Student Name: João Coelho
Student ID: 86448

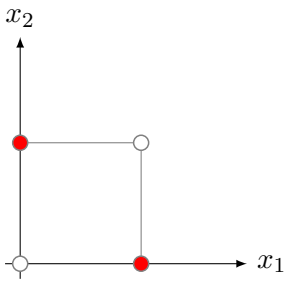
Deep Structured Learning
Homework 1

TÉCNICO LISBOA

Question 1

1a

Let $D = 2$ and $K = 1$. Computing $f(x_1, x_2)$ for the possible points yields $f(0,0) = 0$, $f(1,0) = 1$, $f(0,1) = 1$, and $f(1,1) = 0$. Consider the following plot where the points with output 1 are in red, and the points with output 0 are in white:



Note that this plot is equivalent to that of the XOR problem. There is no straight line that separates the two classes, i.e., the data is not linearly separable. This can be proved by contradiction. The output of the perceptron, o, can be computed as follows:

$$o(x_1, x_2) = \text{sign}((w_1, w_2).(x_1, x_2) + b) = \text{sign}(p(x_1, x_2)) \tag{1}$$

Given the definition of f, the following conditions must hold:

$$p(0,0) \leq 0, p(1,0) > 0, p(0,1) > 0, p(1,1) \leq 0 \tag{2}$$

It comes directly from Eq. (1):

$$0w_1 + 0w_2 + b \leq 0 \tag{3}$$

$$1w_1 + 0w_2 + b > 0 \tag{4}$$

$$0w_1 + 1w_2 + b > 0 \tag{5}$$

$$1w_1 + 1w_2 + b \leq 0 \tag{6}$$

From inequalities (3) and (6) we have $w_1 + w_2 + 2b \leq 0$. From (4) and (5), $w_1 + w_2 + 2b > 0$, which is a contradiction.

This way, a perceptron cannot learn a line that separates the two classes for the specific case of $D = 2$ and $K = 1$, since it does not exist. Hence, f cannot be generally computed by a perceptron.

1b

Consider the following notation, where $W^{[i]}$ is the weight matrix for the connection between layer $i-1$ and i, $b^{[i]}$ is the bias value(vector) for the unit(s) in layer i, and x is the input vector:

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ \vdots & \vdots \\ w_{D1}^{[1]} & w_{D2}^{[1]} \end{bmatrix}, \; b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}, \; W^{[2]} = \begin{bmatrix} w_1^{[2]} \\ w_2^{[2]} \end{bmatrix}, \; b^{[2]} = b_1^{[2]}, \; x = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}.$$

The output of the MLP, o, can be computed by $o = g(z^T W^{[2]} + b_1^{[2]})$, where $z = g(x^T W^{[1]} + b^{[1]})$. This can be written as:

$$\begin{aligned} o &= g(\begin{bmatrix} g(\sum_{i=1}^{D}(x_i w_{i1}^{[1]}) + b_1^{[1]}) & g(\sum_{i=1}^{D}(x_i w_{i2}^{[1]} b_1^{[2]})) \end{bmatrix} \begin{bmatrix} w_1^{[2]} \\ w_2^{[2]} \end{bmatrix} + b_1^{[2]}) \\ &= g(w_1^{[2]} g(\sum_{i=1}^{D}(x_i w_{i1}^{[1]}) + b_1^{[1]}) + w_2^{[2]} g(\sum_{i=1}^{D}(x_i w_{i2}^{[1]}) + b_2^{[1]}) + b_1^{[2]}). \end{aligned} \tag{7}$$

In this exercise, assuming that the output layer also uses heavyside activation, we need to find integer-valued $W^{[1]}$, $W^{[2]}$, $b^{[1]}$, and $b^{[2]}$ such that:

$$\begin{cases} w_1^{[2]} g(\sum_{i=1}^{D}(x_i w_{i1}^{[1]}) + b_1^{[1]}) + w_2^{[2]} g(\sum_{i=1}^{D}(x_i w_{i2}^{[1]}) + b_2^{[1]}) + b_1^{[2]} \geq 0, \; \sum_{i=1}^{D} x_i = K \,, \\ w_1^{[2]} g(\sum_{i=1}^{D}(x_i w_{i1}^{[1]}) + b_1^{[1]}) + w_2^{[2]} g(\sum_{i=1}^{D}(x_i w_{i2}^{[1]}) + b_2^{[1]}) + b_1^{[2]} < 0, \; \sum_{i=1}^{D} x_i \neq K \,. \end{cases} \tag{8}$$

Note that the conditions depend on $\sum_{i=1}^{D} x_i$. This way, it should be helpful to compute that in the hidden layers. It can be achieved by having $\sum_{i=1}^{D}(x_i w_{i1}^{[1]}) = \sum_{i=1}^{D}(x_i w_{i2}^{[1]}) = \sum_{i=1}^{D} x_i$, which is true for $w_{i1}^{[1]} = w_{i2}^{[1]} = 1$, $i \in \{1, ..., D\}$. However, since we also need the network to be resistant to infinitesimal perturbations, let's instead compute $n \sum_{i=1}^{D} x_i$, $n \in \mathbb{Z}$ (to be explained further ahead), by having $w_{i1}^{[1]} = w_{i2}^{[1]} = n$. From (8), denoting $S = \sum_{i=1}^{D} x_i$, we reduced the problem to the following:

$$\begin{cases} w_1^{[2]} g(nS + b_1^{[1]}) + w_2^{[2]} g(nS + b_2^{[1]}) + b_1^{[2]} \geq 0, \; S = K \,, \\ w_1^{[2]} g(nS + b_1^{[1]}) + w_2^{[2]} g(nS + b_2^{[1]}) + b_1^{[2]} < 0, \; S \neq K \,. \end{cases} \tag{9}$$

There are three cases for S from the equations above, $S = K$, $S > K$, and $S < K$. Consider K_1, K_2 such that $K_1 < K < K_2$. Having two hidden units, we need different behaviors for the different situations. One possibility is to activate only one unit when $S = K$, activate no units when $S = K_1$, and activate both when $S = K_2$.

This way, we need to define $b^{[1]}$ such that those activation patterns hold. Assuming that an heavyside neuron is activated if it outputs 1 and deactivated otherwise, the proposed activations can be expressed as a system of inequalities over K_1, K, and K_2:

$$\begin{cases} nK + b_1^{[1]} \geq 0 \\ nK + b_2^{[1]} < 0 \\ nK_1 + b_1^{[1]} < 0 \\ nK_1 + b_2^{[1]} < 0 \\ nK_2 + b_1^{[1]} \geq 0 \\ nK_2 + b_2^{[1]} \geq 0 \end{cases} . \tag{10}$$

For $n = 1$, the system reduces to $-K \leq b_1^{[1]} \leq -K_1 \wedge -K_2 \leq -b_2^{[1]} < K$, which has an integer solution for instance for $b_1^{[1]} = -K, b_2^{[1]} = -K - 1$. However, this would result in a network that is not robust to infinitesimal perturbations, since the left-hand-side of the inequalities may evaluate to 0, which is the discontinuity point of the heavyside function. Consider a perturbation $\epsilon \in \mathbb{R}^D$, where $\sum_{i=1}^{D} \epsilon_i < 0$. It would make $K' = \sum_{i=1}^{D} x_i + \sum_{i=1}^{D} \epsilon_i < \sum_{i=1}^{D} x_i = K$. For instance, the inequality 10.1 would evaluate to $K' - K \geq 0$ which does not hold, i.e., the output of the neuron would change.

In order to overcome this issue, we can have $n = 2$ and restrict the inequalities to be strictly lower or greater than 0:

$$\begin{cases} nK + b_1^{[1]} > 0 \\ nK + b_2^{[1]} < 0 \\ nK_1 + b_1^{[1]} < 0 \\ nK_1 + b_2^{[1]} < 0 \\ nK_2 + b_1^{[1]} > 0 \\ nK_2 + b_2^{[1]} > 0 \end{cases} . \tag{11}$$

One possible integer solution is $b_1^{[1]} = -2K + 1, b_2^{[1]} = -2K - 1$. This way, the absolute value of the left-hand-side of the inequalities always evaluates to at least 1, and no infinitesimal perturbation $\epsilon \in \mathbb{R}^D$ will change the output of the neurons.

Finally, since we found a solution for $b^{[1]}$ that makes the activation pattern possible, we need to find $w_1^{[2]}$, $w_2^{[2]}$, and $b_1^{[2]}$ that, given the activations, produce the correct output. From (9), this yields the following system of inequalities:

$$\begin{cases} w_1^{[2]} + b_1^{[2]} \geq 0 \\ b_1^{[2]} < 0 \\ w_1^{[2]} + w_2^{[2]} + b_1^{[2]} < 0 \end{cases} . \tag{12}$$

One possible solution over the integers is $w_1^{[2]} = 1$, $w_2^{[2]} = -1$, $b_1^{[2]} = -1$. Note that this system does not depend directly on the value of K, since it depends only on the output of the hidden layer activations (which are 0 or 1, in this case). Hence, if the output of the hidden layer activations is resistant to infinitesimal perturbations, so will the final output be.

Summarily, we have $w_{i1}^{[1]} = w_{i2}^{[1]} = 2$, $i \in \{1, ..., D\}$, $b_1^{[1]} = -2K + 1, b_2^{[1]} = -2K - 1$, $w_1^{[2]} = 1$, $w_2^{[2]} = -1$, and $b_1^{[2]} = -1$ as a set of weights and biases for the required MLP structure that computes f and is robust to infinitesimal perturbations.

1c

Assuming the output layer uses the Heavyside step function as activation, the system of inequalities 8 still holds: the input to the activation has to be positive or zero if $\sum_{i=1}^{D} x_i = K$, and negative otherwise. However, the hidden layer activation function g is now ReLU, given by:

$$g_{\text{ReLU}}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} . \tag{13}$$

Note that this function shares a discontinuity point with the heavyside function at $x = 0$. We can resort to the previous defined systems of inequalities (12) and (11) to compute the values for

the weights and biases. However, the system of inequalities (12) needs to change, since an activated neuron with g_{ReLU} will output the input, whereas with heavyside it would output 1:

$$\begin{cases} (nK + b_1^{[1]})w_1^{[2]} + b_1^{[2]} \geq 0 \\ b_1^{[2]} < 0 \\ (nK_2 + b_1^{[1]})w_1^{[2]} + (nK_2 + b_2^{[1]})w_2^{[2]} + b_1^{[2]} < 0 \end{cases} . \tag{14}$$

The values $n = 2$, $b_1^{[1]} = -2K + 1$, and $b_2^{[1]} = -2K - 1$ were already computed as a solution for inequality (11) that is robust to infinitesimal perturbations when the activation function is the Heavyside step function. Since the discontinuity point of the ReLU function if also $x = 0$, the found solution is also robust to infinitesimal perturbations in this case for the hidden layers. To check if there is a solution for $w_1^{[2]}$, $w_2^{[2]}$, and $b_1^{[2]}$ using those values, we first solve the inequality system. Since $K_2 > K$, we can consider $K_2 - K = k \geq 1$ (since, by definition, $K = \sum_{i=1}^{D} x_i$ and $x \in \{0, 1\}^D$). Moreover, let's restrict the first inequality to be strictly positive, in order to avoid the discontinuity point. We can write:

$$\begin{cases} (2K - 2K + 1)w_1^{[2]} + b_1^{[2]} > 0 \\ b_1^{[2]} < 0 \\ (2K_2 - 2K + 1)w_1^{[2]} + (2K_2 - 2K - 1)w_2^{[2]} + b_1^{[2]} < 0 \end{cases} \tag{15}$$

$$\Leftrightarrow \begin{cases} w_1^{[2]} + b_1^{[2]} > 0 \\ b_1^{[2]} < 0 \\ (2k + 1)w_1^{[2]} + (2k - 1)w_2^{[2]} + b_1^{[2]} < 0 \\ k \geq 1 \end{cases} \tag{16}$$

$$\Leftrightarrow \begin{cases} -w_1^{[2]} < b_1^{[2]} < 0 \\ k \geq 1 \\ w_2^{[2]} \leq \frac{2kw_1^{[2]} + w_1^{[2]}}{1-2k} \end{cases} \tag{17}$$

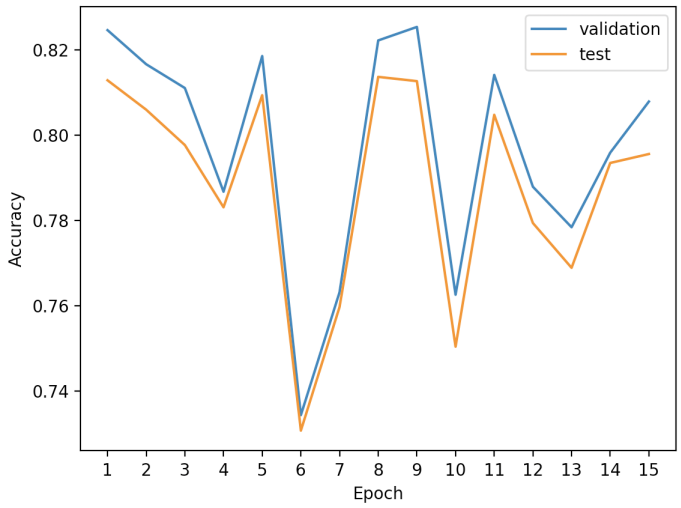
One possible solution for $w_1^{[2]}$ and $b_1^{[2]}$ is $w_1^{[2]} = 3$ and $b_1^{[2]} = -2$. This gives us that $w_2^{[2]} \leq \frac{6k+3}{1-2k}$. The dependency on k needs further analysis. The value k is an integer ranging from 1 to D. The function $f(k) = \frac{6k+3}{1-2k}$ is increasing for positive integers (since $f'(k) = \frac{12}{(1-2k)^2} > 0$, $\forall k \neq \frac{1}{2}$). Hence, if $w_2^{[2]} \leq \frac{6k+3}{1-2k}$ for $k = 1$, then $w_2^{[2]} \leq \frac{6k+3}{1-2k}$ for $k > 1$. As such, a possible solution is $w_2^{[2]} = \frac{6+3}{1-2} =$ -9. Again, note that the found solution evaluates the absolute value of the left-hand-side of the inequalities in (14) to at least one, hence no infinitesimal perturbation $\epsilon \in \mathbb{R}^D$ will change the output of the neurons.

Summarily, we have $w_{i1}^{[1]} = w_{i2}^{[1]} = 2$, $i \in \{1, ..., D\}$, $b_1^{[1]} = -2K + 1, b_2^{[1]} = -2K - 1$, $w_1^{[2]} = 3$, $w_2^{[2]} = -9$, $b_1^{[2]} = -2$ as a set of weights and biases for the required MLP structure that computes f and is robust to infinitesimal perturbations.

Question 2

1a

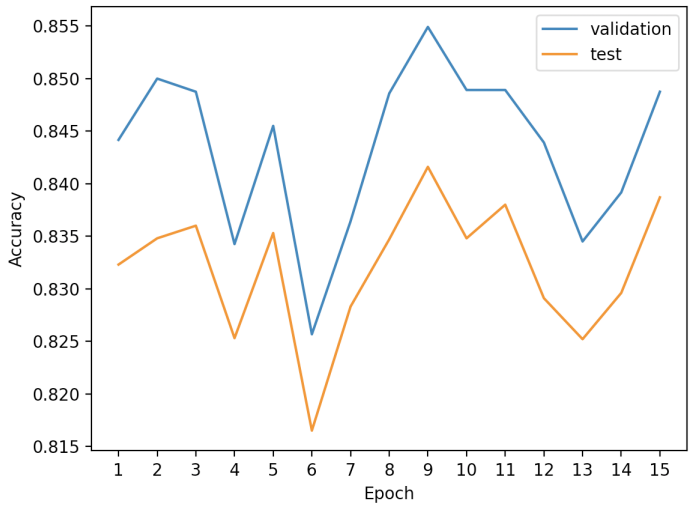
A perceptron trained with learning rate of 1 and no l2 penalty achieved the following validation and test accuracies over 15 epochs:



At the 15^{th} epoch, the model had a validation accuracy of 0.8079 and a test accuracy of 0.7956.

1b

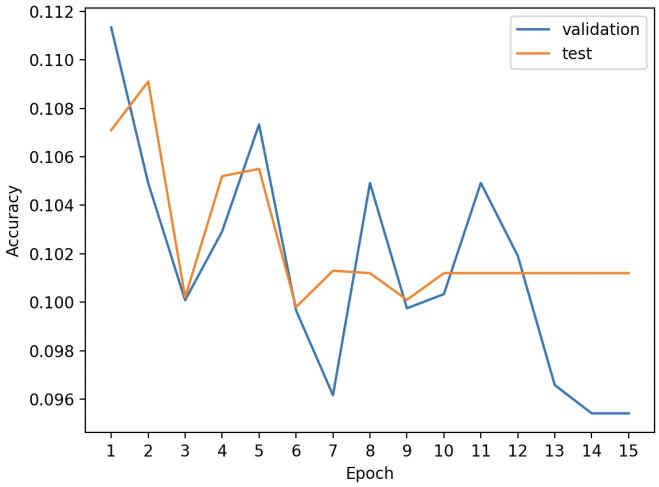
A logistic regression model trained with learning rate of 0.001 and no l2 penalty achieved the following validation and test accuracies over 15 epochs:



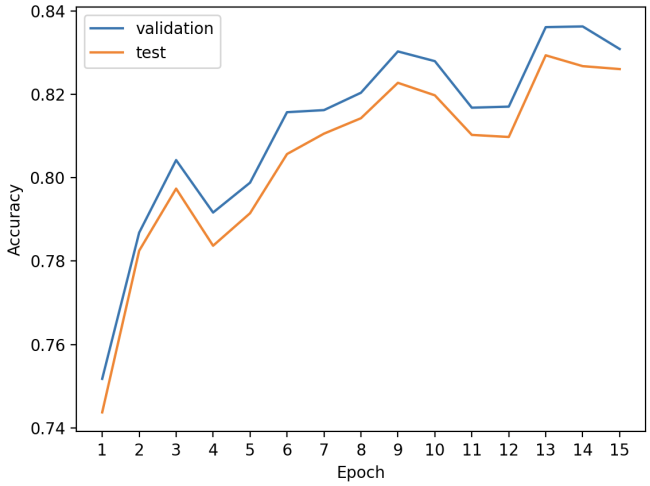
At the 15^{th} epoch, the model had a validation accuracy of 0.8488 and a test accuracy of 0.8387.

2a

An MLP was implemented as described, using a single hidden layer with 100 units. The weights were initialized using an uniform(-1, 1.01) distribution, and biases were initialized to 0. The following plot shows the validation and test accuracies per epoch, for learning rate of 0.1:



The following plot shows the validation and test accuracies per epoch, for learning rate of 0.001:



In this case, a learning rate of 0.001 is much better, achieving a final test accuracy of 0.8261.

2b

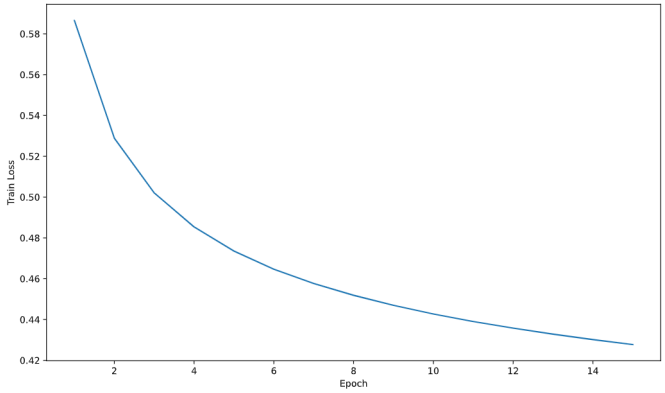
In this model (MLP), the input is mapped to the space defined by the hidden layer(s), which can in fact be seen as learning representations that are then mapped to the output space. The previous models (perceptron, logistic regression) map the input directly to the output space, hence no features are being learned. Also note the importance of non-linear activations: an MLP with only linear activations can be written as a regular perceptron. Hence, the non-linear activation functions play an important role in representation learning, by allowing the model to learn complex patterns.

Question 3

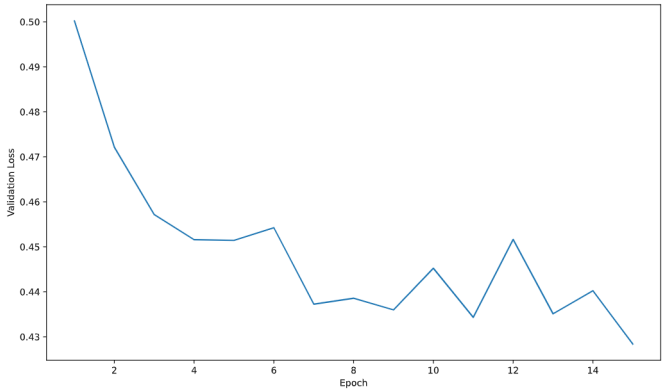
1

A logistic regression model was implemented as described. The best model was achieved using a learning rate of 0.0005, with a validation accuracy of 0.8562.

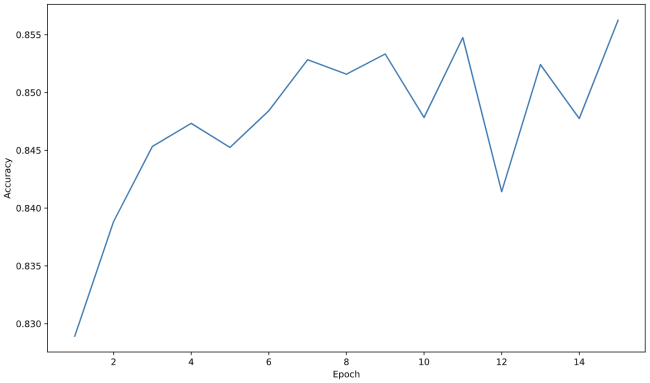
The following plot shows train loss per epoch:



The following plot shows validation loss per epoch:



The following plot shows validation accuracy per epoch:

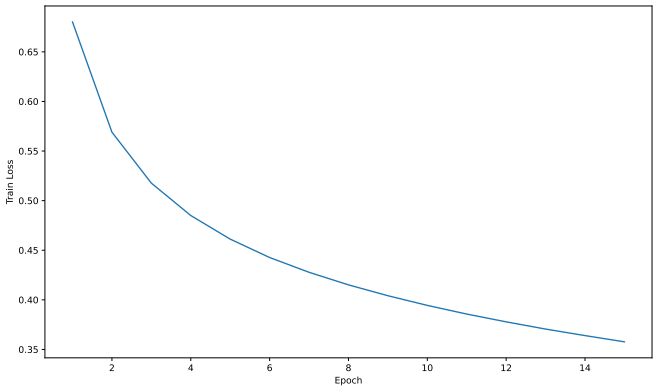


The model achieves a test accuracy of 0.8420, and a macro-averaged F1-score of 0.8415.

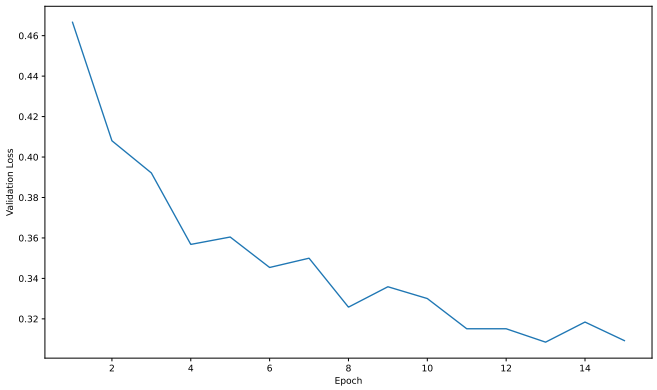
2

An MLP was implemented as described. The best setup had a learning rate 0.001, 2 hidden layers with 200 and 130 units, respectively, and 0.2 dropout, achieving a validation accuracy of 0.8902.

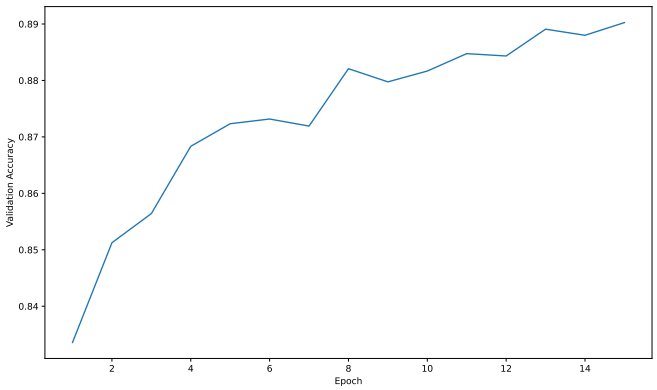
The following plot shows train loss per epoch:



The following plot shows validation loss per epoch:



The following plot shows validation accuracy per epoch:

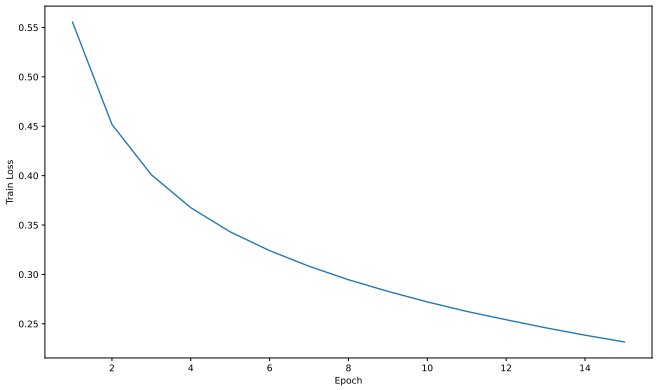


The model achieves a test accuracy of 0.8825.

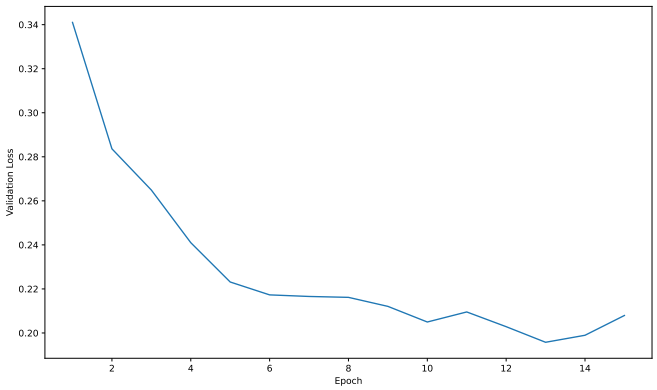
3a

A CNN was implemented as described. The best setup had a learning rate 0.01, 0.5 dropout, kernel size of 3, and the two convolutions had 16 and 32 filters, respectively. A batch size of 4 was used. This setup achieved a validation accuracy of 0.9293.

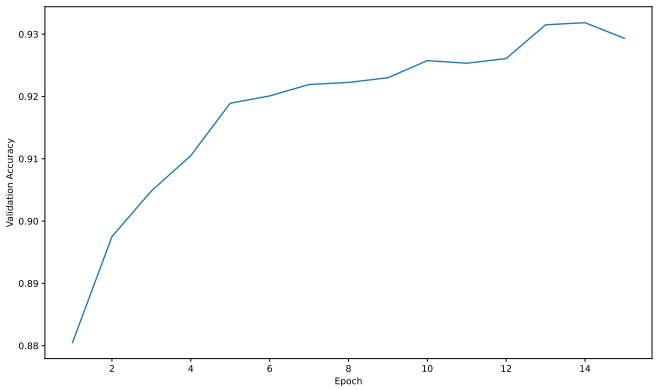
The following plot shows train loss per epoch:



The following plot shows validation loss per epoch:



The following plot shows validation accuracy per epoch:



The model achieves a test accuracy of 0.9295.

3b

In the previous question, the MLP, feature representations are learned by computing a weighted sum of all input features in a fully connected layer.

CNNs learn feature representations by applying convolution between filters and spatial regions of the input, which introduces sparsity to the connections. Parameter tying is also considered by having the filters share weights. Furthermore, in this exercise, max pooling was applied to reduce the dimensions of the resulting features.

The main advantages of using a CNN over a FFNN come from the ability of learning spatial patterns and features. Hence, CNNs are particularly well-suited for tasks concerning images (e.g, image classification).