Instituto Superior Técnico

Master Degree in Applied Mathematics and Computation

Deep Learning

Homework 2

Catarina Franco (102678)

Mariana Henriques (103165)

January 2025

Content

Question 1. **1**
1. . . . 1
2. . . . 3
3. . . . 4

Question 2. **5**
1. . . . 5
2. . . . 8
3. . . . 8
4. . . . 9

Question 3. **9**
1. . . . 9
   a) . . . 9
   b) . . . 9
   c) . . . 10

Note: We both contributed equally to this project. One of us focused on Question 2 (Mariana Henriques) and the other handled Question 3 (Catarina Franco). For Question 1, we worked together to combine our ideas.

Question 1.

1.

Let

- $E(q) = -\text{Ise}(\beta, Xq) + \beta^{-1}\log N + \frac{1}{2}q^Tq + \frac{1}{2}M^2$;
- $E_1(q) = \frac{1}{2}q^Tq + \beta^{-1}\log N + \frac{1}{2}M^2$;
- $E_2(q) = -\text{Ise}(\beta, Xq)$.

1. $E(q) = E_1(q) + E_2(q)$:

$$E(q) = -\text{Ise}(\beta, Xq) + \beta^{-1}\log N + \frac{1}{2}q^Tq + \frac{1}{2}M^2 = \frac{1}{2}q^Tq + \beta^{-1}\log N + \frac{1}{2}M^2 - \text{Ise}(\beta, Xq) = E_1(q) + E_2(q)$$

2. Gradient and Hessian of E_1:

$\nabla E_1(q) = \frac{\partial}{\partial q}\left(\frac{1}{2}q^Tq + \beta^{-1}\log N + \frac{1}{2}M^2\right) = \frac{\partial}{\partial q}\left(\frac{1}{2}q^Tq\right) = q$

$H_{E_1}(q) = \frac{\partial^2}{\partial^2 q}\left(\frac{1}{2}q^Tq + \beta^{-1}\log N + \frac{1}{2}M^2\right) = \frac{\partial}{\partial q}(q) = I \qquad$ (Identity Matrix)

3. Hessian of E_1 is positive semi-definite:

> A square matrix A is said to be diagonally dominant if $|A_{ii}| \geq \sum_{i\neq j}|A_{i,j}|$, for any i. A symmetric diagonally dominant matrix with non-negative diagonal elements is a positive semi-definite.

- $H_{E_1}(q) = I = \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & ... & 0 & 1 \end{bmatrix} \Longrightarrow \begin{cases} |I_{ii}| = 1 & \text{(non-negative diagonal elements)} \\ \sum_{j\neq i}|I_{ij}| = 0 & \text{(because } |I_{ij}| = 0, \forall\ i \neq j) \end{cases} \Longrightarrow |I_{ii}| \geq \sum_{j\neq i}|I_{i,j}|, \forall i;$
- I is a symmetric matrix, because $I = I^T$.

Thus, since the hessian of E_1, H_{E_1}, is a symmetric diagonally dominant matrix with non-negative diagonal elements, H_{E_1} is positive semi-definite.

4. E_1 is a convex function:

> f is a convex function if and only if the hessian of f is positive semi-definite.

Since the hessian of E_1, H_{E_1}, is positive semi-definite, then E_1 is a convex function.

5. Gradient and Hessian of $-E_2$:

$-E_2(q) = \text{Ise}(\beta, Xq) = \beta^{-1}\log(\sum_{i=1}^N \exp(\beta(Xq)_i))$

$\nabla(-E_2(q)) = \frac{\partial}{\partial q}\left(\beta^{-1}\log(\sum_{i=1}^N \exp(\beta(Xq)_i))\right) = \beta^{-1}\frac{\sum_{i=1}^N\left(\frac{\partial}{\partial q}(\exp(\beta(Xq)_i))\right)}{\sum_{i=1}^N \exp(\beta(Xq)_i)} = \beta^{-1}\beta\frac{\sum_{i=1}^N X_i\ \exp(\beta(Xq)_i)}{\sum_{i=1}^N \exp(\beta(Xq)_i)} =$

$= \frac{\sum_{i=1}^N X_i\ \exp(\beta(Xq)_i)}{\sum_{i=1}^N \exp(\beta(Xq)_i)} = X^T\text{softmax}(\beta Xq)$

Let

$$\text{softmax}_i(z) = \frac{exp(z_i)}{\sum_{k=1}^n exp(z_k)}.$$

The Jacobian matrix of softmax function is

$$J = \nabla_z\ \text{softmax}(z) = \begin{bmatrix} \frac{\partial}{\partial z_1}\text{softmax}_1(z) & \frac{\partial}{\partial z_1}\text{softmax}_2(z) & ... & \frac{\partial}{\partial z_1}\text{softmax}_n(z) \\ \frac{\partial}{\partial z_2}\text{softmax}_1(z) & \frac{\partial}{\partial z_2}\text{softmax}_2(z) & ... & \frac{\partial}{\partial z_2}\text{softmax}_n(z) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial}{\partial z_n}\text{softmax}_1(z) & \frac{\partial}{\partial z_n}\text{softmax}_2(z) & ... & \frac{\partial}{\partial z_n}\text{softmax}_n(z) \end{bmatrix},$$

where

$$\frac{\partial}{\partial z_i}\text{softmax}_i(z) = \frac{exp(z_i)}{\sum_{k=1}^n exp(z_k)} - \frac{exp(z_i)\cdot exp(z_i)}{(\sum_{k=1}^n exp(z_k))^2} = \frac{exp(z_i)}{\sum_{k=1}^n exp(z_k)}\left(1 - \frac{exp(z_i)}{\sum_{k=1}^n exp(z_k)}\right) =$$

$$= \text{softmax}_i(z)\ (1 - \text{softmax}_i(z))$$

and

$$\frac{\partial}{\partial z_j}\text{softmax}_i(z) = -\frac{exp(z_i)\cdot exp(z_j)}{(\sum_{k=1}^n exp(z_k))^2} = -\frac{exp(z_i)}{\sum_{k=1}^n exp(z_k)} \cdot \frac{exp(z_j)}{\sum_{k=1}^n exp(z_k)} = -\text{softmax}_i(z)\cdot\text{softmax}_j(z).$$

Thus,

$$J = \begin{bmatrix} \text{softmax}_1(z)\ (1-\text{softmax}_1(z)) & -\text{softmax}_1(z)\cdot\text{softmax}_2(z) & ... & -\text{softmax}_1(z)\cdot\text{softmax}_n(z) \\ -\text{softmax}_2(z)\cdot\text{softmax}_1(z) & \text{softmax}_2(z)\ (1-\text{softmax}_2(z)) & ... & -\text{softmax}_2(z)\cdot\text{softmax}_n(z) \\ \vdots & \vdots & \vdots & \vdots \\ -\text{softmax}_n(z)\cdot\text{softmax}_1(z) & -\text{softmax}_n(z)\cdot\text{softmax}_2(z) & ... & \text{softmax}_n(z)\ (1-\text{softmax}_n(z)) \end{bmatrix} =$$

$$= \begin{bmatrix} \text{softmax}_1(z) & 0 & ... & 0 \\ 0 & \text{softmax}_2(z) & ... & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & \text{softmax}_n(z) \end{bmatrix} -$$

$$+ \begin{bmatrix} \text{softmax}_1(z)\cdot\text{softmax}_1(z) & \text{softmax}_1(z)\cdot\text{softmax}_2(z) & ... & \text{softmax}_1(z)\cdot\text{softmax}_n(z) \\ \text{softmax}_2(z)\cdot\text{softmax}_1(z) & \text{softmax}_2(z)\cdot\text{softmax}_2(z) & ... & \text{softmax}_2(z)\cdot\text{softmax}_n(z) \\ \vdots & \vdots & \vdots & \vdots \\ \text{softmax}_n(z)\cdot\text{softmax}_1(z) & \text{softmax}_n(z)\cdot\text{softmax}_2(z) & ... & \text{softmax}_n(z)\cdot\text{softmax}_n(z) \end{bmatrix} =$$

$$= \text{diag}(\text{softmax}(z)) - \text{softmax}(z)\cdot\text{softmax}(z)^T.$$

$H_{(-E_2)}(q) = \frac{\partial^2}{\partial q^2}\left(\beta^{-1}\log(\sum_{i=1}^N \exp(\beta(Xq)_i))\right) = \frac{\partial}{\partial q}(X^T\text{softmax}(\beta Xq)) = X^T\cdot J\cdot\beta X = X^T(\text{diag}(\text{softmax}(\beta Xq)) -$

$+\ \text{softmax}(\beta Xq)\cdot\text{softmax}(\beta Xq)^T)\beta X = X^T\beta(\text{diag}(\text{softmax}(\beta Xq)) - \text{softmax}(\beta Xq)\cdot\text{softmax}(\beta Xq)^T)X$

6. Hessian of $-E_2$ is positive semi-definite:

Considering $\theta_i(z) = \text{softmax}_i(z)$, notice that

$$|J_{ii}| = \theta_i(1-\theta_i), \forall i = \{1, \cdots, n\} \qquad \text{and} \qquad |J_{ij}| = \theta_i\cdot\theta_j, \forall i \neq j.$$

Then, for some i,

$$\sum_{j\neq i}|J_{i,j}| = \sum_{j\neq i}\theta_i\cdot\theta j = \theta_i\sum_{j\neq i}\theta_j = \theta_i\left(\frac{\sum_{j\neq i}^n exp(z_j)}{\sum_{j=1}^n exp(z_j)}\right)$$

$$|J_{ii}| = \theta_i(1-\theta_i) = \theta_i\left(1 - \frac{exp(z_i)}{\sum_{j=1}^n exp(z_j)}\right) = \theta_i\left(\frac{\sum_{j=1}^n exp(z_j) - exp(z_i)}{\sum_{j=1}^n exp(z_j)}\right) = \theta_i\left(\frac{\sum_{j\neq i}^n exp(z_j)}{\sum_{j=1}^n exp(z_j)}\right)$$

$$\implies |J_{ii}| = \sum_{j\neq i}|J_{i,j}| \quad \implies \quad |J_{ii}| \geq \sum_{j\neq i}|J_{i,j}|,\ \forall j$$

Thus, since J is a symmetric diagonally dominant matrix with non-negative diagonal elements, then J is positive semi-definite. Thus, by definition,

$$x^T J\ x \geq 0,\ \forall x \in \mathbb{R}^{\kappa}. \tag{1}$$

Thus, noticing that, $\forall y \in \mathbb{R}^n$,

$$y^T X^T J X y = (Xy)^T J (Xy). \tag{2}$$

Taking $x = Xy \in \mathbb{R}^n$, then

$$(2) = x^T J x \overset{(1)}{\geq} 0 \quad \implies \quad y^T(X^T J X)y \geq 0,\ \forall y \in \mathbb{R}^n.$$

Thus, by definition, the hessian of $(-E_2)$, $H_{(-E_2)} = X^T J X$, is positive semi-definite.

7. $-E_2$ is a convex function:

> f is a convex function if and only if the hessian of f is positive semi-definite.

Since the hessian of $-E_2$, $H_{(-E_2)}$, is positive semi-definite, then $-E_2$ is a convex function.

8. E_2 is a concave function:

> if $-f$ is a convex function, then f is a concave function.

Since $-E_2$ is a convex function, then E_2 is a concave function.

2.

The algorithm CCCP works as follow: at each iteration t,

- linearize the concave function E_2 using a first-order Taylor approximation around q_t,

$$E_2(q) \approx \tilde{E}_2 := E_2(q_t) + (\nabla E_2(q_t))^T (q - q_t); \tag{3}$$

- computes a new iterate by solving the convex optimization problem

$$q_{t+1} = \arg\min_q E_1(q) + \tilde{E}_2(q). \tag{4}$$

From (3) and taking into account that

$$\nabla E_2(q_t) = -X^T \text{softmax}(\beta X q) \tag{5}$$

and

$$E_1(q) = \frac{1}{2} q^T q + \beta^{-1} \log N + \frac{1}{2} M^2, \tag{6}$$

we have that

$$E_2(q) \approx \tilde{E}_2 := E_2(q_t) + (\nabla E_2(q_t))^T (q - q_t) \overset{(5)}{=} E_2(q_t) - (X^T \text{softmax}(\beta X q_t))^T (q - q_t)$$

$$\implies \quad E(q) = E_1(q) + E_2(q) \approx E_1(q) + \tilde{E}_2(q) := E_1(q) + E_2(q_t) - (X^T \text{softmax}(\beta X q_t))^T (q - q_t) \overset{(6)}{=} \frac{1}{2} q^T q + \beta^{-1} \log N +$$

$$+ \frac{1}{2} M^2 + E_2(q_t) - (X^T \text{softmax}(\beta X q_t))^T (q - q_t)$$

$$\implies \quad E_1(q) + \tilde{E}_2(q) := \frac{1}{2} q^T q + \beta^{-1} \log N + \frac{1}{2} M^2 + E_2(q_t) - (X^T \text{softmax}(\beta X q_t))^T (q - q_t). \tag{7}$$

From (4), we have that

$$q_{t+1} = \arg\min_q (E_1(q) + \tilde{E}_2(q)) \overset{(7)}{=} \arg\min_q \left(\frac{1}{2} q^T q + \beta^{-1} \log N + \frac{1}{2} M^2 + E_2(q_t) - (X^T \text{softmax}(\beta X q_t))^T (q - q_t) \right) =$$

$$= \arg\min_q \left(\frac{1}{2} q^T q - (X^T \text{softmax}(\beta X q_t))^T q \right), \tag{8}$$

because we can drop constant terms independent of q of the objetive.
Since (8) is a optimization problem, we must have

$$\frac{\partial}{\partial q} \left(\frac{1}{2} q^T q - (X^T \text{softmax}(\beta X q_t))^T q \right) = 0 \quad \Leftrightarrow \quad q - X^T \text{softmax}(\beta X q_t) = 0 \quad \Leftrightarrow \quad q = X^T \text{softmax}(\beta X q_t)$$

$$\implies \quad q_{t+1} = \arg\min_q (E_1(q) + \tilde{E}_2(q)) = X^T \text{softmax}(\beta X q_t)$$

3.

Notice that the first update is given by

$$q_{t+1} = X^T \text{softmax}(\beta X q_t),$$

where

- $X \in \mathbb{R}^{N \times D}$ is the input matrix;
- $\beta = \frac{1}{\sqrt{D}} \in \mathbb{R}$;
- $q_t \in \mathbb{R}^D$ is the current state vector (query).

Considering now

- the query matrix $Q = [q_t^{(1)}, \cdots, q_t^{(N)}]^T \in \mathbb{R}^{N \times D}$ (already computed);
- the key matrix $K \in \mathbb{R}^{N \times D}$;
- the value matrix $V \in \mathbb{R}^{N \times D}$;
- identity projection matrices $W_K = W_V = I$,

then the computation perfomed in the cross-attention layer of a transformer with a single attention head, identity projection matrices $W_K = W_V = I$ with input $X \in \mathbb{R}^{N \times D}$, is given by

$$Q' = \text{Attention}(Q, K, V) = \text{softmax}(\beta Q K^T)V = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V.$$

It is important to notice that, since $W_K = W_V = I$, then $K = V = X$. Thus,

$$Q' = \text{Attention}(Q, X, X) = \text{softmax}(\beta Q X^T)X = \text{softmax}\left(\frac{QX^T}{\sqrt{D}}\right)X.$$

Taking $Q = q_t$, then

$$Q' = \text{Attention}(q_t, X, X) = \text{softmax}\left(\frac{q_t X^T}{\sqrt{D}}\right)X.$$

Thus, the first update $q_t \mapsto q_{t+1}$ is identical to the computation performed in the cross-attention layer of a transformer with the single attention head, identity projection matrices $W_K = W_V = I$ with input $X \in \mathbb{R}^{N \times D}$.

Question 2.

1.

For a learning rate of 0.1 we obtained the following results:

- Final validation accuracy: 0.5783;
- Final test accuracy: 0.5763.
- The train loss curve and the validation accuracy curve as a function of the epoch number:

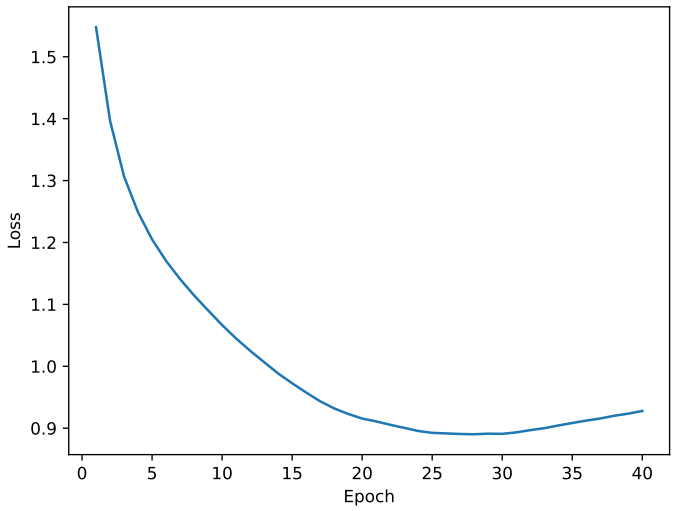

Figure 1. Train loss curve for a learning rate of 0.1.

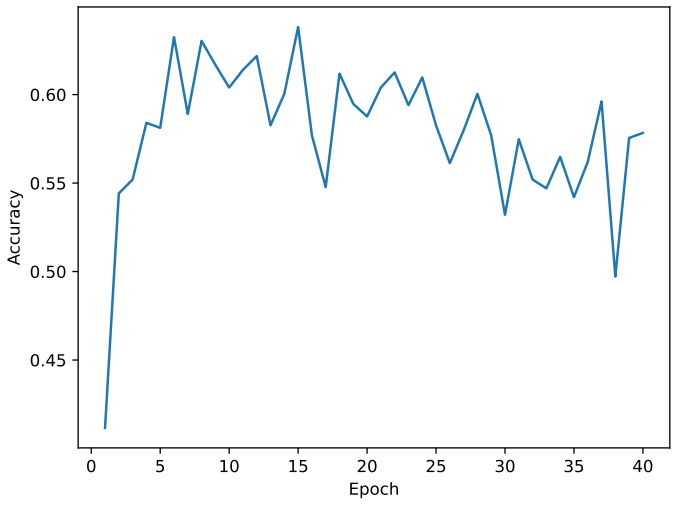

Figure 2. Validation accuracy for a learning rate of 0.1.

For a learning rate of 0.01 we obtained the following results:

- Final validation accuracy: 0.6937;
- Final test accuracy: 0.6860.
- The train loss curve and validation accuracy curve as a function of the epoch number:

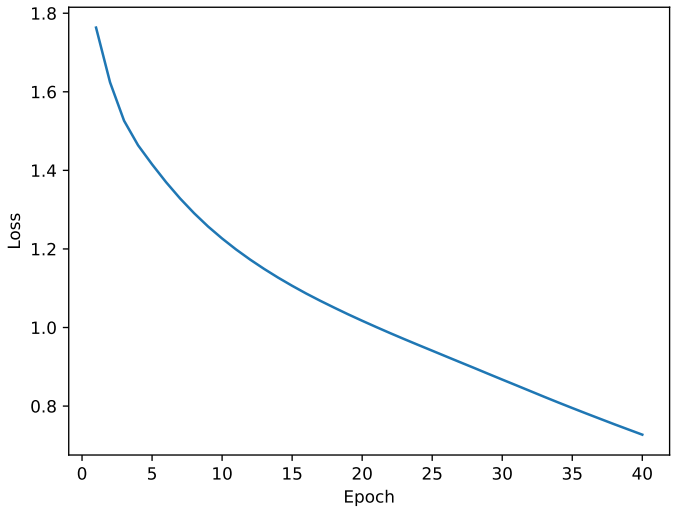

Figure 3. Train loss curve for a learning rate of 0.01.

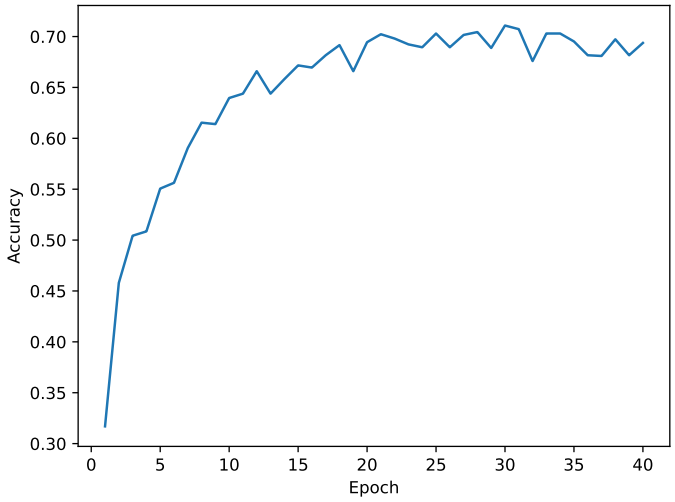

Figure 4. Validation accuracy for a learning rate of 0.01.

For a learning rate of 0.001 we obtained the following results:

- Final validation accuracy: 0.5848;
- Final test accuracy: 0.5790.
- The train loss curve and validation accuracy curve as a function of the epoch number:

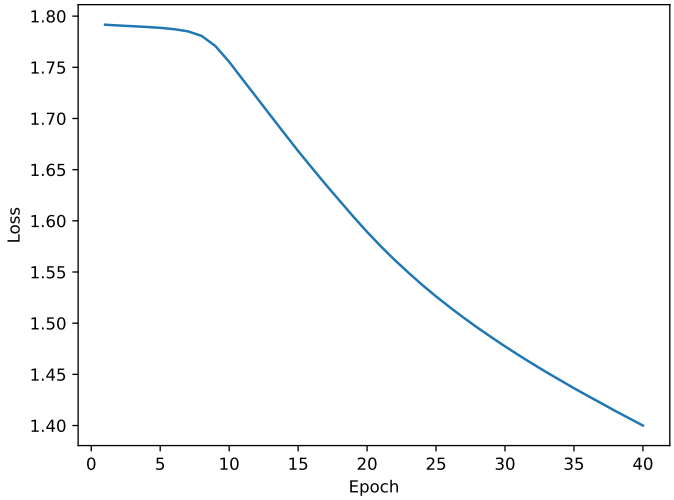

Figure 5. Train loss curve for a learning rate of 0.001.

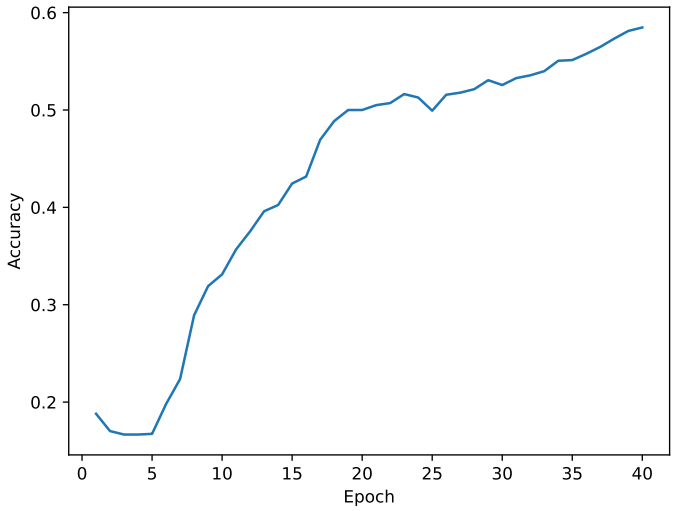

Figure 6. Validation accuracy for a learning rate of 0.001.

The best configuration was obtained with a learning rate of 0.01, since it was the one with the highest final test accuracy.

2.

For a leaning rate of 0.01, which was the optimal configuration defined in the previous question, we obtained the following results:

- Final validation accuracy was 0.7557;
- Final test accuracy was 0.7443;
- The train loss curve and validation accuracy curve as a function of the epoch number:

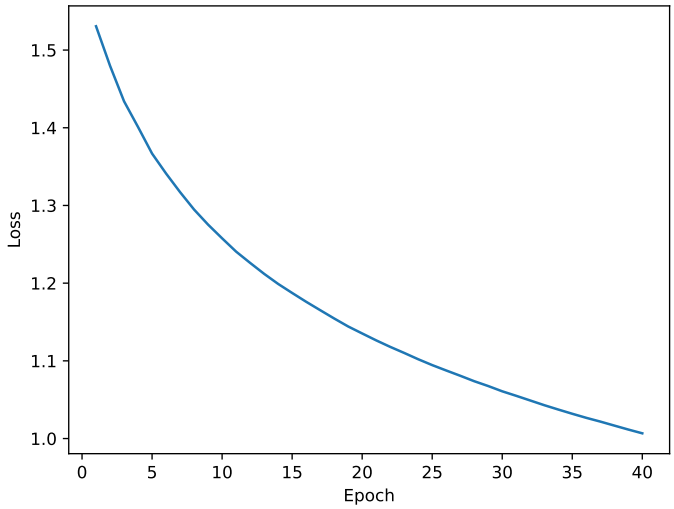

Figure 7. Train loss curve for a learning rate of 0.01 with batch normalization

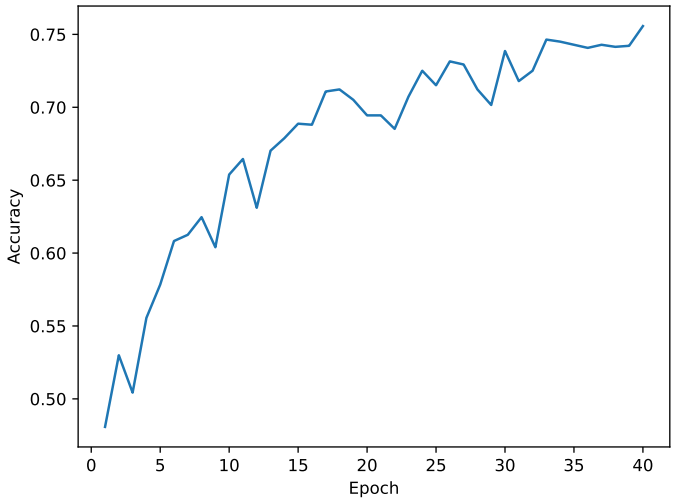

Figure 8. Validation accuracy for a learning rate of 0.01.with batch normalization

3.

In exercise 1, the total number of trainable parameters was 5340742 while in exercise 2, it was significantly reduced to 756742. This difference can be justified by the way the outputs from the convolutional layers are processed before being passed to the fully connected layers. In Exercise 1, the output of the final convolutional block is flattened directly into a one-dimensional vector. This flattening process preserves the spatial dimensions (height and width) of the feature maps, which results in a large number of input features for the first fully connected layer. Since the number of trainable parameters in a fully connected layer depends on the size of its input, the first fully connected layer ends up dominating the total parameter count, leading to a significantly higher number of trainable parameters in exercise 1.

In contrast, exercise 2 addresses this by replacing the flattening operation with global average pooling. Global average pooling reduces the spatial dimensions of the output feature maps to 1×1, effectively condensing each feature map into a single value. As a result, the input size to the first fully connected layer is drastically reduced, as it now depends solely on the number of output channels from the last convolutional block rather than the full spatial dimensions of the feature maps. This reduction in input size leads to a much smaller first fully connected layer, which accounts for the significantly lower total number of trainable parameters in exercise 2.

4.

Small kernels are preferred in convolution layers because they are more parameter-efficient, computationally lighter, and less prone to overfitting compared to larger kernels. This method adds more non-linearities, enhancing the network's ability to capture complex patterns. Additionally, small kernels make optimization easier by reducing the number of parameters and supporting better gradient flow.

Pooling layers, such as max or average pooling, reduce spatial dimensions of data, making the network computationally efficient and less sensitive to input shifts. Pooling highlights dominant features and reduces overfitting by lowering the feature map size passed to fully connected layers. However, pooling may result in some loss of fine-grained information. Together, small kernels and pooling create a balance between efficiency, generalization, and performance in convolutional neural networks.

Question 3.

1.

a)

We obtained the following results:

- Best validation error rate: 0.3061;
- Test CER: 0.3165;
- Test WER: 0.8090.

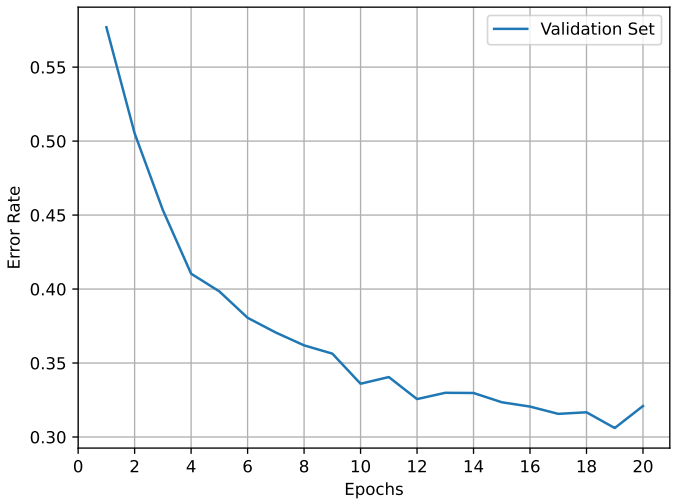

Figure 9. Validation CER as a function of the epoch number.

b)

We obtained the following results:

- Best validation error rate: 0.2138;
- Test CER: 0.2072;
- Test WER: 0.7160.

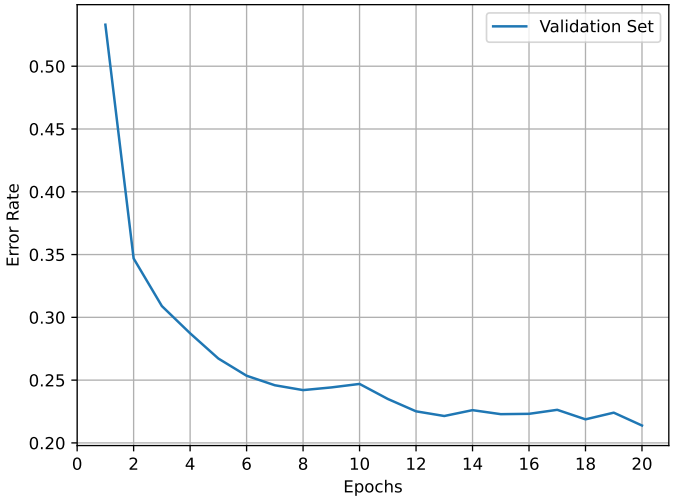

Figure 10. Validation CER as a function of the epoch number with attention.

c)

We obtained the following results:

- Test CER: 0.2255;
- Test WER: 0.7540;
- Test WER@3: 0.6400.