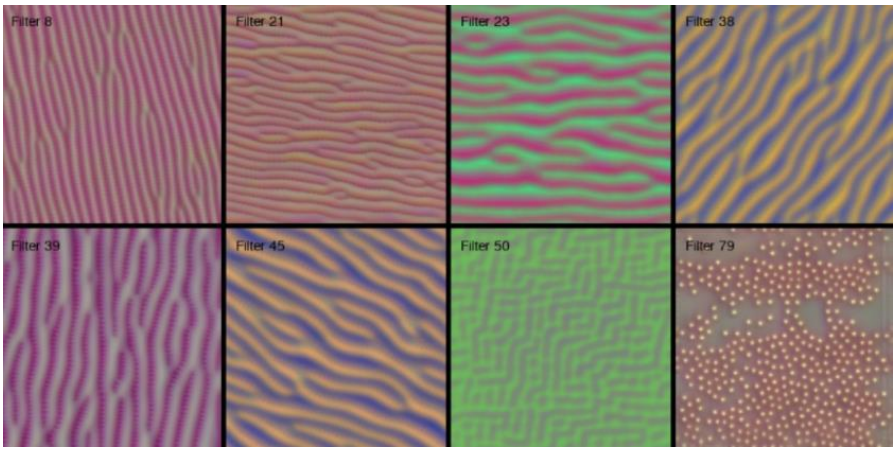


Figure 6

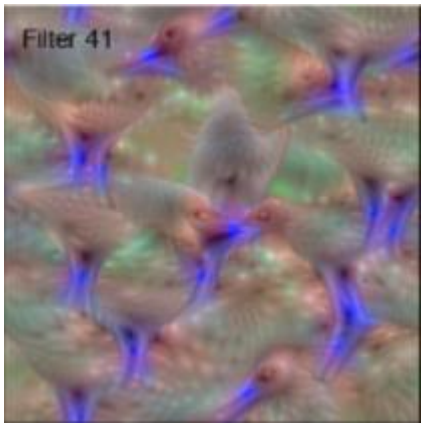


Figure 7

Question 3

Exercise 3.1 a)

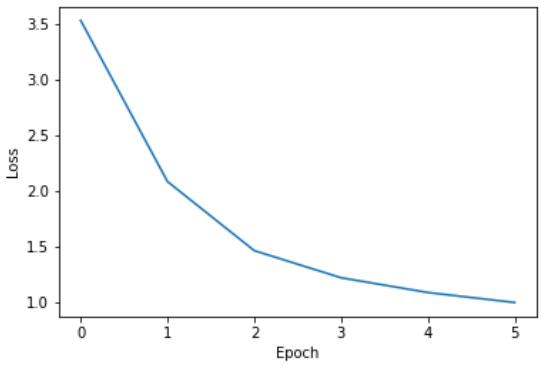


Figure 8

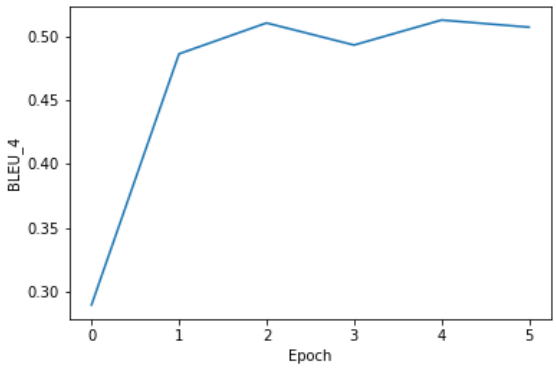


Figure 9

The final test Bleu-4 value was 0.4981. The number can be observed in Figure 10.

```
INFO: Test Bleu-4: 0.4981077727904016
```

Figure 10

Exercise 3.1 b)

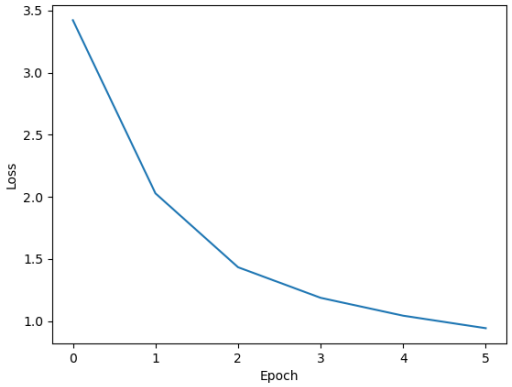


Figure 11

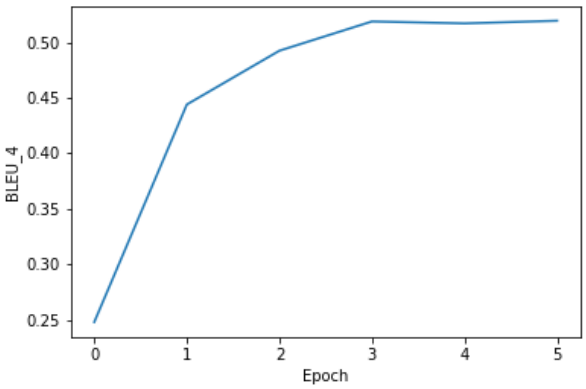


Figure 12

The final test Bleu-4 value was 0.5142. The number can be observed in Figure 13.

```
INFO: Test Bleu-4: 0.5142337254319148
```

Figure 13

Exercise 3.1 c)

The code used to plot the images and respective captions can be found in the Python Notebook "HW2_c.ipynb".

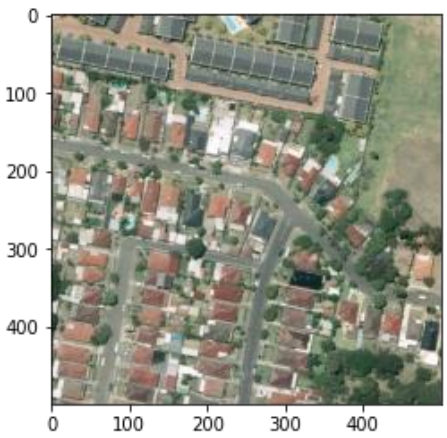
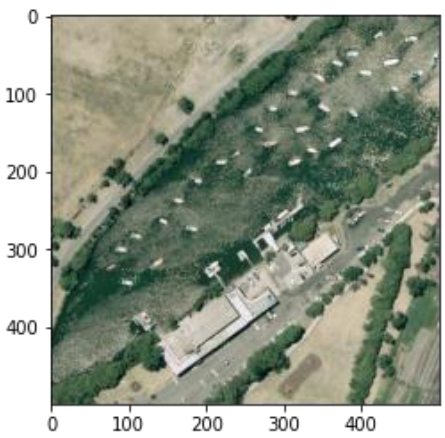
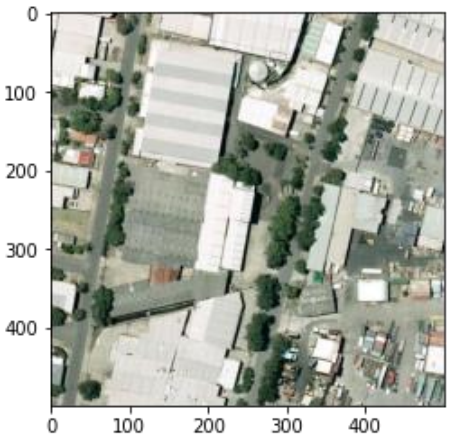
		
a residential area with houses arranged neatly and some roads go through this area	A curved river with some green waters and a highway passed by	an industrial area with many white buildings and some roads go through this area

Table 3

Conclusion

These classification problems represented great opportunities for us to have a more hands-on approach of the theoretical concepts and models presented in the classes, and allowed us to experiment and go further, understanding better the applications of Convolutional Neural Networks, which comprise a fundamental part of the theory behind unsupervised learning. Furthermore, these exercises generated a lot of discussion, because we were left in a situation where we could think by ourselves and come up with distinct suggestions and approaches.

Overall, these exercises were great learning experiences and provided knowledge that we will take for the rest of the trimester and eventually apply in our professional lives.