TÉCNICO LISBOA

Homework 2

Deep Learning

Group 5

Duarte Calado de Almeida 95565

André Lopes Rodrigues 96576

The contribution of each member was as follows: André Rodrigues did Questions 1.1 (a), 1.1 (c), 2.1, 2.2, 2.4 and 3.1, while Duarte Almeida did Questions 1.1 (b), 1.2, 2.3, 2.5, 3.2 and 3.3. The elaboration of this report was made in colaboration by both students.

Question 1

1. (a) For a convolutional layer with a single $M \times N$ filter with no padding and a stride of s, the element (i, j) of the output $\boldsymbol{z}$ can be written as:

$$z(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} w(m,n) x(row(i) - 1 + m, col(j) - 1 + n)$$

where row and col are sequences of row and column indices, respectively, such that $row(i) = 1 + s(i-1)$ and $col(j) = 1 + s(j-1)$. Given that $1 \leq m \leq M$, $1 \leq n \leq N$ and that the maximum row and column indices of $\boldsymbol{x}$ are respectively H and W, we have that the maximum row index of $\boldsymbol{z}$ is the largest integer i that satisfies:

$$row(i) - 1 + M \leq H \Leftrightarrow 1 + s(i-1) - 1 + M \leq H \Leftrightarrow s(i-1) \leq H - M \Leftrightarrow i \leq \frac{H-M}{s} + 1$$

that is, the number of rows in z is $\lfloor \frac{H-M}{s} \rfloor + 1$. Analogously, the maximum column index in $\boldsymbol{z}$ is the largest integer j that satisfies $col(j) - 1 + N \leq W$, which is $\lfloor \frac{H-M}{s} \rfloor + 1$. **Thus, $\boldsymbol{z}$ is a $(H - M + 1) \times (W - N + 1)$ matrix**.

(b) Without loss of generality, consider the vectorization operation where, for a matrix $\boldsymbol{x} \in \mathbb{R}^{X \times Y}$, $\boldsymbol{x}(i,j) = \boldsymbol{x'}_{X(i-1)+j}$, i.e., one that is made row-wise.
To prove that there exists a matrix $\boldsymbol{M} \in \mathbb{R}^{HW \times H'W'}$ such that $\boldsymbol{z'} = \boldsymbol{M}\boldsymbol{x'}$, we first note that $\boldsymbol{x'} \in \mathbb{R}^{HW}$ and $\boldsymbol{z'} \in \mathbb{R}^{H'W'}$, following the proof above. It then suffices to prove that, for every $i \in \{1, \ldots, H'W'\}$, $z'_i = \boldsymbol{m}^T \boldsymbol{x'}$, i.e., z'_i is a linear combination of the entries of $\boldsymbol{x'}$.
For every $i \in \{1, \ldots, H'W'\}$, define both $\alpha_i = \lfloor \frac{i-1}{H'} \rfloor + 1$ and $\beta_i = ((i-1) \mod W' + 1)$ that make $z'_i = z(\alpha_i, \beta_i)$. We then have that:

$$\begin{aligned} z'_i = z(\alpha_i, \beta_i) &= \sum_{m=1}^{M} \sum_{n=1}^{N} w(m,n) x(\alpha_i - 1 + m, \beta_i - 1 + n) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} w(m,n) x'_{H(\alpha_i - 2 + m) + (\beta_i - 1 + n)} \end{aligned}$$

thus proving the desired result.
To specify each entry $M(i,j)$ of $\boldsymbol{M}$, note that $\boldsymbol{z'} = \boldsymbol{M}\boldsymbol{x'}$ implies that $\nabla_{\boldsymbol{x'}} \boldsymbol{z'} = \boldsymbol{M}^T$ and, hence, $\frac{\partial z'_i}{\partial x'_j} = M^T(j,i) = M(i,j)$. Let α, β, γ and δ be numbers that make $z'_i = z(\alpha, \beta)$ and $x'_j = x(\gamma, \delta)$ and which are defined in the same fashion as before. We then have that

$$M(i,j) = \frac{\partial z'_i}{\partial x'_j} = \frac{\partial z(\alpha, \beta)}{\partial x(\gamma, \delta)} = \frac{\partial}{\partial x(\gamma, \delta)} \left(\sum_{m=1}^{M} \sum_{n=1}^{N} w(m,n) x(\alpha - 1 + m, \beta - 1 + n) \right)$$

For $x(\gamma, \delta)$ to appear sum above, there must exist some natural numbers m^* and n^* that cumulatively satisfy:

$$\begin{cases} \gamma = \alpha - 1 + m^* \wedge 1 \leq m^* \leq M \\ \delta = \beta - 1 + n^* \wedge 1 \leq n^* \leq N \end{cases} \Rightarrow \begin{cases} m^* = \gamma - \alpha + 1 \wedge 1 \leq m^* \leq M \\ n^* = \delta - \beta + 1 \wedge 1 \leq n^* \leq N \end{cases} \Rightarrow \begin{cases} 0 \leq \gamma - \alpha \leq M - 1 \\ 0 \leq \delta - \beta \leq N - 1 \end{cases}$$

Defining

$$\begin{aligned} \alpha &= \lfloor \frac{i-1}{H'} \rfloor + 1 \\ \beta &= ((i-1) \mod W' + 1) \\ \gamma &= \lfloor \frac{j-1}{H} \rfloor + 1 \\ \delta &= ((j-1) \mod W + 1) \end{aligned}$$

we can now formalize a definition for $M(i, j)$:

$$M(i,j) = \begin{cases} w(\gamma - \alpha + 1, \delta - \beta + 1), & 0 \leq \gamma - \alpha \leq M - 1 \ \wedge \ 0 \leq \delta - \beta \leq N - 1 \\ 0, & \text{otherwise} \end{cases}$$

(c) All the operations with trainable parameters that take place in this network are a convolution with a $M \times N$ filter and the linear map present in the fully connected layer at the end. The last operation takes as input the vectorization of the result of the 2×2 max-pooling operation with stride 2, which has size

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

and maps it to $\mathbb{R}^3$. Hence, this architectures yields a total number of parameters of $MN + 3\left(\lfloor \frac{H-M+1}{2} \rfloor \times \lfloor \frac{W-N+1}{2} \rfloor\right)$ (ignoring bias terms).
If instead of the convolutional and max-pooling layers we had a single fully connected layer yielding an output with the same dimension as $\boldsymbol{h}_2$ (i.e., mapping $\mathbb{R}^{HW}$ to $\mathbb{R}^{H''W''}$), we would have

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

parameters (ignoring bias terms). Notice that the space complexity of the first solution in terms of the image size is $O(HW)$, which is clearly superior to the complexity of $O(H^2W^2)$ that the solution that uses only fully connected layers exhibits.

2. Considering that all projection matrices are equal to 1, the matrices of queries, keys and values are defined as:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{x'}\, \boldsymbol{W_Q} = \boldsymbol{x'} \cdot 1 = \boldsymbol{x'} \\ \boldsymbol{K} &= \boldsymbol{x'}\, \boldsymbol{W_K} = \boldsymbol{x'} \cdot 1 = \boldsymbol{x'} \\ \boldsymbol{V} &= \boldsymbol{x'}\, \boldsymbol{W_V} = \boldsymbol{x'} \cdot 1 = \boldsymbol{x'} \end{aligned}$$

and, as such, given that the length of query/key vectors satisfies $d_{\boldsymbol{K}} = 1$, the matrix of attention probabilities is defined as:

$$\boldsymbol{P} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_{\boldsymbol{K}}}} \right) = \text{softmax}(\boldsymbol{x'}\boldsymbol{x'}^T)$$

which means that each element of $\boldsymbol{P}$ is defined as:

$$P_{ij} = \frac{\exp\{x'_i x'_j\}}{\sum_{k=1}^{HW} \exp\{x'_i x'_k\}}$$

Thus, $\boldsymbol{P}$ can be seen as an entry-wise division of two matrices, $\mathcal{M}$, and $\mathcal{N}$, where

$$\mathcal{M}_{ij} = \exp\{x'_i x'_j\} \Rightarrow \mathcal{M} = \exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}$$

where the exponential is taken entry-wise. Moreover:

$$\mathcal{N}_{ij} = \sum_{k=1}^{HW} \exp\{x'_i x'_k\} \Rightarrow \mathcal{N} = \begin{bmatrix} \sum_{k=1}^{HW} \exp\{x'_1 x'_k\} \\ \sum_{k=1}^{HW} \exp\{x'_2 x'_k\} \\ \vdots \\ \sum_{k=1}^{HW} \exp\{x'_{HW} x'_k\} \end{bmatrix} \big[1, 1, \dots 1\big] = \exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}\mathbf{1}\mathbf{1}^T$$

Hence, $\boldsymbol{P}$ can be written in matrix form as

$$\boldsymbol{P} = \frac{\exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}}{\exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}\mathbf{1}\mathbf{1}^T}$$

where division and exponentiation are performed in an entry-wise fashion. Given that the attention output $\boldsymbol{Z}$ is defined as $\boldsymbol{PV}$, it can be expressed in terms of $\boldsymbol{x'}$ in the following manner:

$$\boldsymbol{Z} = \boldsymbol{PV} = \frac{\exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}}{\exp\Big\{\boldsymbol{x'}\boldsymbol{x'}^T\Big\}\mathbf{1}\mathbf{1}^T}\,\boldsymbol{x'}$$

having the i-th in the following form ($i \in \{1, \dots, HW\}$):

$$Z_i = \frac{\sum_{j=1}^{HW} \exp\{x'_i x'_j\} x'_j}{\sum_{k=1}^{HW} \exp\{x'_i x'_k\}}$$

Question 2

1. A Convolutional Neural Network (CNN) has fewer parameters than a fully connected network with the same input size and same number of classes because of the way that the layers that make up the network perform computations. In a fully connected network, every neuron of a layer is connected to every neuron of the layer before, with the number of parameters of the layer being obtained by multiplying the number of units in the previous layer by the number of units in the current layer. This means that by stacking additional layers, the number of parameters will grow in a rapid manner. In contrast, a CNN computes a convolution between a kernel and a shifting image window instead of taking into account the input in its integrity, with the size of the kernel being usually much smaller than that of the image. Even though CNNs also contain fully connected layers, they usually appear in later stages of the network pipeline, and as such their number of units is much smaller than the size of the input that is fed to the network.

2. On one hand, the use of filters in CNNs configures an operation that takes into account spatial structure, and as such is successful in capturing features that are relevant to the discrimination of the contents of an image (e.g., characteristic edges in characters and numbers). On another hand, the use of pooling layers endows CNNs with the capability of recognizing objects regardless of their shift, rotation and scale variability.
   Fully connected layers are insensitive to spatial structure in virtue of the blind vectorization of the input and the use of successive afine maps and activation functions that consider all of the inputs at all stages of the classification pipeline. Thus, they are usually outperformed by CNNs in image classification tasks, such as recognition of numbers and characters, among others.

3. In the absence of spatial correlation and structure, the architecture of CNNs may be underparametrized in comparison with neural networks of fully connected layers and as such may not be able to achieve a level of generalization in such tasks as the latter does. The explanation for that lies in the possibility that various spatially uncorrelated inputs may be now jointly relevant to the extraction of an important feature that would go unrecognized in CNNs due to the inexistence of an output unit that would weigh said units in some convolutional layer.

4. The convolutional network was implemented and tested for 20 epochs, using Adam tuning with the following learning rate values.

	Learning Rate		
	0.00001	0.0005	0.01
Validation Accuracy	0.9559	0.9843	0.9298

The best observed configuration was the one with a learning rate of 0.0005. The plots of the training loss and validation accuracy over the number of epochs for that configuration are shown below.

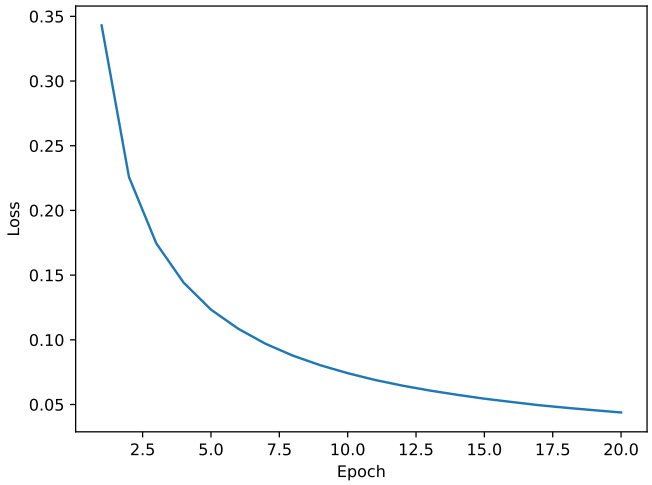

Figure 1: Training Loss over 20 epochs for $\eta = 0.0005$

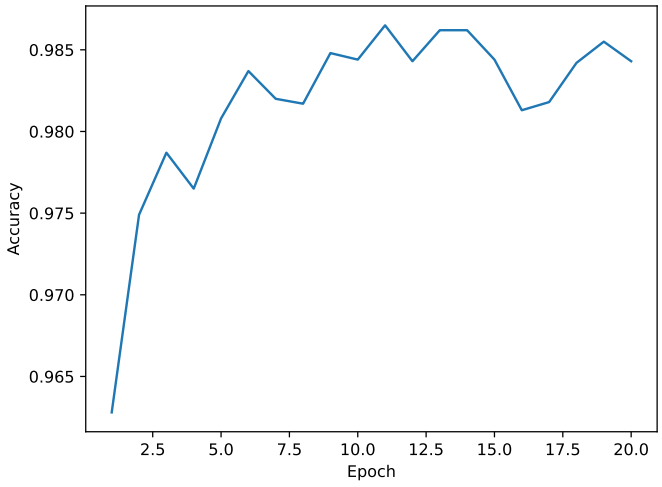

Figure 2: Validation Accuracy over 20 epochs for $\eta = 0.0005$

5. The obtained activations maps and the corresponding original example are as follows:

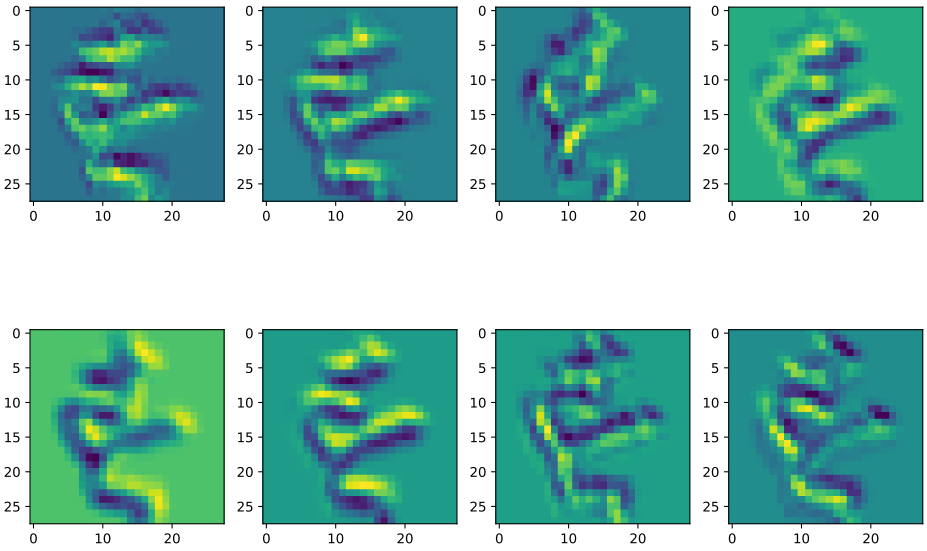

Figure 3: Activation maps of the first convolutional layer

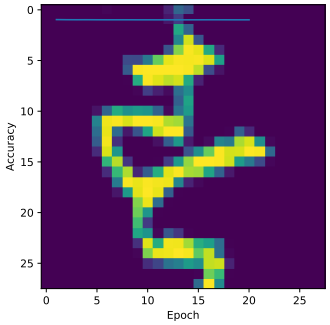

Figure 4: Original Training Image

Through the comparison of the various activation maps and the original training example, we can verify that each activation map highlights different sets of distinctive features of the character in the original image (namely, different sets of edges and corners appear yellow and dark blue in each activation map).

Question 3

1. (a) After implementing the `forward()` methods for both the encoder and decoder, the model was trained over 50 epochs, using a learning rate of 0.003, a dropout rate of 0.3, a hidden size of 128, and a batch size of 64. The final validation error rate was 0.4995 and the final test error rate was 0.5001. The validation error rate over the epochs was also plotted.

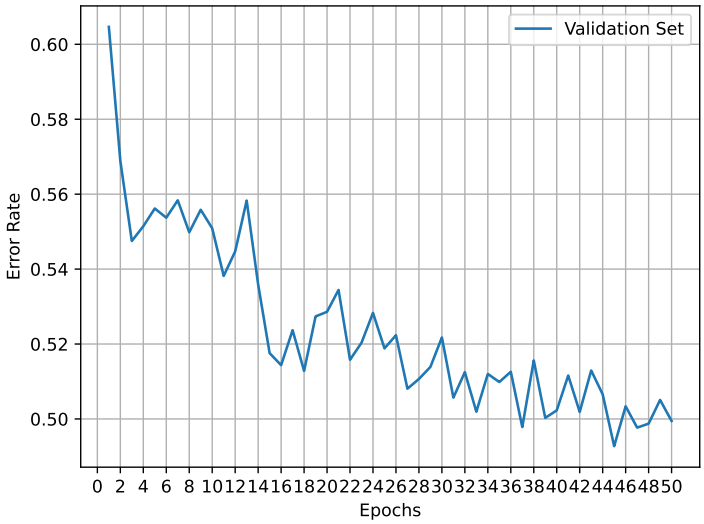

Figure 5: Validation error rate over 50 epochs

(b) After adding the bilinear attention mechanism to the decoder, the model was trained with the same hyperparameters used in the previous exercise. The final validation error rate was 0.3523 and the final test error rate was 0.3629. The validation error rate over the epochs was also plotted.

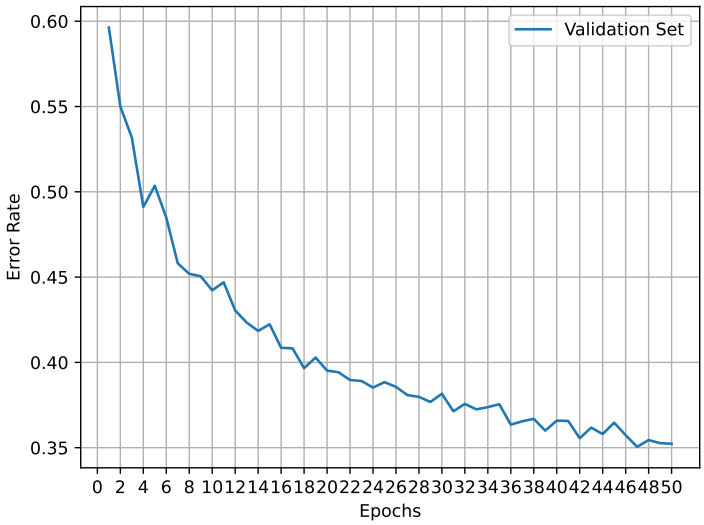

Figure 6: Validation error rate over 50 epochs for the model with attention mechanism

(c) At test time, given a source sequence $\boldsymbol{x}$, the predicted target token is computed at time step t in the following way:

$$\hat{y}_t = \underset{y_t}{\text{argmax}}\ \mathbb{P}(y_t|\text{START}, \hat{y}_1, \ldots, \hat{y}_{t-1}, \boldsymbol{x})$$

and thus a purely greedy search method is used to find $\text{argmax}_{\boldsymbol{y}}\ \mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$ (i.e., the most probable target sequence given the source sequence). Not only does this approach greatly reduces the original search space, it can also critically compromise the rest of the predicted translated sequence once a wrong token is produced. Thus, we could obtain better translations while maintaining the used architecture simply by employing **beam search** with some beam size k, where k is chosen so as to achieve a desirable balance between speed and accuracy. More specifically, at each time step t we maintain a set of k candidate partial target sequences $S_t = \{\boldsymbol{y}_t^{(1)}, \ldots, \boldsymbol{y}_t^{(k)}\}$, with $S_0 = \{\text{START}\}$. At each time step $t+1$, we proceed to expand S_t in the following way:

$$S_{t+1}^* = \bigcup_{i=1}^{k} \left\{ (\boldsymbol{y}_t^{(i)}, \hat{y}) \mid \hat{y} \in \underset{\hat{y}}{\overset{k}{\text{argmax}}}\, \mathbb{P}(\hat{y}|\boldsymbol{y}_t^{(i)}, \boldsymbol{x}) \right\}$$

$$S_{t+1} = \underset{\boldsymbol{y} \in S_{t+1}^*}{\overset{k}{\text{argmax}}}\, \mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$$

where the notation $\text{argmax}_{x \in D}^{k} f(x)$ denotes the elements of D that yield the k largest values of $f(x)$ among all the elements of D. Thus, we can obtain better approximate solutions using this method, as it expands the search space in comparison to the previous method and mitigates the previous problem of sticking to a unique choice of $\hat{y}_t$ by maintaining various candidate partial target sequences.