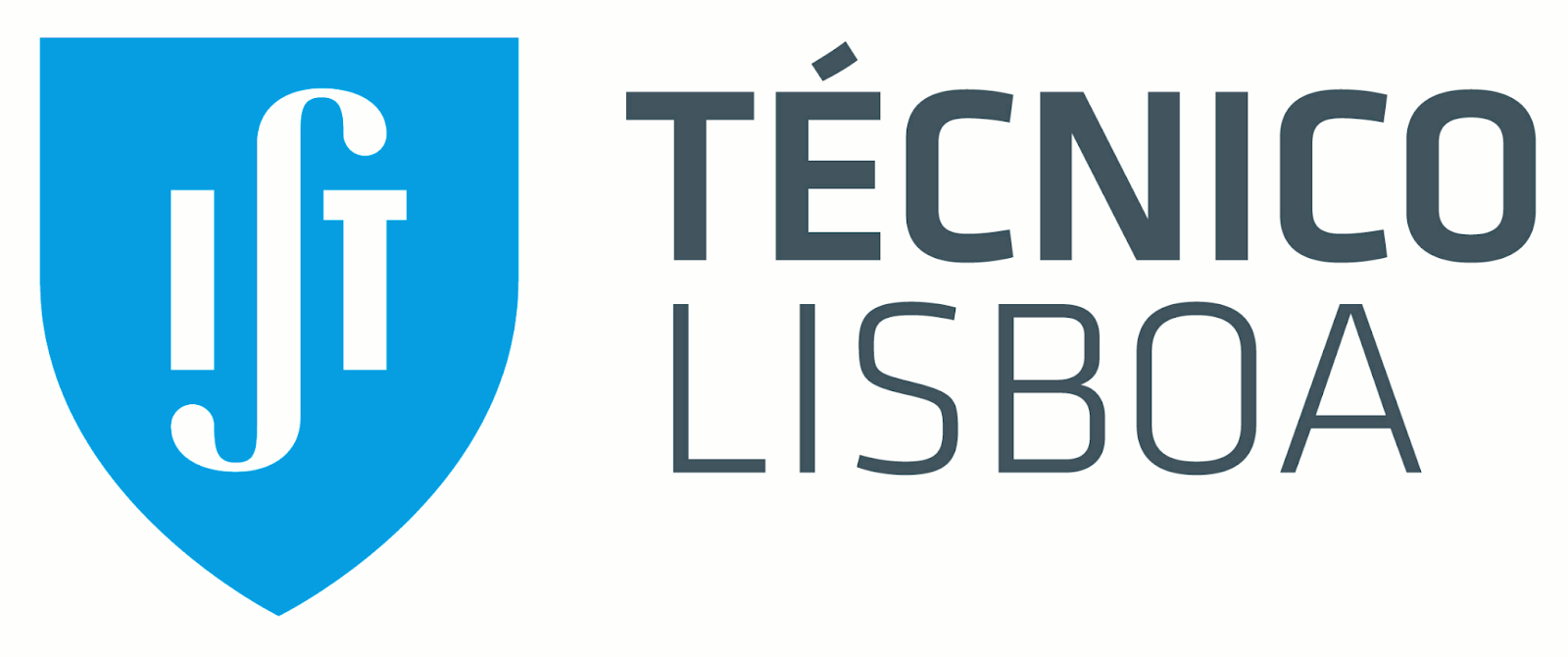
****

**Deep Learning**

2022/2023

**Homework 2 - Group 29**

Guilherme Pereira 105806  
 Miguel Belbute 96453

## Introduction

This project was developed for the course Deep Learning taught at Instituto Superior Técnico under the professors [Francisco Saraiva de Melo](https://fenix.tecnico.ulisboa.pt/homepage/ist40236) (theoretical lessons) and [Andreas Miroslaus Wichert](https://fenix.tecnico.ulisboa.pt/homepage/ist24622) (practical lessons).

## Contribution of each element

Guilherme Pereira’s contributions to the project were mainly under the code and development of the algorithms needed for the different models, with contributions to the writing of this report. Guilherme was also responsible for reviewing the data and demonstrations developed by his colleague.

Miguel Belbute’s contributions to the project encompass the writing of this report as well as answering the demonstrations and most of the theoretical questions. Miguel also contributed to writing and reviewing the algorithms produced by his colleague.

*In the spirit of the Deep Learning course, we prompted GPT-3 with “Can you write us a poem about deep learning?”. This was its response…*

*“Deep learning, a mystery to uncover,*

*A neural network, like a lover,*

*With layers deep and computations grand,*

*It's a force to be reckoned with, in a machine learning land.*

*Through data it trains, with a steady pace,*

*A network of neurons, in a hidden space,*

*It learns patterns and trends, with every pass,*

*With each iteration, it improves and surpasses.*

*From image recognition, to natural language,*

*Deep learning models, they've become quite the advantage,*

*A tool for the future, in this digital age,*

*A true technological marvel, on a data-driven page.*

*It's a journey of discovery, with no end in sight,*

*As we continue to push the boundaries, to new heights,*

*Deep learning, a puzzle to solve,*

*A never-ending quest for knowledge to evolve.”*

**Question 1**

1. **Convolutional Neural Network**
2. Considering the input is a set of images and knowing the first convolutional layer corresponds to a single filter with stride 1 and no padding, we can determine the dimensions of by utilising the following expression, taken from the theoretical classes[1]:

Since the stride is 1 for each dimension, this can also be simplified to:

That being said, we can easily obtain the dimensions:

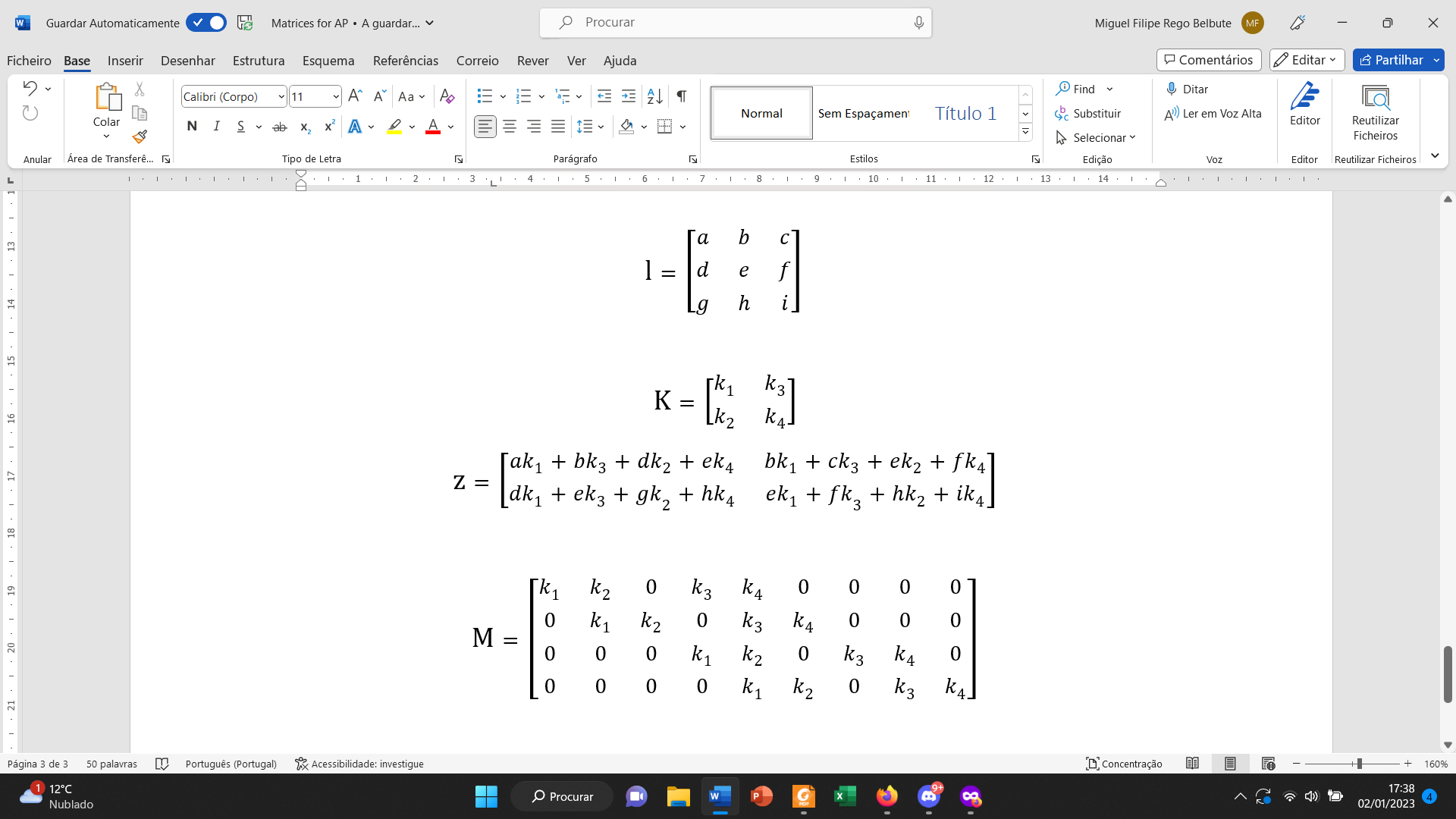
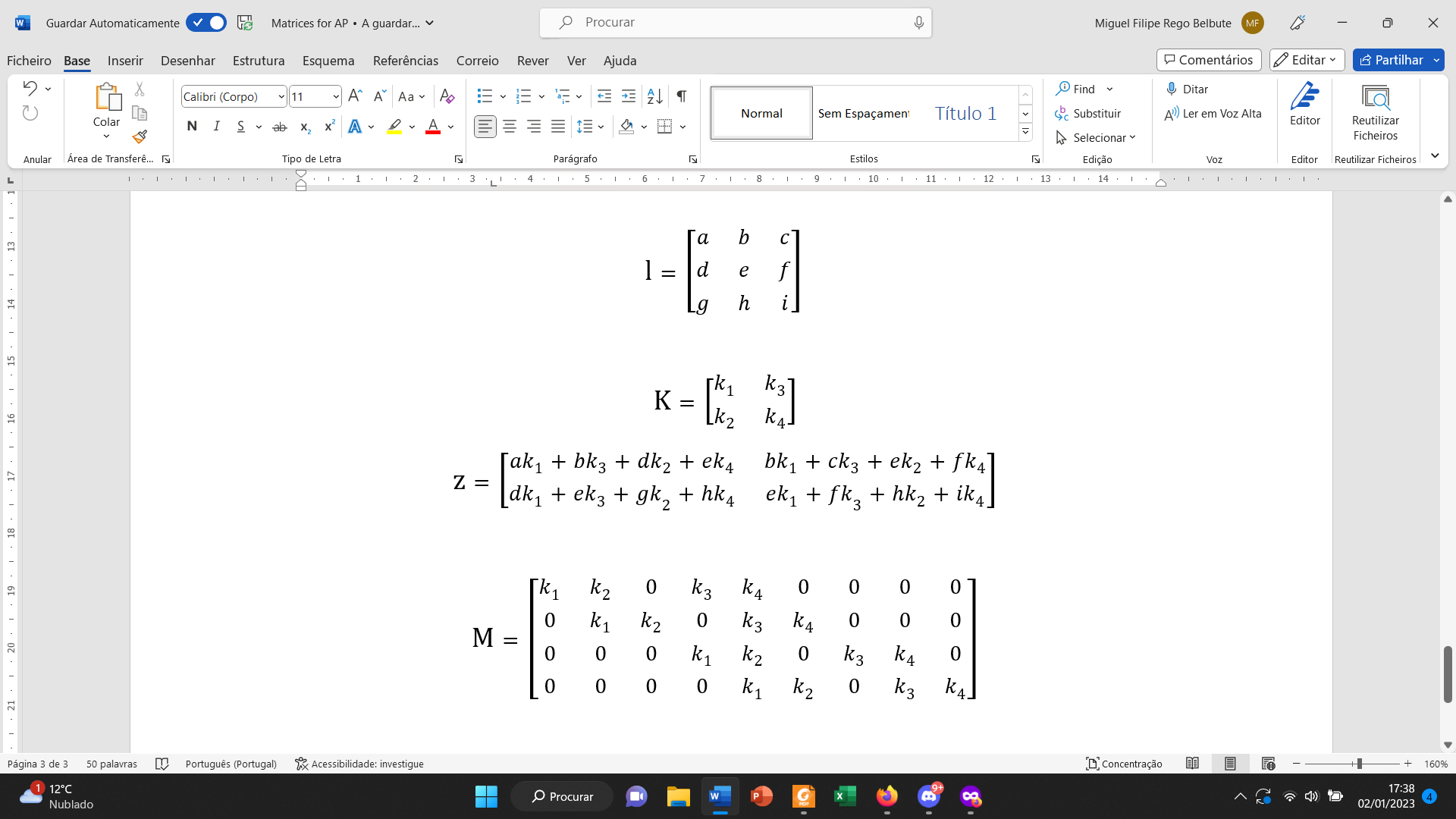
In other words, with and .

1. Once again considering the input and the result of the convolution , we can demonstrate there is indeed a matrix through which we can obtain from a linear transformation of , which are, respectively, the vectorized matrices of and .

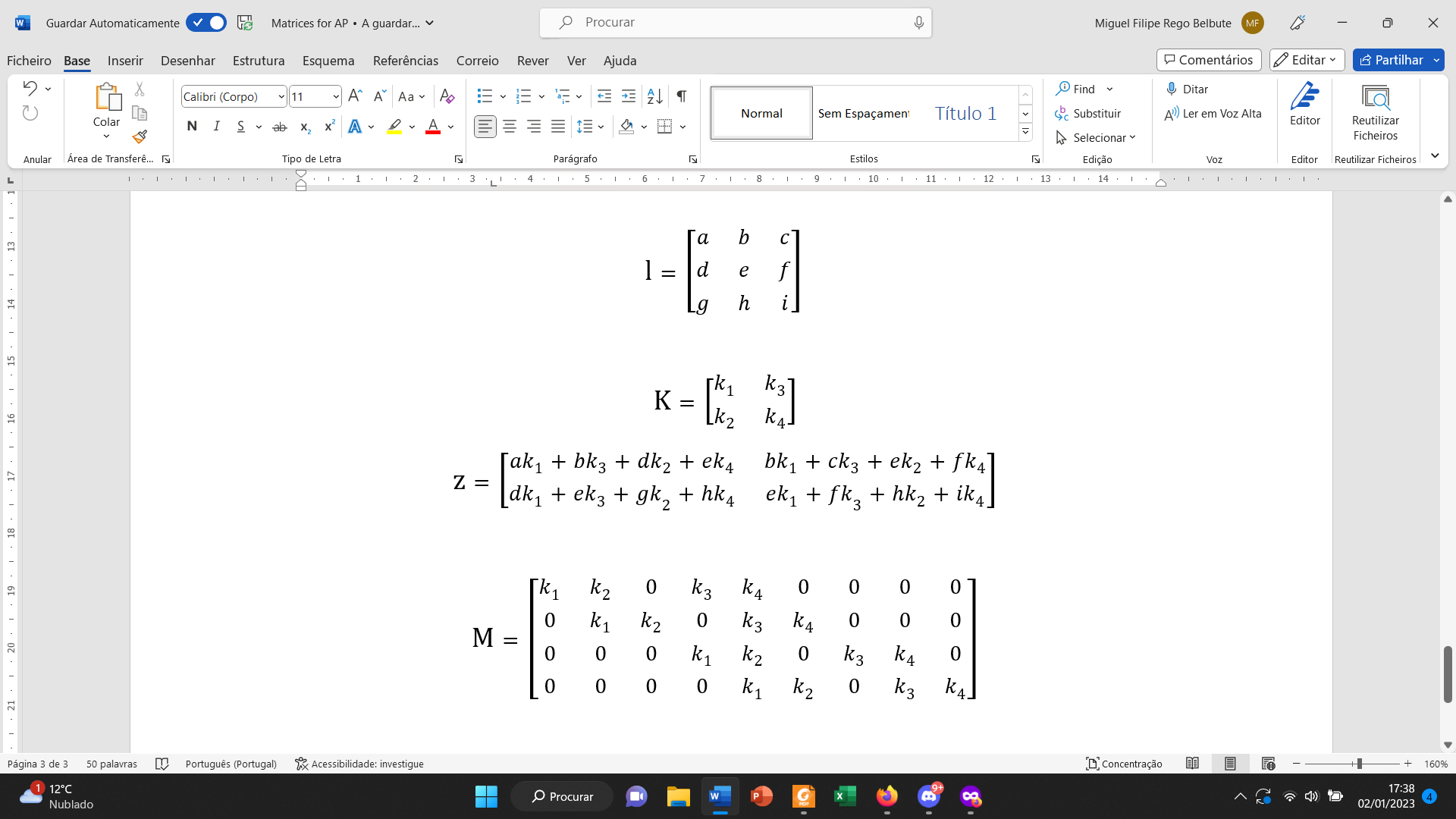
Firstly, let us infer the dimensions of this matrix :

which in turn is confirmed by the question itself.

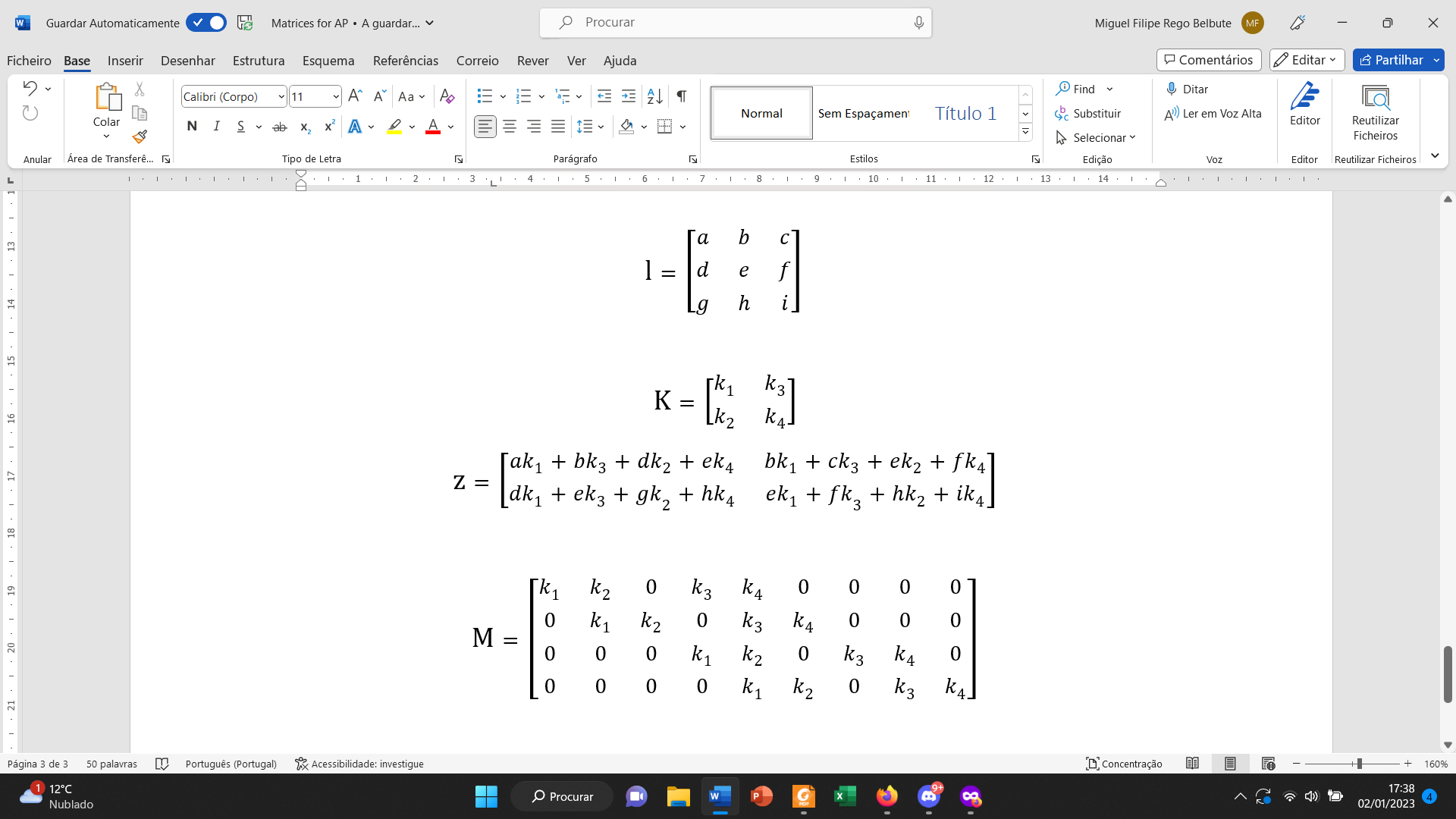
In order to find the general expression for element of , let us now examine a simple application of a convolutional filter, without loss of generalisation. With a input matrix and a kernel matrix ,

we would get the resulting , product of the convolution operation between and :



Vectorizing both and , thus obtaining and , we can now try to find a matrix that satisfies the equation (which represents a specific application of the present exercise). Comparing this equation with the one previously calculated, we can deduce the structure of matrix :



One should that is , which is in accordance with the inferred dimensions for the general case. Moreover, we can verify is a sparse matrix (most elements are zero) and is defined by a certain pattern that repeats along its rows, made up of the kernel’s elements.

Upon further research, it was found that this type of matrix is called a **circulant matrix** (due to its circular, repeating nature), which in turn is a particular kind of a Toeplitz matrix[2][3]. This specific matrix contains shifted versions of the kernel as its elements.

In true fashion, we can always define a circulant matrix that helps establish a convolution operation through matrix multiplication - this approach is a common topic of research related to the improvement of the overall performance of convolution operations[4]. One big advantage of this representation is that it makes it easier and more efficient to compute the backpropagation of the network.

Despite there being known computer algorithms to compute for any given dimensions (such as the one referenced in [5], where is the transpose of the doubly blocked matrix shown in step 5), we could not find a general expression for element (i, j) of .

1. The total number of parameters in the network () is given by the number of parameters in the convolutional layer () plus the number of parameters in the fully connected section (). One should note that the pooling layer does not have any learnable parameters.

The is easily calculated through the formula below

Disregarding the bias term, considering that our kernel only has two dimensions and since the network only has a single convolutional filter, we get:

In order to calculate we must first compute the dimensions of the output . Since , we can infer that has the same dimensions of - , with and .

After the max pooling layer, the dimensions of can be computed through the same expression used for :

where and the stride is also equal to 2,

By flattening , we get , which is a column vector.

Our fully connected section (FC) of the network must now linearly transform the into a column vector, since we have 3 final classes. The corresponding weights matrix () will therefore need to be a matrix, which means that

Finally, the **total number of parameters in the Convolutional Neural Network (CNN)** **is**

If we were to **replace the convolutional and max pooling layers by a fully connected layer, the number of parameters would drastically increase** (one of the reasons why convolutional networks can be preferred when compared to fully connected networks).

In order to implement a fully connected layer, the new weight matrix () would have to consider both the input ( - the flattened image) and the output () dimensions:

In other words, for us to have , the weight matrix would have the following dimensions:

Finally, the **total number of parameters in the Fully Connected Network (FCN) would be**

Let us now do a brief comparison between the number of parameters in each case:

Disregarding the common parameters associated with the FC layer present in both networks (in curly brackets above) and verifying that there is another common component for any given and input dimensions (in square brackets above), we can state that **the fully connected network would only have fewer parameters than the convolutional network if .**

Once again recalling that and are the input dimensions of an image, we can safely state that the FCN has more parameters than the CNN for virtually any practical and reasonable situations.

1. **Single-head Self-attention**

Considering the given information, let us first obtain the query, key and values vectors (, and , respectively) by projecting the embedding matrix to a lower dimension[1]:

One should note that , and are all column vectors. This is coherent with the fact that each element of corresponds to a single pixel - .

Let us now calculate the expression for the query-value affinity scores :

from where we can also compute the **attention probabilities** . Assuming a scaled dot product attention in order to contradict the peaking of the softmax function[1], is given by

where represents the length of the key vector (in this case, we have ).

Lastly, the **attention output** is given by:

**Question 2**

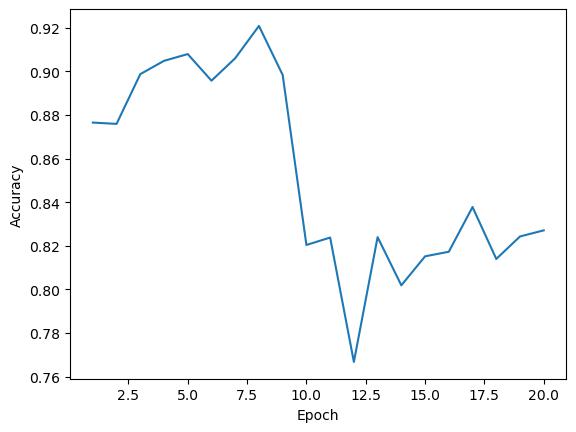
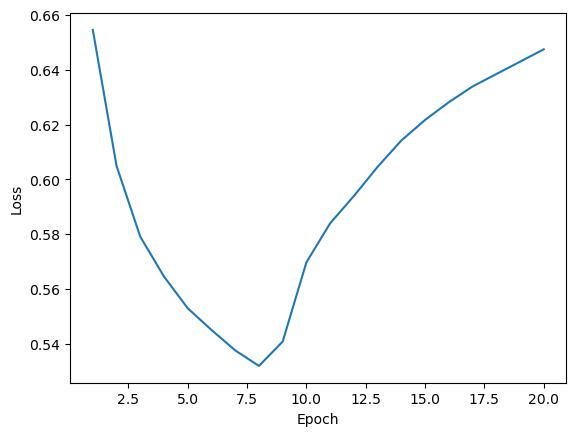
1. A CNN does in fact have fewer free parameters than a fully-connected network with equal input size and number of classes (like we demonstrated in Question 1). This is due to the fact that, in a CNN, the parameters are reused (tied/shared) as the kernel in a convolutional layer shifts across the input.
2. Despite the fewer parameters, CNNs do achieve a better generalisation on images and patterns that represent letters and numbers than a fully-connected network.

While a fully-connected network doesn’t acknowledge any order in its inputs (since they are laid out in a single vector), a CNN takes advantage of the local spatial coherence present in images and patterns. By using convolution in images, they are able to process an image in fewer operations while preserving the relationship each pixel/grid space has with their neighbours (local connectivity).

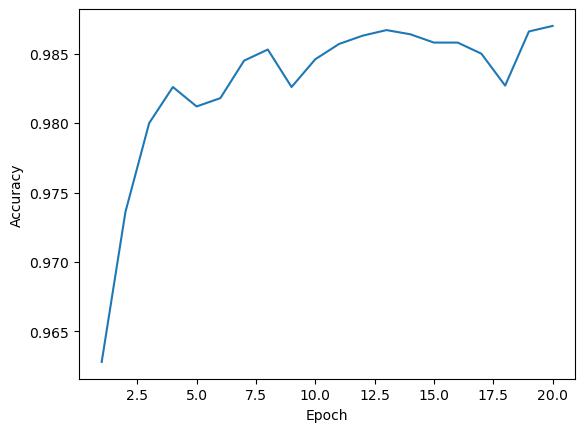
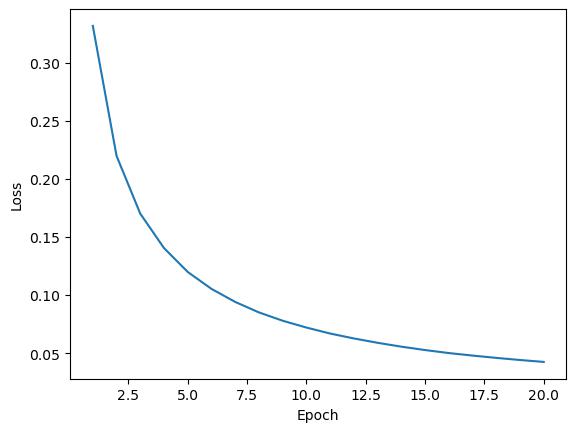
Another aspect of CNNs that also gives them an advantage over fully-connected networks is the presence of pooling layers. These layers are responsible for effectively downscaling the given input images or patterns while preserving relevant information in each pixel/grid space, further simplifying the input for further processing. This step is impossible in a fully-connected layer since the vector that holds the input cannot be downscaled without losing information as coherence isn’t preserved as already discussed.

1. In the case of a one dimensional CNN, that network is still expected to achieve better generalisation than a fully-connected network.

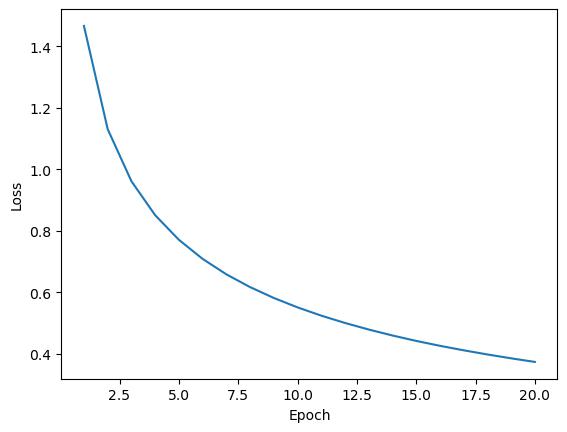
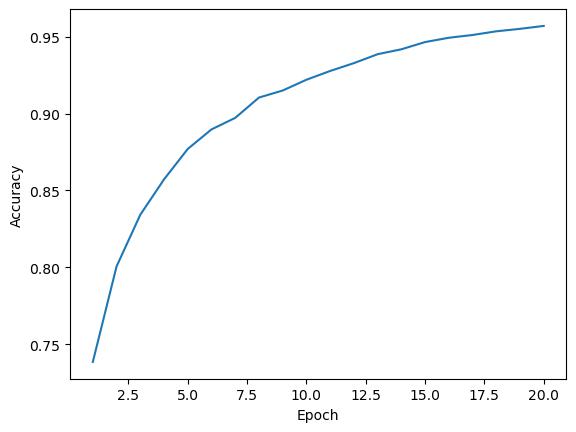
Although the preservation of local connectivity between the independent sensors is redundant (due to the inexistence of a specific spatial structure), a CNN can still identify patterns in the sequence of sensors given as inputs. By finding and learning a pattern, the CNN can apply the same transformation to that pattern immediately upon its recognition again in a different position (invariance).



*Graph Set 1 - From left to right, all corresponding to a LR 0.01: validation accuracy, training loss*

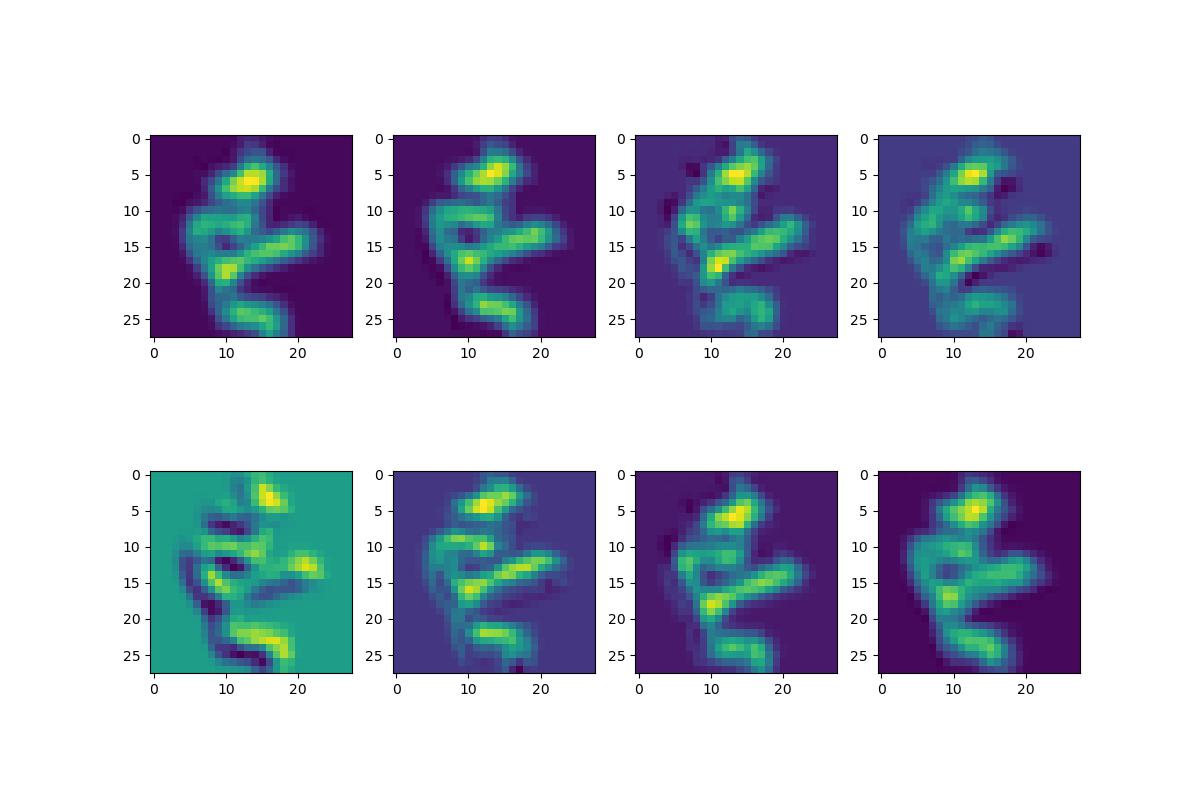
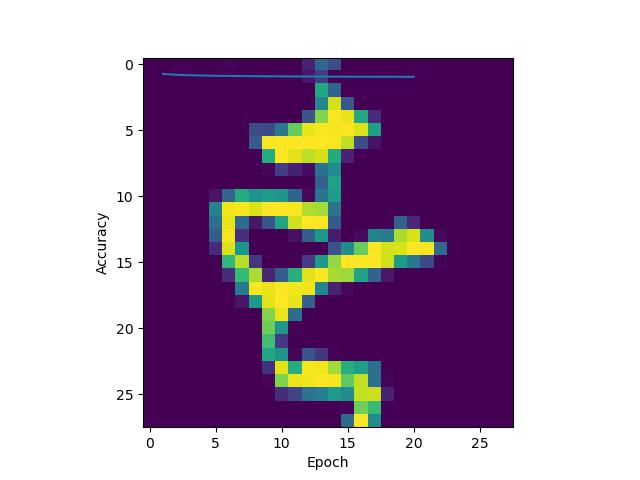


*Graph Set 2 - From left to right, all corresponding to a LR 0.0005: validation accuracy, training loss*



*Graph Set 3 - From left to right, all corresponding to a LR 0.00001: validation accuracy, training loss*

The learning rate with the best configuration was 0.0005, with a final validation accuracy higher than 0.985 and a training loss lower than 0.05.

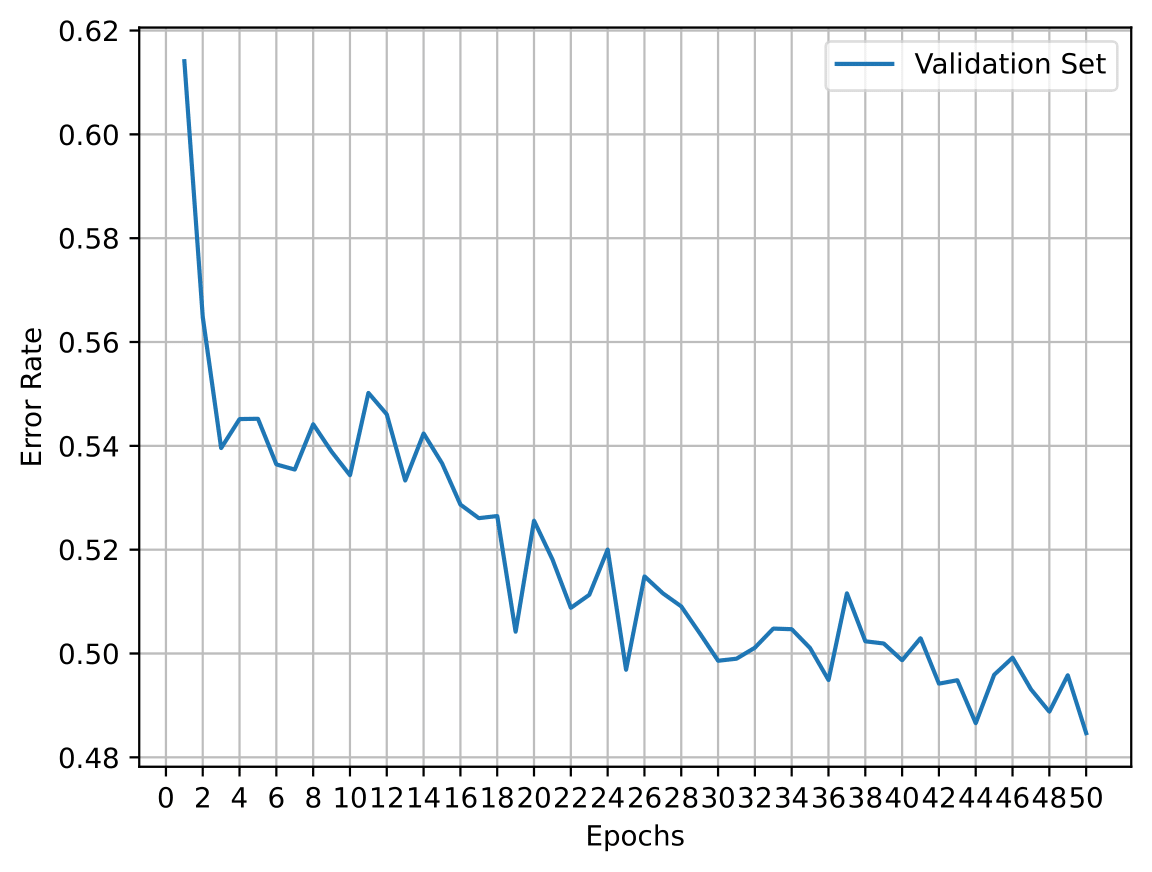
1. 

*Image Set 1 - From left to right: original image, activation maps*

From the comparison of the activation maps with the original image, what appears to be highlighted in the activation maps are the silhouettes/features of the character present in the original image with an emphasis on the yellow regions.

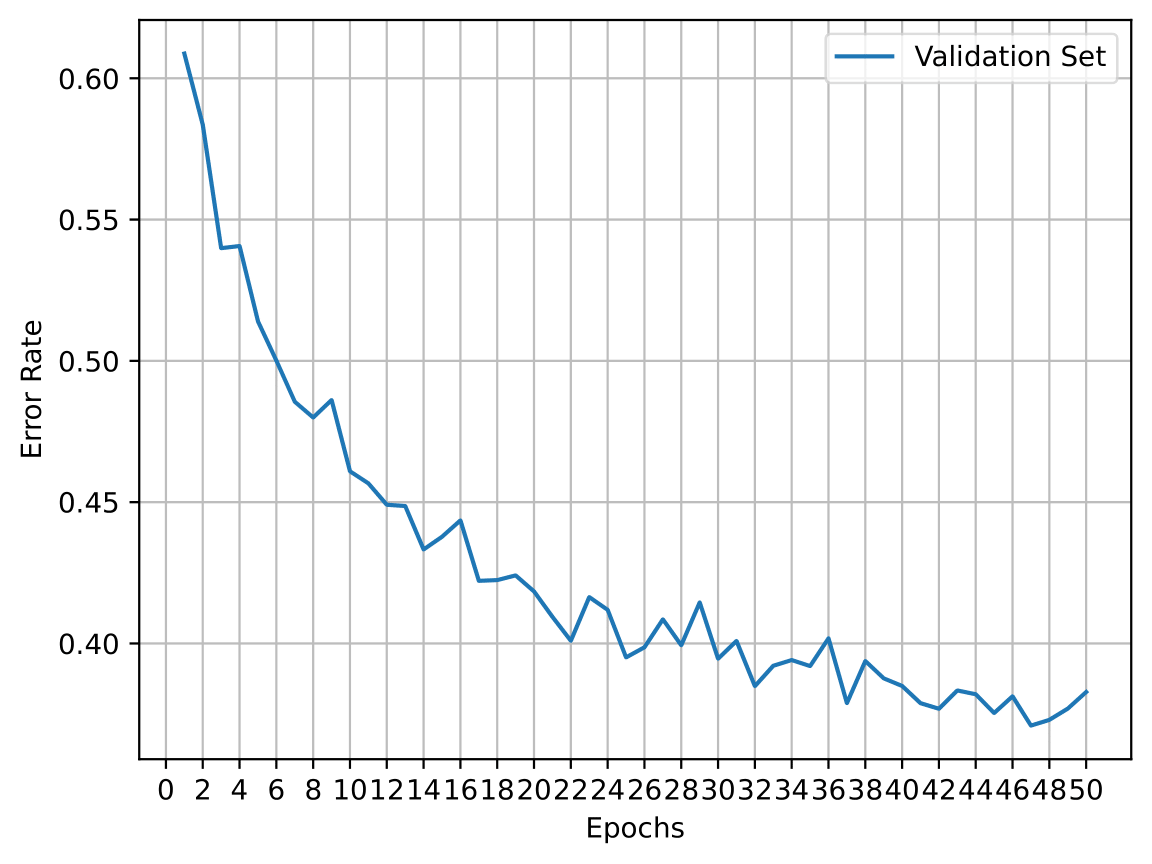
**Question 3**

1. **Character-level Machine Translation**



*Graph Set 4 - Validation error rate over 50 epochs of Encoder-Decoder model with Bidirectional LSTM and Auto-regressive LSTM*

The final validation error rate is 0.4847 and the final test error rate is 0.4906.



*Graph Set 5 - Validation error rate over 50 epochs of Encoder-Decoder model with Bidirectional LSTM and Auto-regressive LSTM*

The final validation error rate is 0.3828 and the final test error rate is 0.3862.

1. There are multiple possible ways of further improving the model’s performance without altering its encoder-decoder architecture, most of which are popular current topics of research. However, before mentioning these improvements, let us reflect on the optimizations it already has.

The encoder-decoder model consisting of Recurrent Neural Networks (RNN) suffers from shortcomings like the vanishing gradient, mainly due to its chronological architecture - information belonging to the beginning of the sequence is often “lost” during training if the network is big enough and does not have mechanisms to handle long-term dependencies.

The development of techniques like the **Long Short-Term Memory (LSTM)** help minimise this issue by implementing gates (extra weights) that selectively forget or retain information from previous time steps. The **Attention** mechanism was also introduced as a means to solve this long-distance dependence problem by reviewing and selecting the most important information in the whole input sequence. Lastly, by simultaneously capturing the “past” and “future” context of any given token, **Bidirectional** encoders also boost the results of machine translation algorithms.[8][9][10]

In the current implementation of our model, the decoding process comes to a halt when the sequence reaches a maximum length of 50 characters or when the token marking the end of the sequence is reached. One way of improving the model’s performance would be to **increase the maximum possible length of the generated sequence** (increasing the for-loop duration), thus giving more time to generate a correct translation. In other words, this would increase the likelihood of the model reaching the final token with a complete and acceptable translation.

On the other hand, we could **apply a Beam Search technique** instead of utilising a single best candidate for each position of the generated sequence (greedy search), multiple tokens could be considered based on conditional probability. This also means that rather than having just the highest-probability sequence at each step, the model would store a list of multiple highest-probability sequences. Although this could take a toll performance wise, the quality of the results would undoubtedly increase because multiple translation alternatives would be explored.[11]

An **Early Stop** mechanism could also be considered during training in order to prevent overfitting and lack of generalisation capabilities. This technique would stop the training process once the accuracy of the model on a validation started decreasing.[12]

Finally, perhaps the most general improvement for a machine translation algorithm - quite simply **expanding the size and the diversity of the dataset** would develop the generalisation of the model thus yielding better results.

**References**

*[1] - Martins, A.F.T. et al. (no date) Theoretical Lectures, Deep Learning (DEI). Instituto Superior Técnico (IST). Available at: https://fenix.tecnico.ulisboa.pt/disciplinas/AP-Dei/2022-2023/1-semestre (Accessed: December 21, 2022).*

*[2] - Circulant matrix (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Circulant\_matrix (Accessed: January 4, 2023).*

*[3] - Convolution (2023) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Convolution (Accessed: January 4, 2023).*

*[4] - Wang, Y. et al. (2018) “Towards ultra-high performance and energy efficiency of deep learning systems: An algorithm-hardware co-optimization framework,” Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). Available at: https://doi.org/10.1609/aaai.v32i1.11653.*

*[5] - Alisaaalehi (no date) Convolution As Multiplication: Step by Step Explanation of 2D Convolution Implemented as Matrix Multiplication Using Toeplitz Matrices, GitHub. Available at: https://github.com/alisaaalehi/convolution\_as\_multiplication (Accessed: January 5, 2023).*

*[6] - Editorial Team (2020) Beginners Guide to Convolutional Neural Network from Scratch Kuzushiji-MNIST at: https://towardsai.net/p/machine-learning/beginner-guides-to-convolutional-neural-network-from-scratch-kuzushiji-mnist-75f42c175b21 (Accessed: December 30, 2022).*

*[7] - Simeon Kostadinov (2019) Understanding Encoder-Decoder Sequence to Sequence Model at: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346 (Accessed: January 02, 2023).*

*[8] - Long Short-Term Memory (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Long\_short-term\_memory (Accessed: January 14, 2023).*

*[9] - Sharma, A. (2019) Attention-Based Neural Machine Translation, Medium. Towards Data Science. Available at: https://towardsdatascience.com/attention-based-neural-machine-translation-b5d129742e2c (Accessed: January 14, 2023).*

*[10] - Zhang, A. et al. (no date) 10.4. Bidirectional Recurrent Neural Networks, Dive into Deep Learning. Available at: https://d2l.ai/chapter\_recurrent-modern/bi-rnn.html (Accessed: January 14, 2023).*

*[11] - What is Beam Search? Explaining the Beam Search Algorithm (no date) Width.ai. Available at: https://www.width.ai/post/what-is-beam-search (Accessed: January 14, 2023).*

*[12] - Early stopping (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Early\_stopping (Accessed: January 14, 2023).*

*[13] - Pytorch attention documentation (2018) Pytorch. Michael Petrochuk Revision d7814a29. Available at: https://pytorchnlp.readthedocs.io/en/latest/\_modules/torchnlp/nn/attention.html (Accessed: January 15, 2023).*