Homework 2 – Deep Learning (DEI)

Rafael Sargento 103344, Diogo Miranda 102536

January 2025

Question 2

Question 2.1

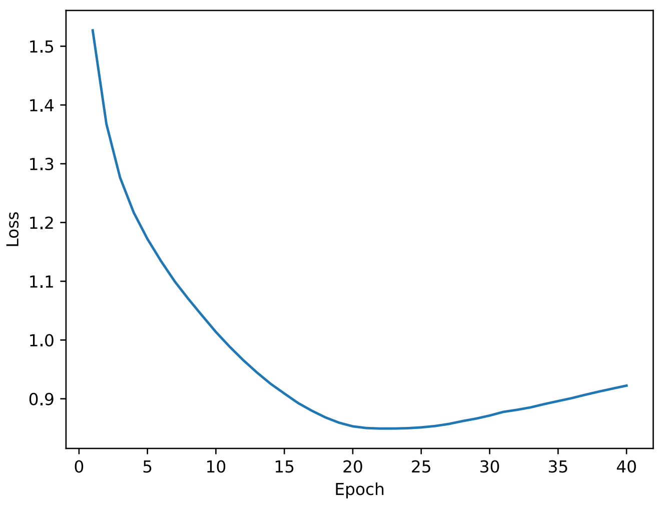
The implementation of the simple convolutional network with convolutional blocks with no batch normalization ran for **40 epochs** with three different values of learning rate: **0.1, 0.01, 0.001.**

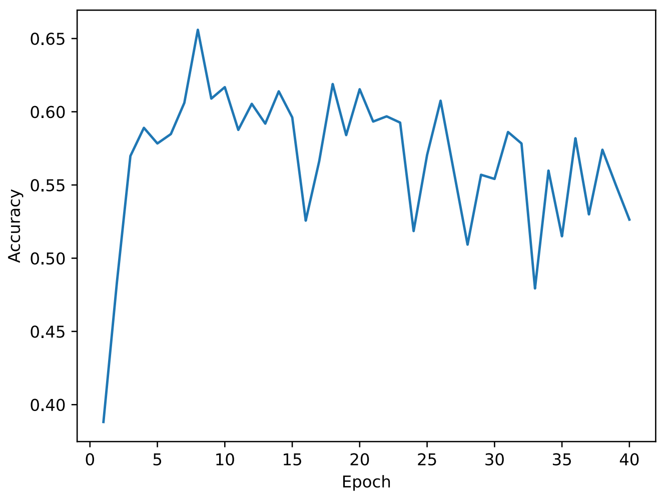
0.1 Learning Rate

The high learning rate (0.1) likely causes the model to adapt too quickly, leading to fluctuations in validation accuracy and preventing convergence. This rapid adjustment risks overfitting specific samples while missing general patterns. The rising loss after 20 epochs indicates that the model is overfitting. The final test accuracy ends up being pretty mediocre as well with **only ~50%** accuracy.

Final Test acc: 0.5093

Final Validation acc: 0.5264



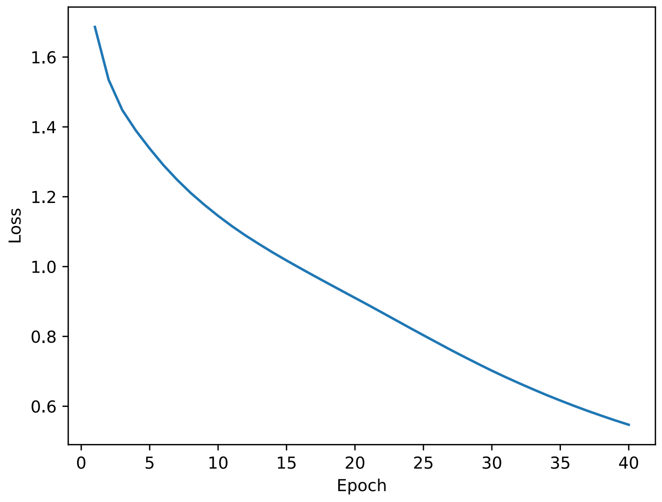


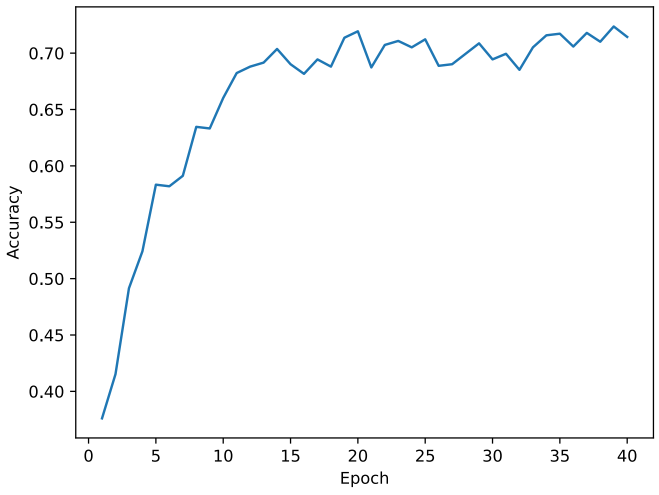
0.01 Learning Rate

The model, with a learning rate of 0.01, shows a smoother but gradual decrease in loss over the 40 epochs. This suggests that the model is learning at a steadier pace, reducing the risk of overfitting early in the training process. From the 3 rates used this one was the one with the highest final accuracy with a value of **~70% which indicates 0.01 as the best learning rate configuration.**

Final Test acc: 0.7013

Final Validation acc: 0.7144



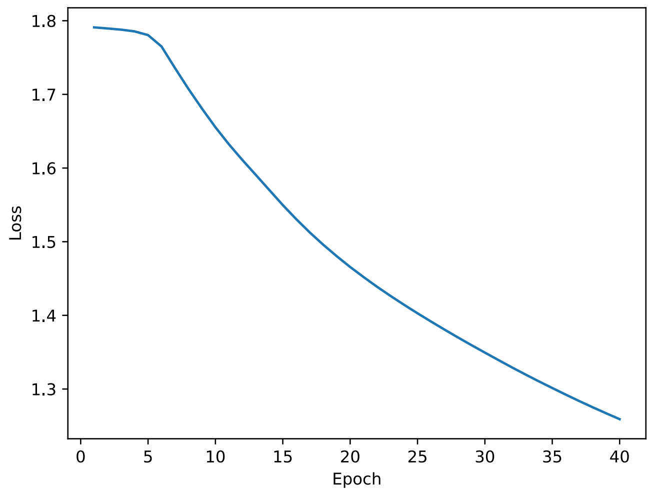


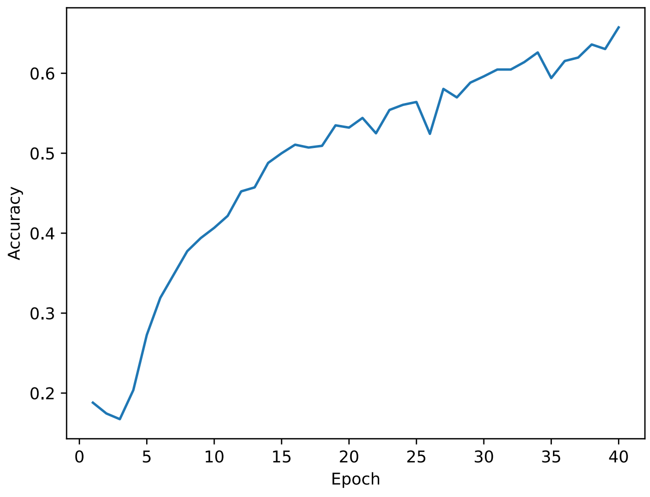
0.001 Learning Rate

The model, with a learning rate of 0.001, shows a gradual steadier decrease in loss (and a steadier increase in accuracy) over the 40 epochs. This suggests that the model is learning at a steadier pace, reducing the risk of overfitting early in the training process. However, the relatively slow rate of improvement might also indicate that the learning rate could be too conservative.

Final Test acc: 0.6380

Final Validation acc: 0.6574



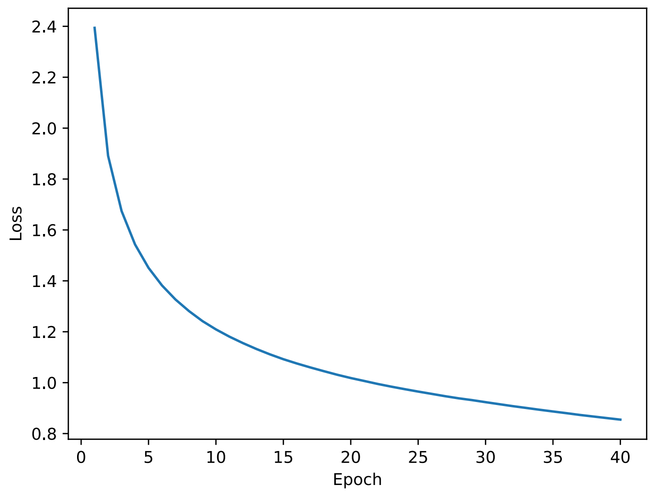


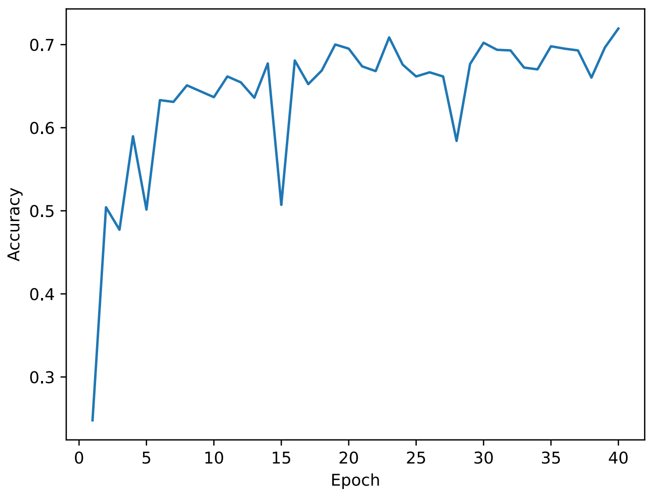
Question 2.2

0.1 Learning Rate

Final Test acc: 0.7203

Final Validation acc: 0.7194

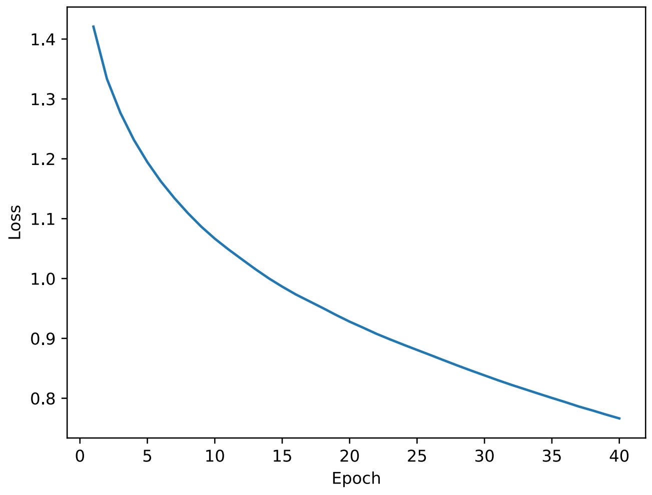


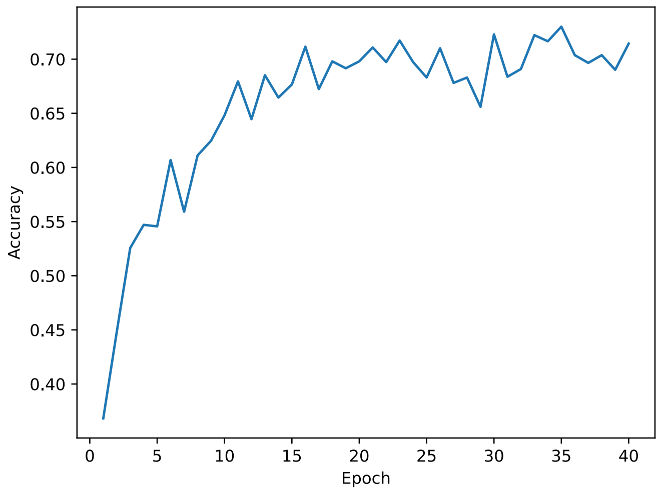


0.01 Learning Rate

Final Test acc: 0.7010

Final Validation acc: 0.7144

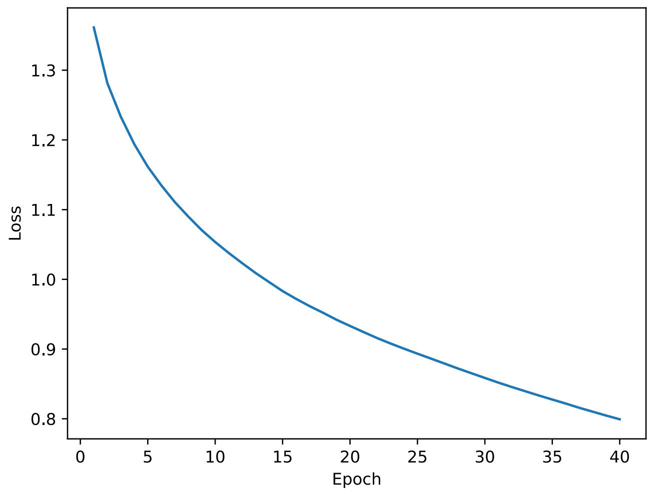


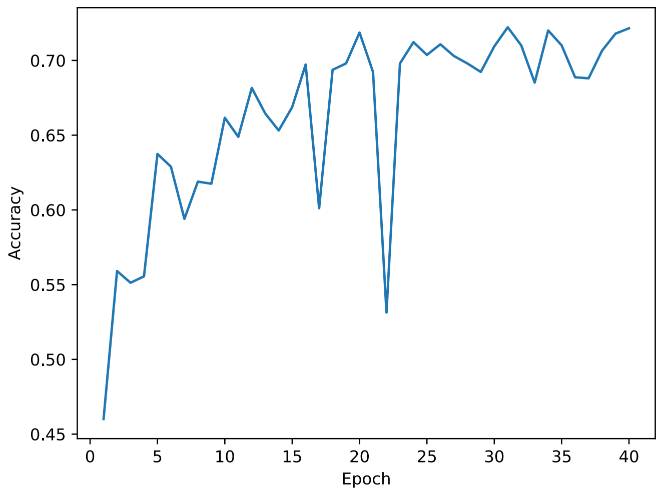


0.001 Learning Rate

Final Test acc: 0.7147

Final Validation acc: 0.7215





Comment:

After plotting the training loss and validation accuracies for the optimized configuration there's a noticeable difference in the final accuracy for 2 out of the 3 learning rates used, where we can clearly see the model improved, however, it did not perform as expected for the 0.01 learning rate, likely due to insufficient gradient updates that failed to converge effectively, suggesting a sensitivity to step size for this rate. This is also noticeable for the 0.001 learning rate, meanwhile, the final accuracy here still improves ~10%.

Question 2.3

The number of trainable parameters differs significantly between the two models. The first model has **5.340.742 parameters**, while the second has only **756.742 parameters**. This large reduction is primarily due to using **global average pooling** in the second model. Without it, **flattening the output** of the **last convolutional block** creates a very large input size for the first layer of the MLP, leading to a much **higher number of parameters**. For instance, the first fully connected layer with 1,024 outputs in the first model requires many more weights due to the large input size. In contrast, **global average pooling reduces** the output of the **last convolutional block** to a single value per channel, significantly decreasing the input size to the fully connected layer and thus **reducing the number of parameters**.

While batch normalization also adds trainable parameters, its contribution is relatively small compared to the impact of using global average pooling.

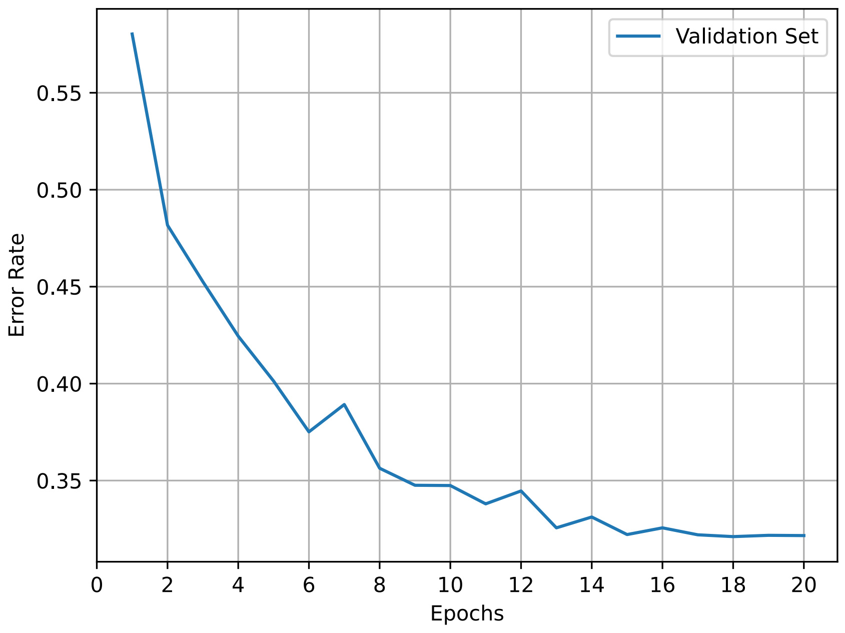
Question 2.4

The use of **smaller kernel sizes** allows the model to **focus on local patterns** which are very useful when doing fe**ature extraction**, when using **bigger kernel sizes**, or even a kernel with the size of the image, the processing becomes **computationally expensive** and **not efficient**, missing small **details** which have enormous importance in classification. **Smaller kernels** also offer the advantage of having fewer parameters, which not only **reduces the computational cost** as mentioned but also the **risk of overfitting**.

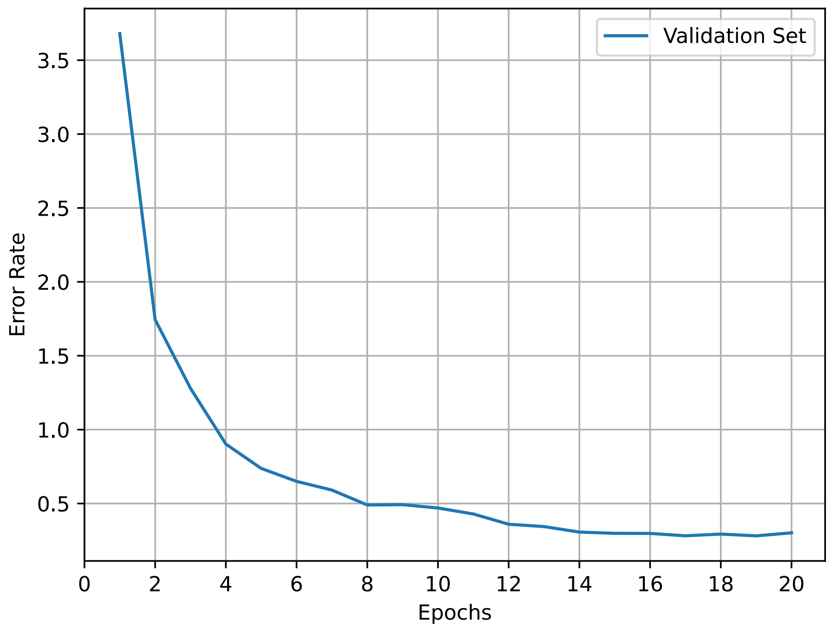
Similarly, we use **pooling layers** with a similar intent, pooling layers **reduce the spatial dimension of feature maps** which decreases **computational cost** and **memory usage** in future layers, this spatial reduction also results in **fewer trainable parameters** in fully connected layers which **reduce the risk of overfitting**. Other cool effects of **pooling layers** is retaining the most **prominent feature in a region**, which helps the model focus on the most **critical features and discard less important details or noise.**

Question 3.1

a)



Best validation error rate: 0.3211, Test CER: 0.3168, Test WER: 0.8170 b)



```
Best validation error rate: 0.2813, Test CER: 0.2585, Test WER: 0.7850
```

c)

```
Test CER: 0.2850, Test WER: 0.8380
Printing first 10 examples with multiple predictions:
('remainder', {'ramainder', 'romaender', 'romainder'})
('misperception', {'misperception', 'misperseption'})
('crudités', {'crudite', 'crudita'})
('Jingchuan', {'jinnuhu', 'ginguon', 'ginghwah'})
('scare', {'skar', 'scaer', 'scar'})
('glossal', {'glossel', 'glossal'})
('touchedness', {'tuchidnis', 'tuctideness', 'tuccidness'})
('alkaloses', {'alcalosis', 'alcalocias', 'alcolosis'})
('companions', {'companyens', 'companiens', 'companeans'})
('slaggy', {'slagy', 'slagge', 'slagey'})
Test WER@3: 0.7460
```

In our group for Homework 1, Rafael Sargento worked on solving everything related to Question 1, while Diogo Miranda focused on Questions 2 and 3. It is worth noting, however, that all answers given were discussed as a group. This means that, although one person finalized each part, the entire process of arriving at the solutions was done collaboratively.

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

$$\boxed{\nabla E_1(q) = q + 0 + 0 = q}$$

$E_2(q) = -\mathrm{lse}(\beta, Xq)$

$\mathrm{lse}(\beta, z) = \beta^{-1} \log\left(\sum_{i=1}^{N} \exp(\beta z_i)\right)$

$\nabla_z \mathrm{lse}(\beta, z) = \mathrm{softmax}(\beta z)$

$\mathrm{lse}(\beta, Xq) = \beta^{-1} \log\left(\sum_{i=1}^{N} \exp(\beta (Xq)_i)\right)$

$(Xq)_i = \sum_{j=1}^{D} X_{ij} q_j$

$\nabla_q \mathrm{lse}(\beta, Xq) = \nabla_z \mathrm{lse}(\beta, z) \cdot \nabla_q z$

$\hookrightarrow z = Xq$

$\nabla_q z = X^T$

$\nabla_q \mathrm{lse}(\beta, Xq) = X^T \mathrm{softmax}(\beta Xq)$

$E_2(q) = -\mathrm{lse}(\beta, Xq)$

$-E_2(q) = \mathrm{lse}(\beta, Xq)$

$$\boxed{\nabla(-E_2(q)) = X^T \mathrm{softmax}(\beta Xq)}$$

continues in the next page

$$\boxed{H\,E_1(q) = \nabla^2 E_1(q) = I} \rightarrow \text{identity matrix}$$

$\downarrow$

diagonal matrix with only positive eigenvalues, so it is clearly PSD

$$H(-E_2(q)) = \nabla^2(-E_2(q)) = \nabla_q(X^T \text{softmax}(\beta X q))$$

let $p = \text{softmax}(\beta X q)$ be a vector of size N

$$\nabla(-E_2(q)) = X^T p \longrightarrow \nabla^2(-E_2(q)) = X^T \nabla_q p$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$

$\uparrow$

$p = \text{softmax}(\beta z)$ where $z = Xq$

$$\nabla_q p = \nabla_z p \cdot \nabla_q z = \nabla_z p \cdot X$$

Kronecker delta

$$p_i = \text{softmax}(\beta z)_i \qquad \frac{\partial p_i}{\partial z_i} = \beta(p_i(\delta_{ij} - p_j))$$

$$\nabla_z p = \beta(\text{diag}(p) - pp^T)$$

$$\nabla_q p = \beta(\text{diag}(p) - pp^T)X$$

$$\boxed{H(-E_2(q)) = \nabla(-E_2(q)) = X^T \nabla_q p = \beta X^T(\text{diag}(p) - pp^T)X}$$

continues in the next page

$\mathrm{diag}(p) - pp^T$ is PSD iff $v^T(\mathrm{diag}(p) - pp^T)v \geq 0$ for any $v \in R^N$

$$v^T(\mathrm{diag}(p) - pp^T)v = v^T \mathrm{diag}(p) v - v^T p p^T v$$

$$= \sum_{i=1}^{N} p_i v_i^2 - \left(\sum_{i=1}^{N} p_i v_i\right)^2$$

By the Jensen's inequality $\left(\sum_{i=1}^{N} p_i v_i\right)^2 \leq \sum_{i=1}^{N} p_i v_i^2$

$$\sum_{i=1}^{N} p_i v_i^2 - \left(\sum_{i=1}^{N} p_i v_i\right)^2 \geq 0 \Leftrightarrow v^T(\mathrm{diag}(p) - pp^T)v \geq 0$$

$\Downarrow$

$\mathrm{diag}(p) - pp^T$ is PSD

$x^T(\mathrm{diag}(p) - pp^T)x$ is also PSD for any matrix X

$\Downarrow$

$M = \mathrm{diag}(p) - pp^T$, we know that $v^T M v \geq 0$ for all any vector v

$v = Xy$ $\quad y^T(X^T M X)y = (Xy)^T M (Xy) \geq 0$

$\Downarrow$

y is any vector

$$\boxed{H(-E_q(q)) = \beta x^T(\mathrm{diag}(p) - pp^T)x \quad \text{is PSD}}$$ for any $\beta \geq 0$

② from the previous exercise we know that

$$\nabla E_2(q) = -X^T \mathrm{softmax}(\beta X q)$$

$$\tilde{E}_2(q) = E_2(q_t) + \nabla E_2(q_t)^T (q - q_t)$$

$$= E_2(q_t) - (X^T \mathrm{softmax}(\beta X q_t))^T (q - q_t)$$

$$= E_2(q_t) - (X^T \mathrm{softmax}(\beta X q_t))^T q + (X^T \mathrm{softmax}(\beta X q_t))^T q_t$$

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

$$q_{t+1} = \arg\min_q \left(E_1(q) + \tilde{E}_2(q) \right)$$

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

$$\nabla \left(\frac{1}{2} q^T q - (X^T \mathrm{softmax}(\beta X q_t))^T q \right) = 0$$

$$\Leftrightarrow \quad q - X^T \mathrm{softmax}(\beta X q_t) = 0$$

$$\Leftrightarrow \quad q = X^T \mathrm{softmax}(\beta X q_t)$$

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

③ For the special case $\beta = \frac{1}{\sqrt{D}}$ the Hopfield update is

$$q_{t+1} = X^T \text{softmax}\left(\frac{1}{\sqrt{D}} X q_t\right)$$

For the transformer with a single attention head and identity projection matrices $(W_K = W_V = I)$, the cross attention is:

$$\text{Attention Output} = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) V = \text{softmax}\left(\frac{QX^T}{\sqrt{D}}\right) X$$

$Q \rightarrow$ query

$K = V = X \rightarrow$ are the keys and values because $W_K = W_V = I$

$$\boxed{\text{Comparassion}}$$

Similarities	Differences
- The attention scores are both computed using the scaled dot product $\frac{1}{\sqrt{D}} X q_t$ and are then normalized with softmax.	- In the transformer multiple queries can be processed in a batch and in the Hopfield update only a single query vector q_t is processed
- Both compute weighted aggregation of the rows of X	- Hopfield outputs a single updated query and the transformer can output a matrix when multiple queries are used with each row representing a single query output
- For $\beta = \frac{1}{\sqrt{D}}$ the Hopfield update and the single query cross attention are identical	- Hopfield is a standalone iterative algoritm and the cross attention layer is part of a broader transformer architecture

R: The Hopfield update rule and the transformer cross attention for a single query are matematically identical. The transformer can generalize to handle multiple queries simultaneously