Deep Learning (IST, 2022-23)

Homework 2

Group 3

Martim Santos 95638, Marina Gomes 95637

Identical to the first homework, this project was divided equally amongst the members of the group so each member contributed the same. We decided to solve the homework together, combining our ideas in the various exercises. Each time one of the members had an idea for any question, that idea was discussed and materialized if it made sense. As a result of that, both members have total knowledge about what was done and are able to explain every detail since both participated in every task.

Question 1

Question 1.1

Question 1.1a

Given an $N \times N \times D$ image, $F \times F \times D$ filters, K channels, and stride S, we know that the resulting output will be of size $M \times M \times K$, where

$$M = (N - F)/S + 1$$

We have $x \in \mathbb{R}^{H \times W}$ and a convolutional layer with a single $M \times N$ filter, a stride of 1 and no padding, therefore the output will have the following dimensions:

$$\begin{aligned} H' &= (H - M)/1 + 1 & W' &= (W - N)/1 + 1 \\ &= H - M + 1 & &= W - N + 1 \end{aligned}$$

We then have $\boldsymbol{z} = \text{conv}(W, x) \in \mathbb{R}^{(H-M+1)\times(W-N+1)}$.

Question 1.1b

We know that $z = \text{conv}(W, x)$ which means that for each z_{ij} there is an expression that relates an entry of x and an entry of W. Having

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ w_{M1} & \dots & \dots & w_{MN} \end{bmatrix} \in \mathbb{R}^{M \times N}, \qquad x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1W} \\ x_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{H1} & \dots & \dots & x_{HW} \end{bmatrix} \in \mathbb{R}^{H \times W}$$

We can deduce that

$$z_{ij} = \sum_{m=1}^{M} \sum_{n=1}^{N} x_{i+m-1, j+n-1} \cdot w_{m,n}$$

We also have

$$x' = vec(x) = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{H1} \\ x_{12} \\ \vdots \\ x_{HW} \end{bmatrix} \in \mathbb{R}^{HW \times 1}, \qquad z' = vec(z) = \begin{bmatrix} z_{11} \\ z_{21} \\ \vdots \\ z_{H'1} \\ z_{12} \\ \vdots \\ x_{H'W'} \end{bmatrix} \in \mathbb{R}^{H'W' \times 1} \quad \text{since } z \in \mathbb{R}^{H' \times W'}$$

Then there in a matrix M such that $z' = Mx'$. That matrix $M \in \mathbb{R}^{H'W' \times HW}$ in order to respect the dimensions of z' and x'.

Let

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & \dots & M_{1,HW} \\ M_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ M_{H'M',1} & \dots & \dots & M_{H'M,HW} \end{bmatrix}$$

Matrix M will have weights in the positions that multiply with $x_{i,j}$ if that $x_{i,j}$ is a specific window.

Therefore, we can deduce that every entry of M can be written according to each $z_{m,n}$ in z' and $x_{i,j}$ in x'. We then have

$$z_{m,n} = \begin{cases} M_{(n-1)H'+m,(j-1)H+i} = w_{i-m+1,j-n+1} & \text{if } x_{i,j} \text{ is in the window, that is } m \leq i < m+M \wedge n \leq j < n+N \\ 0 & \text{Otherwise} \end{cases}$$

Question 1.1c

Given that we have a convolutional layer with a single $M \times N$ filter, we start by having $1 \times M \times N$ parameters.

A ReLU activation function is then applied to the result of the convolution. This function does not have any parameters.

The result $h_1 \in \mathbb{R}^{H' \times W'}$ then goes through a 2×2 max pooling layer with a stride of 2 and no padding which also does not have any parameters. This is followed by a Flatten, obtaining $h_2 \in \mathbb{R}^{1 \times (\frac{H'}{2} + \frac{W'}{2})}$.

The output layer, a fully connected layer, will then have a weight matrix with $3 \times (\frac{H'}{2} + \frac{W'}{2})$ parameters since it performs a linear operation to transform their input features into output features.

Lastly, a softmax activation is applied to obtain y without any aditional parameters. We then conclude that the number of parameters in the network is

$$M \times N + 3 \times \left(\frac{H'}{2} + \frac{W'}{2} \right) = M \times N + 3 \times \left(\frac{(H - M + 1)}{2} + \frac{(W - N + 1)}{2} \right)$$

If the convolutional and max pooling layers are replaced by a fully connected layer yielding an output with the same dimension as h_2, we would not have the filter parameters but instead a new weight matrix with $(\frac{H'}{2} + \frac{W'}{2}) \times (H \times W)$ parameters. The number of parameters of this new network would be

$$\left(\frac{H'}{2} + \frac{W'}{2} \right) \times (H \times W) + 3 \times \left(\frac{H'}{2} + \frac{W'}{2} \right)$$

It is important to note that the values of M and N must be less than or equal to the values of H and W, respectively, as the size of the kernel cannot be larger than the dimensions of the image. In practice, kernel sizes in the range of 3×3 to 5×5 are commonly used.

Based on this analysis, it can be inferred that the network containing two fully connected layers will have a significantly larger number of parameters compared to the original network.

Question 1.2

Given that $x' = \text{vec}(x) \in \mathbb{R}^{M\times N}$ and $W_Q = W_K = W_V = 1$ we can state that the Query, Keys and Values vectors will be, respectively:

$$Q = x' \cdot W_Q = x' \qquad K = x' \cdot W_K = x' \qquad V = x' \cdot W_V = x'$$

Having this vectors we can know calculate the attention probabilities P and the attention output Z in function of x', respectively:

$$P = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right) = softmax\left(\frac{x' \cdot x'^T}{\sqrt{1}}\right) = softmax\left(x' \cdot x'^T\right)$$

$$Z = P \cdot V = softmax\left(x' \cdot x'^T\right) \cdot x'$$

Question 2

Question 2.1

In a fully-connected network, each neuron in a layer is connected to each neuron in the previous layer, which means that the number of parameters will correspond to the product of the number of neurons in both layers.

On the other hand, convolutional neural networks (CNNs) take advantage of the spatial structure of the input data by sharing parameters (weigths and biases) in the convolutional layers. This reduces the number of free parameters that need to be learned.

Question 2.2

CNNs use convolutional layers, which apply a set of shared parameters to small parts of the input to extract features and pass them to the next layer, where more complex features are extracted. This allows CNNs to learn features that are invariant to shifts, rotations and scaling which is usefull for image and pattern classification. Also, by having a smaller number of parameters that need to be learned, CNNs are more able to generalize to new data with a reduced overfitting risk.

Additionally, CNNs usually alternate convolutional layers with pooling layers. These pooling layers are used to subsample the input data by picking the maximum value of each small region. This reduces the spatial dimensions of the data and makes the model more robust.

Question 2.3

The advantage of using CNNs is their ability to use the spatial structure of the input data to learn features through mechanisms like sharing weights and subsampling.

If the input has no spatial structure, the CNN will not be able to use those mechanisms in a useful way and it's convolution layers may not be able to learn meaningful parameters. In this case, a fully connected network may be more appropriate, as it can learn any

arbitrary function that maps the input to the output without any assumptions of the input data structure.

Question 2.4

After implementing the Convolutional Neural Network with the asked structure, we obtained the following results after training 20 epochs of the model using Adam tuning and a batch size of 8.

Learning Rate	0.00001	**0.0005**	0.01
Training Loss	0.3731	0.0426	0.4849
Validation accuracy	0.9564	**0.9869**	0.9067

Table 1: Obtained results for the different learning rates

The best configuration is when the learning rate is 0.0005. For this configuration, the model showed a final validation accuracy of 0.9869 and a final test accuracy of 0.9559. Below are plots that show the training loss and the validation accuracy, both as a function of the epoch number for the best configuration.

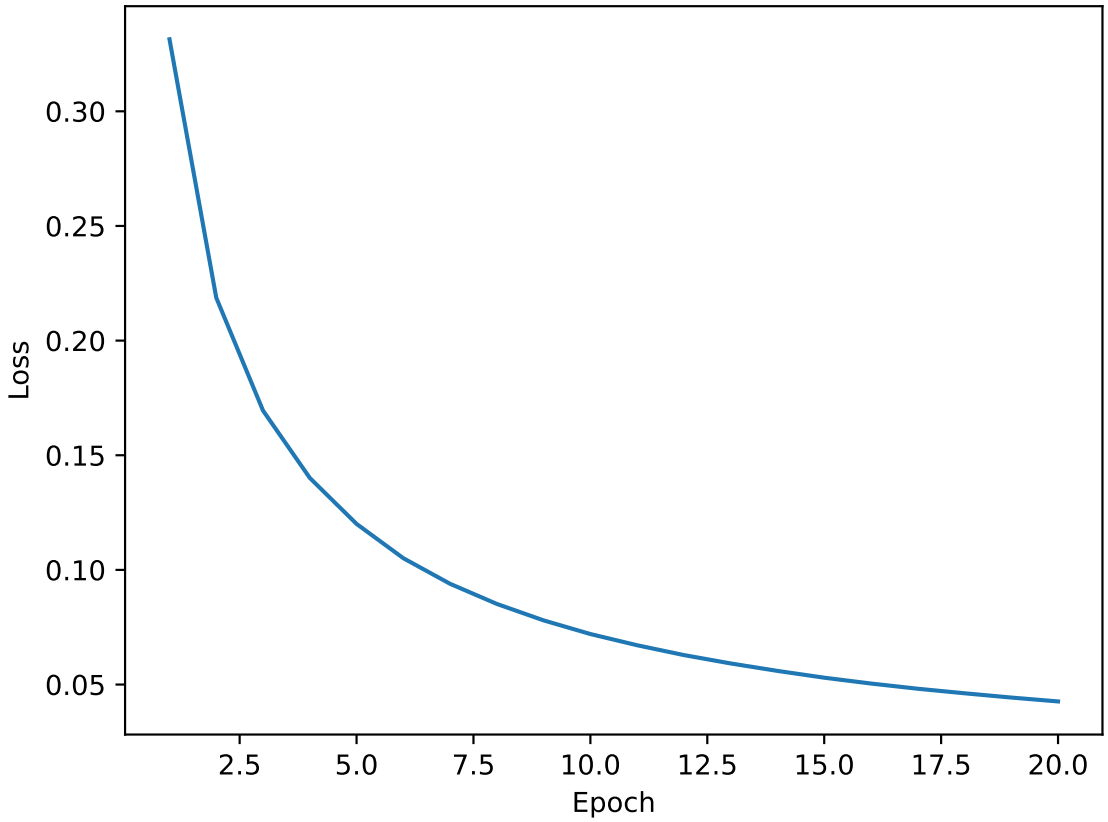

Figure 1: CNN – Training loss as a function of the Epoch number.

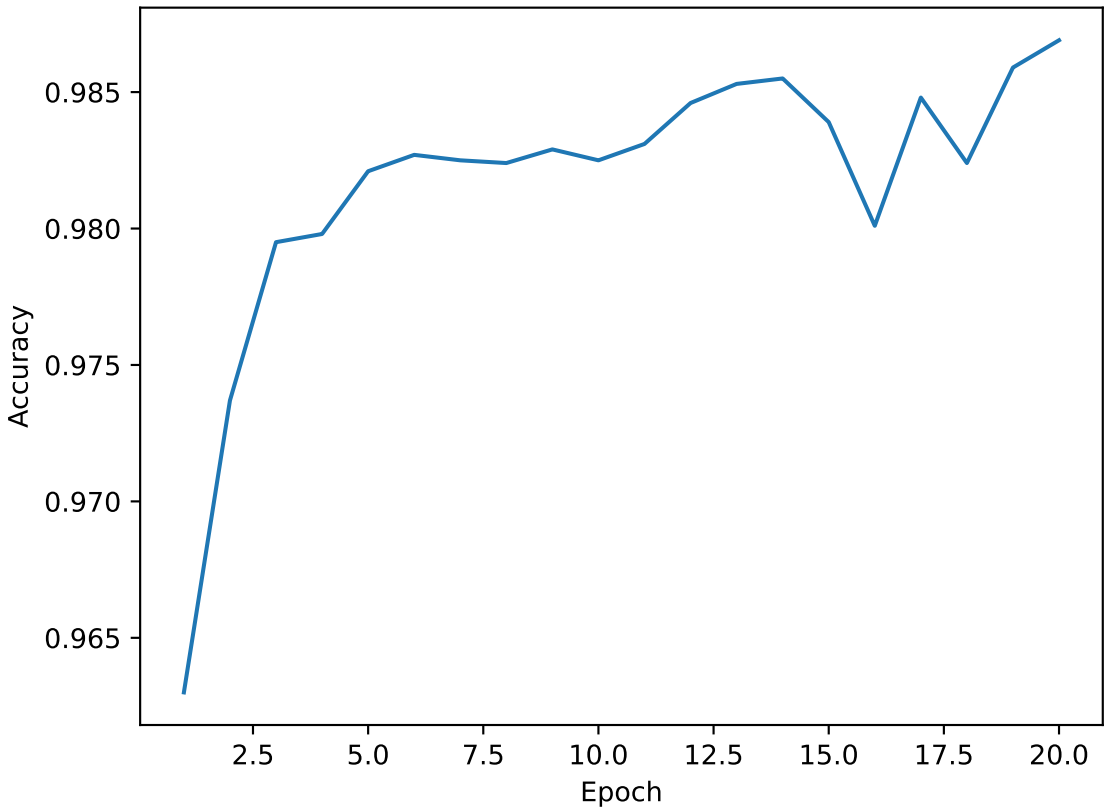

Figure 2: CNN – Validation accuracy as a function of the Epoch number.

Question 2.5

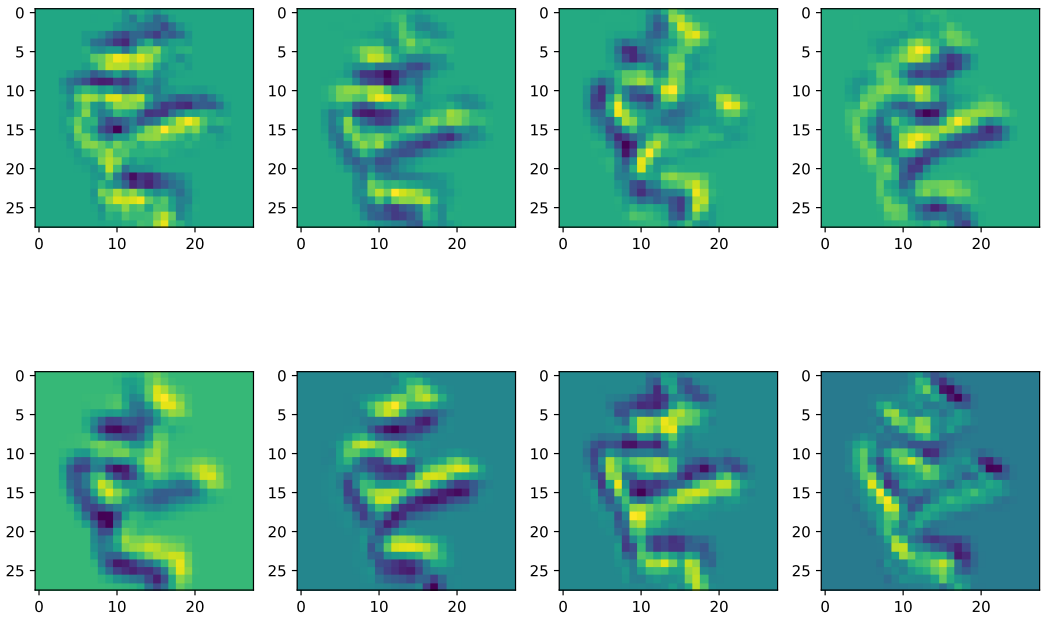

Figure 3: CNN – Activation maps of the first convolutional layer.

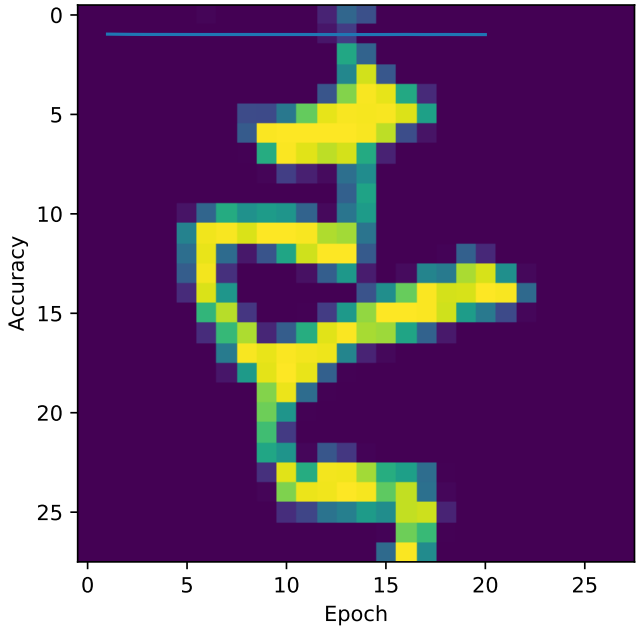

Figure 4: CNN − Original training example.

The activation maps of the first convolutional layers highlight features such as edges, textures and patterns of the input image at various scales and orientations. From the comparison of the activation maps and the original image we can see that each activation has pixels highlighted, specifically those that are most similar to the features being detected by the filter being used.

Question 3

Question 3.1

Question 3.1a

After implementing and trainig the character-level machine translation model with the asked structure, asked parameters and without an attention mechanism, we obtained the following results.

Final Validation error rate	0.4892
Final Test error rate	0.4898

Table 2: Final error rate without attention

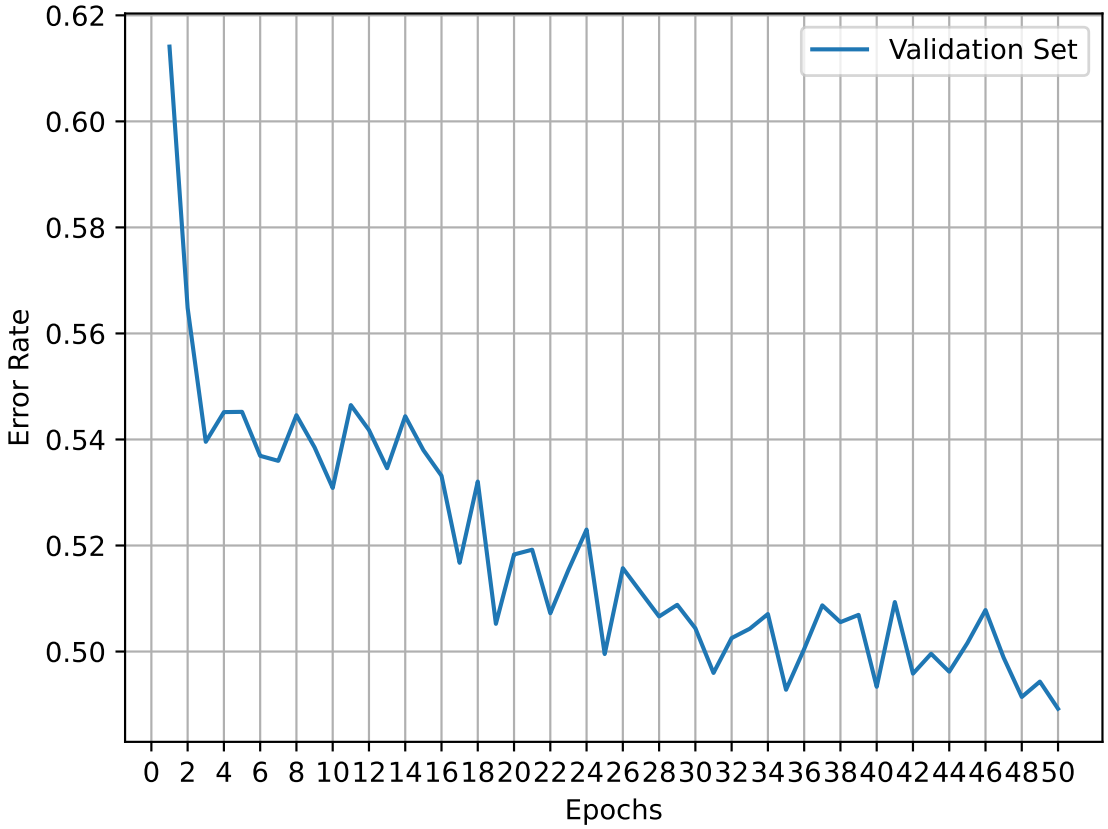

Figure 5: CMT – Validation error rate over epochs without attention.

Question 3.1b

By adding an attention mechanism to the decoder of the previous model (bilinear attention), we obtained the following results.

Final Validation error rate	0.3649
Final Test error rate	0.3748

Table 3: Final error rate with attention.

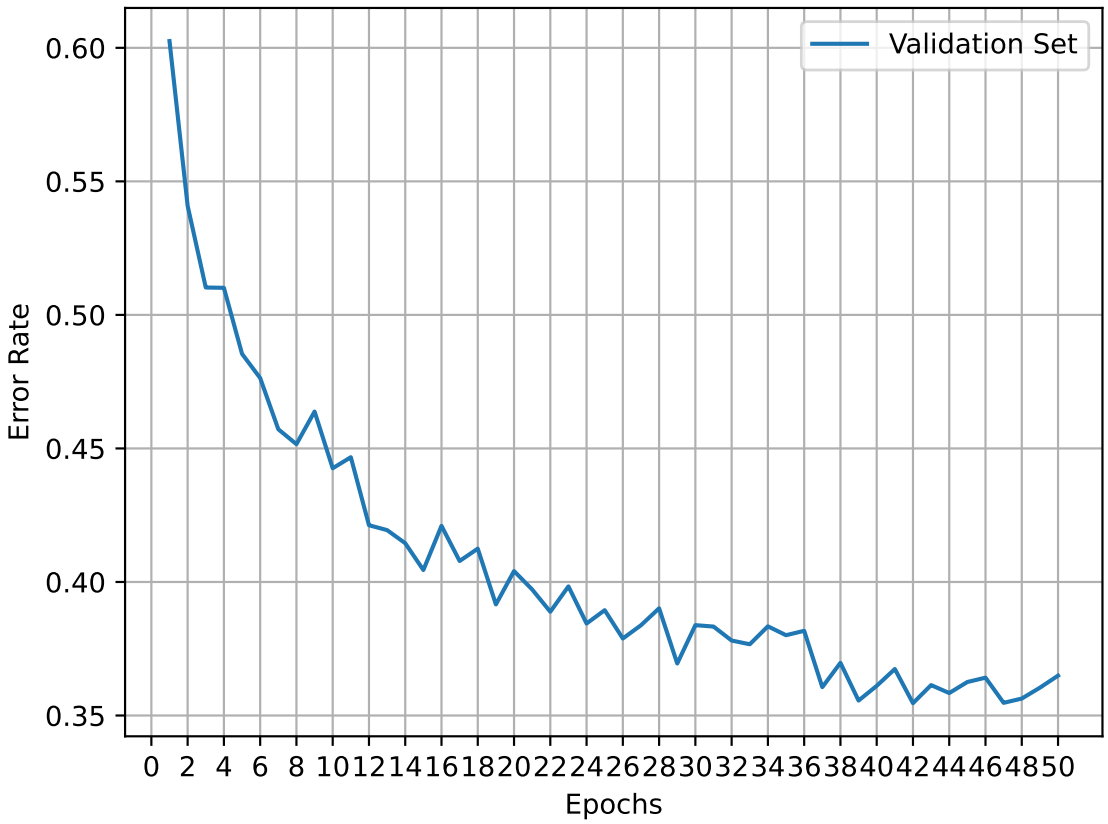

Figure 6: CMT − Validation error rate over epochs with attention.

Question 3.1c

One way of improving the results without changing the architecture of the model is by implementing beam search in the decoding process in `test()` instead of the current greedy search. Beam search is a heuristic search algorithm that explores a set of the most probable options for a node. In contrast to the greedy search, which always chooses the locally optimal solution, beam search evaluates a broader selection of options and can find a globally optimal solution and therefore lead to translations that are more precise and accurate.