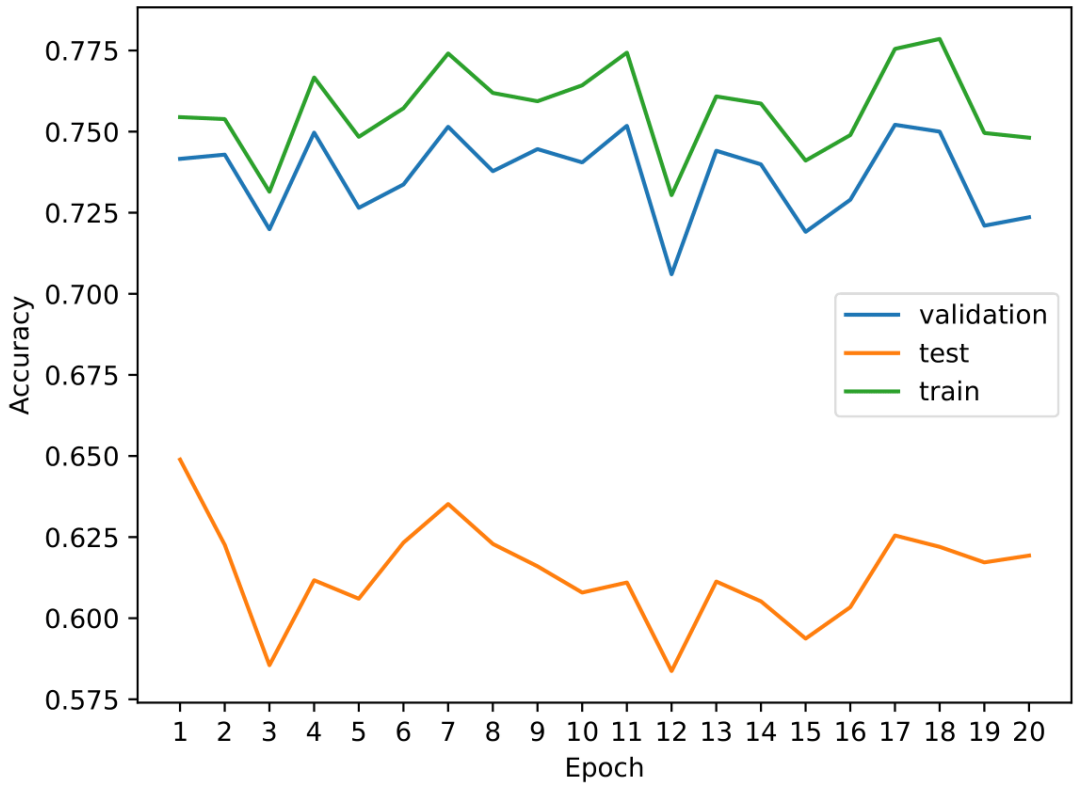
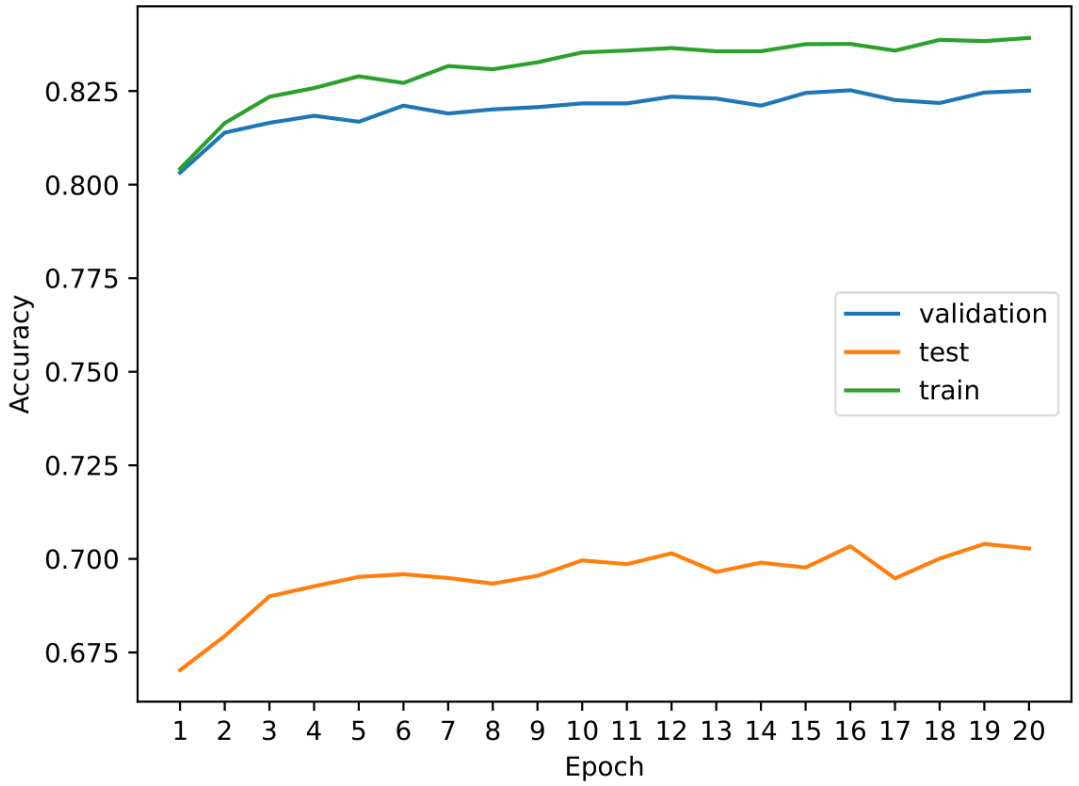
Question 1

1a) Final validation accuracy: 0.7236
Final test accuracy: 0.6193



1b) Final validation accuracy: 0.8251
Final test accuracy: 0.7028

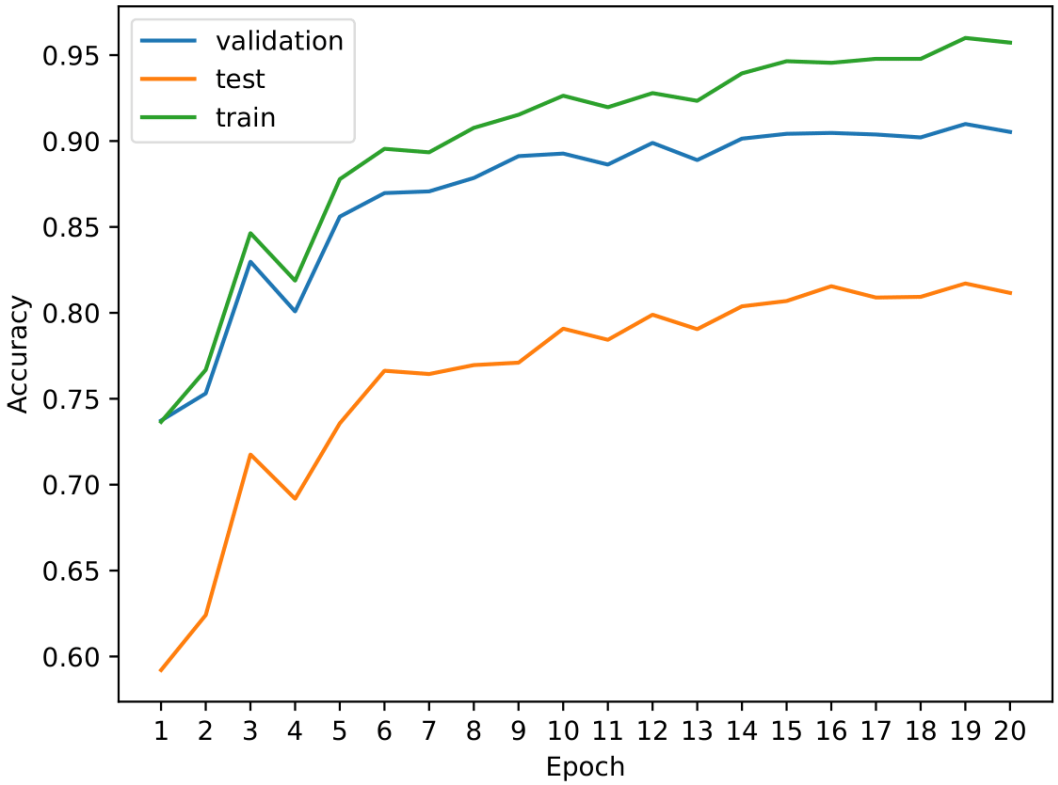


2a) The use of non-linear activations enables the MLP to capture more complex relationships between the inputs and outputs. If the problem isn't linearly separable, a perceptron is unable to solve it, however an MLP with non-linear activation function is capable of learning a more complex decision boundary, which also depends on the number of hidden units, as stated by the Universal Approximation Theorem.

If the activation function is linear, the combination of the various layers can be re-written as being equivalent to a single layer with a linear activation function.
For example, with an activation function g(x) = x, and with 2 layers, you could rewrite the output $y = g(z(g(z(x))))$ as
$g(z(g(W_1x + b_1))) = g(z(W_1x + b_1)) = g(W_2W_1x + W_2b_1 + b_2) = W_2W_1x + W_2b_1 + b_2$

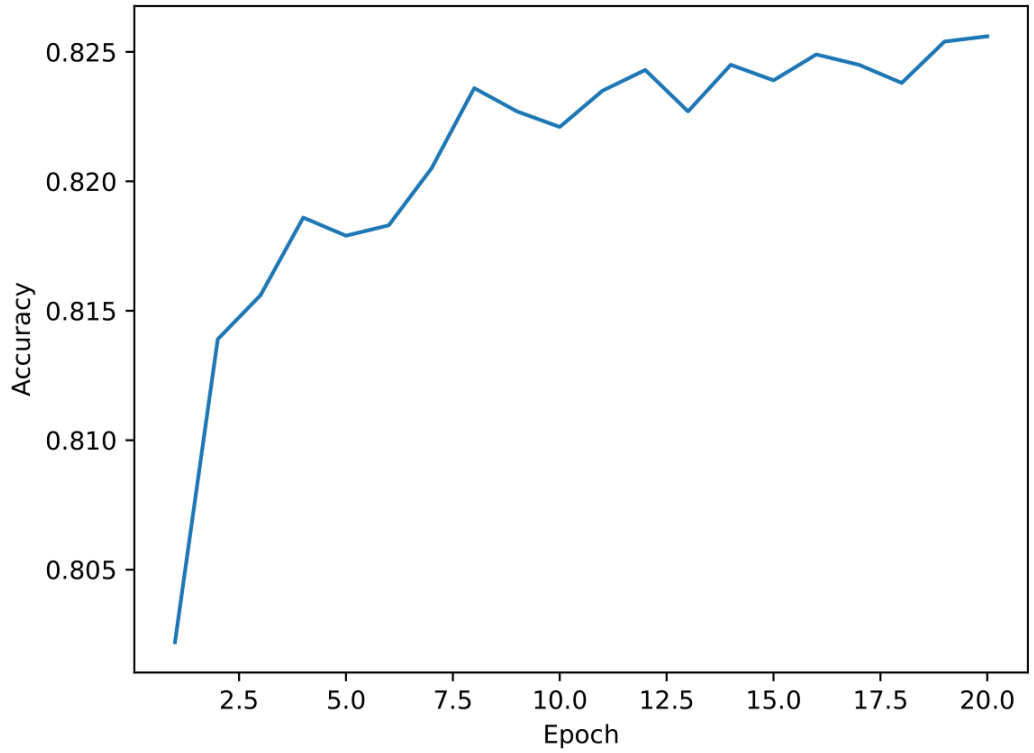
In this particular task the use of MLP instead of the simple perceptron helps us to capture a better representation of the pixels' information.

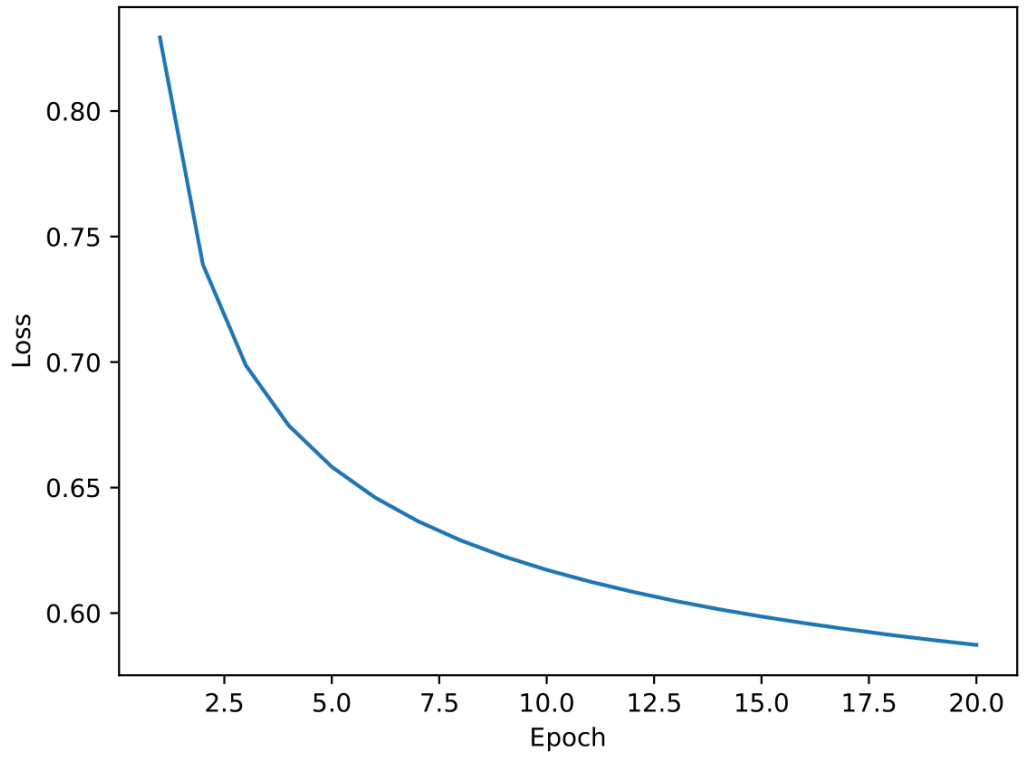
2b) Final validation accuracy: 0.9053
Final test accuracy: 0.8116



Question 2

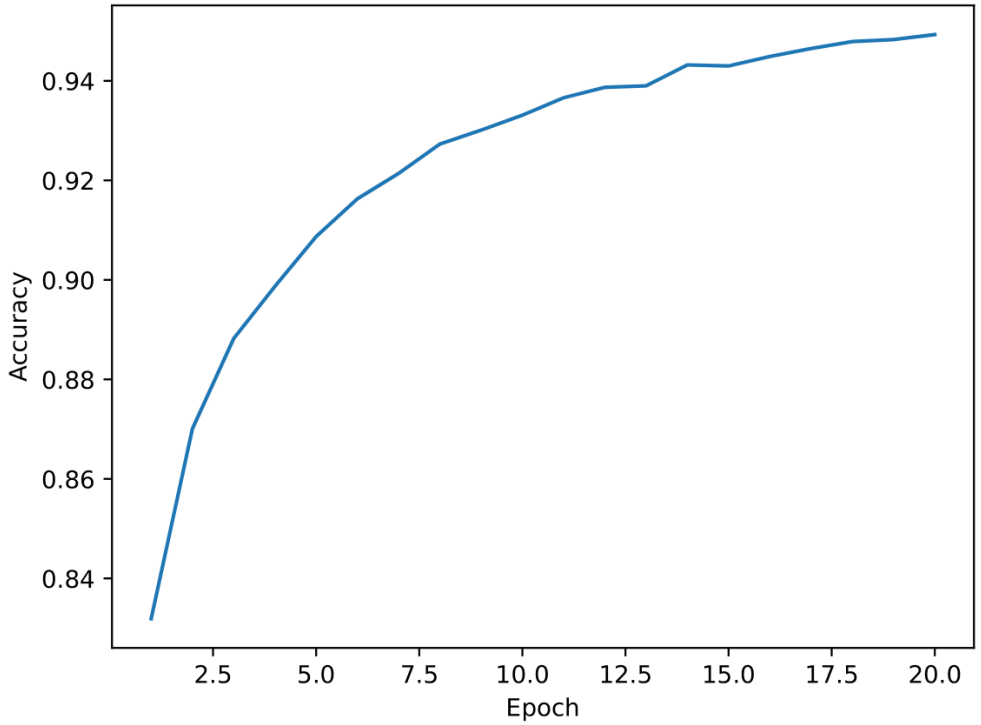
1) Final validation accuracy: 0.8256
Final test accuracy: 0.7019
Best configuration: 0.001 learning rate

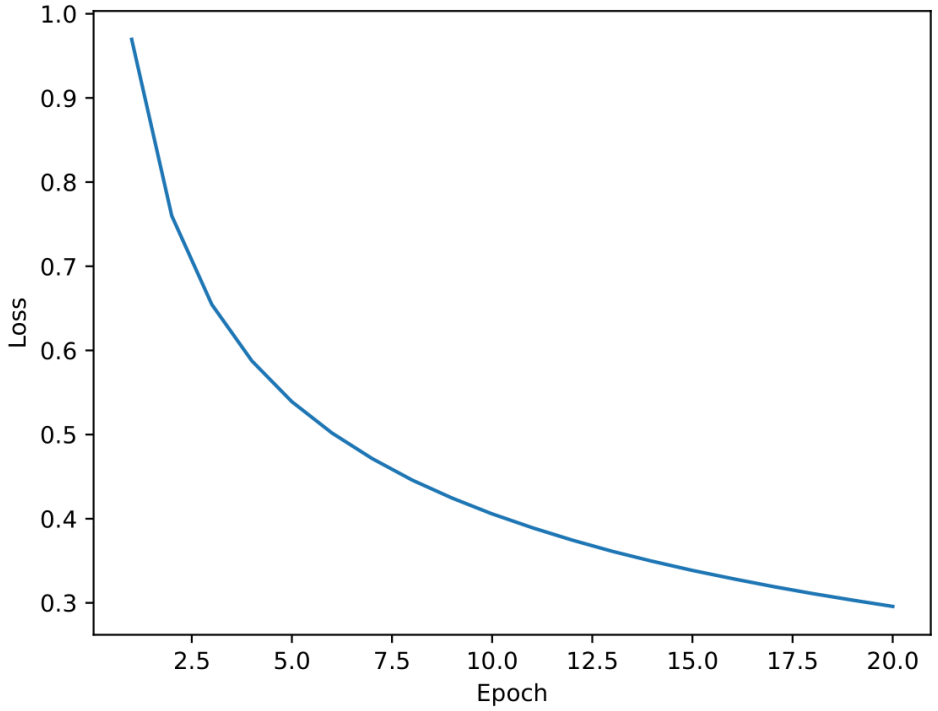




2) Final validation accuracy: 0.9493
Final test accuracy: 0.8814
Best configuration:

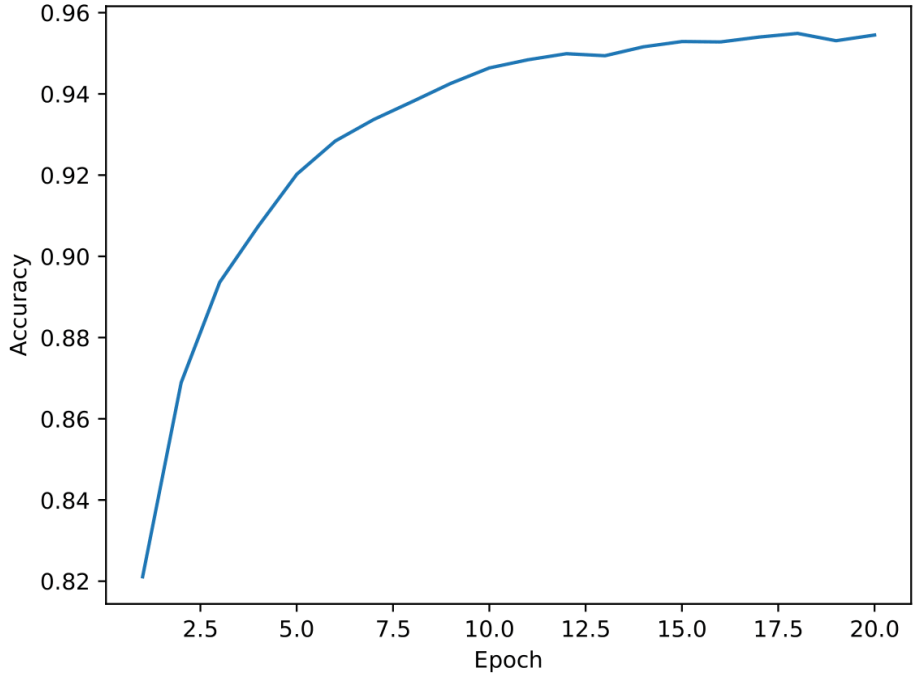
- 0.01 learning rate
- 200 hidden units per layer
- 1 layer
- 0.3 Dropout
- ReLU activation
- SGD optimizer

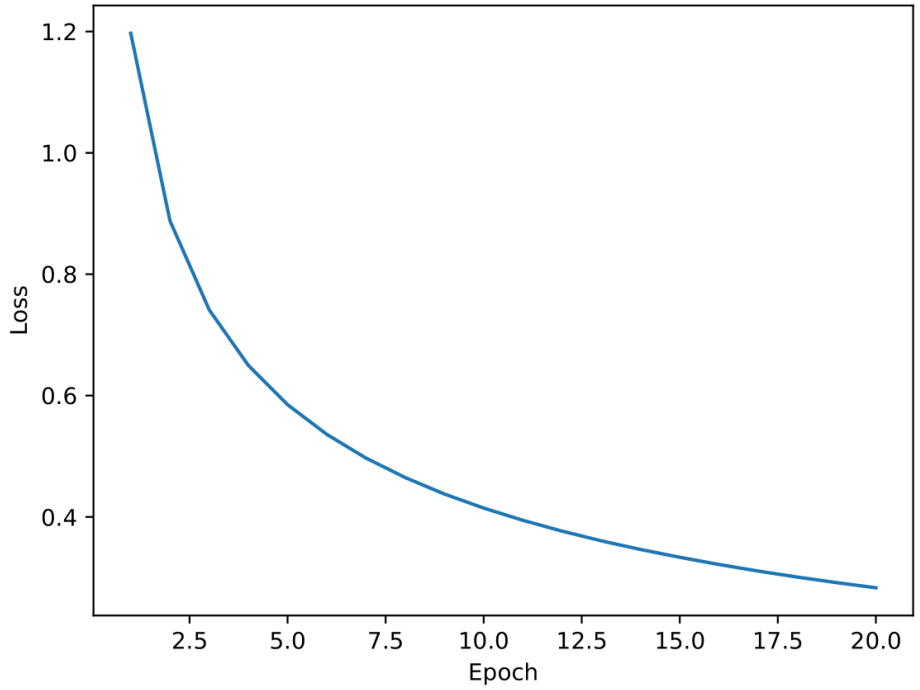




3) Final validation accuracy: 0.9545
Final test accuracy: 0.8949
Best configuration:

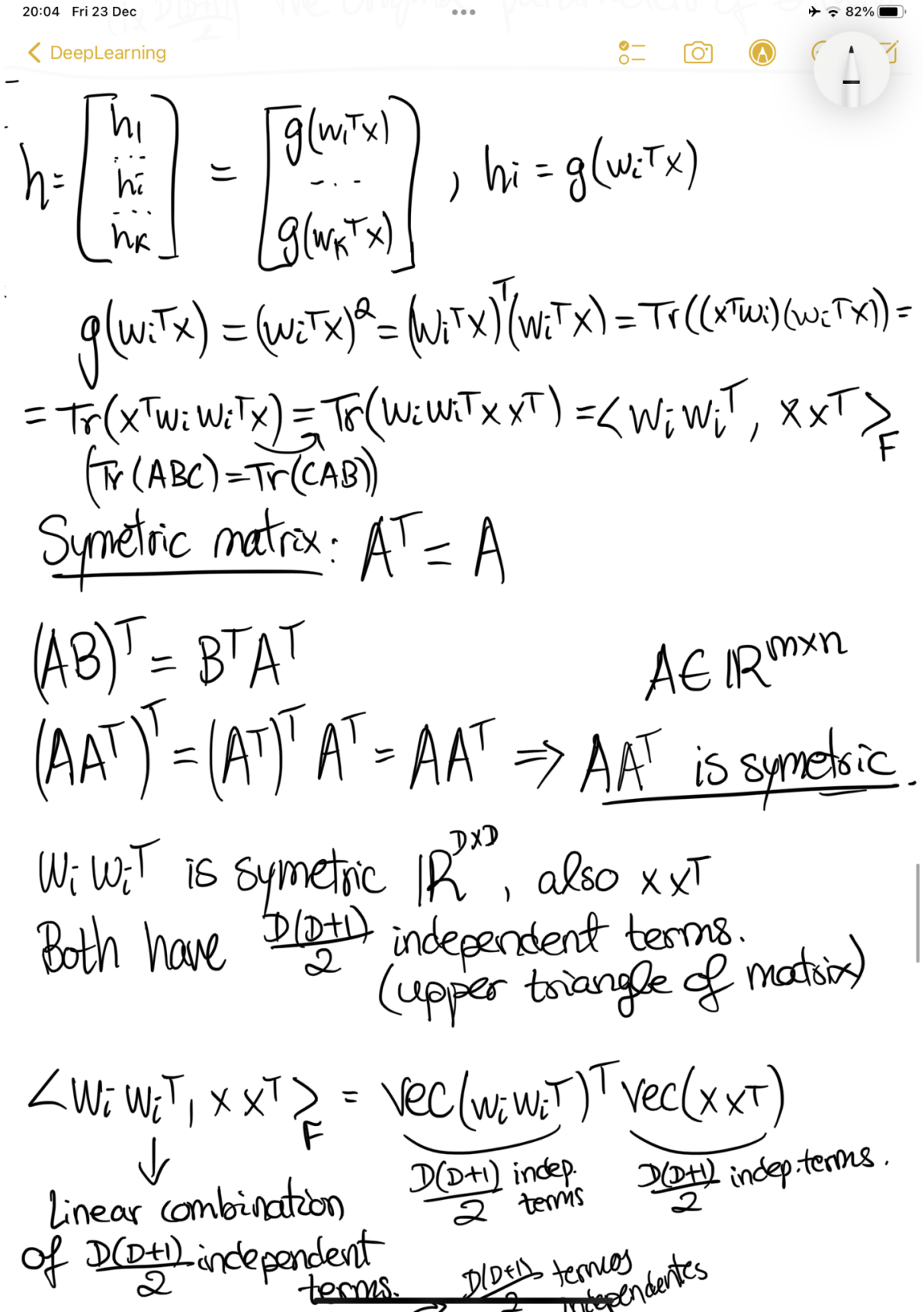
- 0.01 learning rate
- 200 hidden units per layer
- 2 layers
- 0.3 Dropout
- ReLU activation
- SGD optimizer





Question 3

1)


20:04 Fri 23 Dec
82%
DeepLearning
$h = \begin{bmatrix} h_1 \\ \dots \\ h_i \\ \dots \\ h_K \end{bmatrix} = \begin{bmatrix} g(w_1^T x) \\ \dots \\ g(w_K^T x) \end{bmatrix}$, $h_i = g(w_i^T x)$
$g(w_i^T x) = (w_i^T x)^2 = (w_i^T x)^T (w_i^T x) = Tr((x^T w_i)(w_i^T x)) =$
$= Tr(x^T w_i w_i^T x) = Tr(w_i w_i^T x x^T) = \langle w_i w_i^T, x x^T \rangle_F$
$(Tr(ABC) = Tr(CAB))$
Symetric matrix: $A^T = A$
$(AB)^T = B^T A^T$
$A \in \mathbb{R}^{m \times n}$
$(AA^T)^T = (A^T)^T A^T = AA^T \Rightarrow AA^T$ is symetric.
$w_i w_i^T$ is symetric $\mathbb{R}^{D \times D}$, also $x x^T$
Both have $\frac{D(D+1)}{2}$ independent terms.
(upper triangle of matrix)
$\langle w_i w_i^T, x x^T \rangle_F = vec(w_i w_i^T)^T vec(x x^T)$
$\frac{D(D+1)}{2}$ indep. terms
$\frac{D(D+1)}{2}$ indep. terms.
Linear combination of $\frac{D(D+1)}{2}$ independent terms.
$\frac{D(D+1)}{2}$ terms independentes

terms. $\frac{D(D+1)}{2}$ independent

$$h = \begin{bmatrix} h_1 \\ \dots \\ h_i \\ \dots \\ h_K \end{bmatrix} = \begin{bmatrix} \text{vec}(w_1 w_1^T)\,\text{vec}(xx^T) \\ \dots \\ \text{vec}(w_i w_i^T)\,\text{vec}(xx^T) \\ \dots \\ \text{vec}(w_K w_K^T)\,\text{vec}(xx^T) \end{bmatrix} = \begin{bmatrix} a_1^T\theta \\ \dots \\ a_i^T\theta \\ \dots \\ a_K^T\theta \end{bmatrix} \phi(x)$$

$(K \times 1)$ $\quad (K \times \frac{D(D+1)}{2})$ $\quad (\frac{D(D+1)}{2} \times 1)$

Each row h_i in vector h is a sum of the $(D \times D)$ $\frac{D(D+1)}{2}$ independent terms resulting from $\text{vec}(ww^T)^T \text{vec}(xx^T)$.

Therefore we can write h as $A_\theta \phi(x)$ where (symetric) $A_\theta = \begin{bmatrix} a_1^T\theta \\ a_i^T\theta \\ a_K^T\theta \end{bmatrix}$ and each $a_i\theta$ is a vector $(\frac{D(D+1)}{2} \times 1)$ containing all the $\frac{D(D+1)}{2}$ independent terms of $\text{vec}(w_i w_i^T)$ multiplied by their frequency on the $\text{vec}(w_i w_i^T)$, $a_i\theta = [w_1^2 \; w_2^2 \dots w_3^2 \dots (2)w_{K-1}w_K]$.

That means: $a_{i\theta} \in \mathbb{R}^{\frac{D(D+1)}{2}}$, $i \in \{1, \dots, K\}$

$\hookrightarrow A_\theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$.

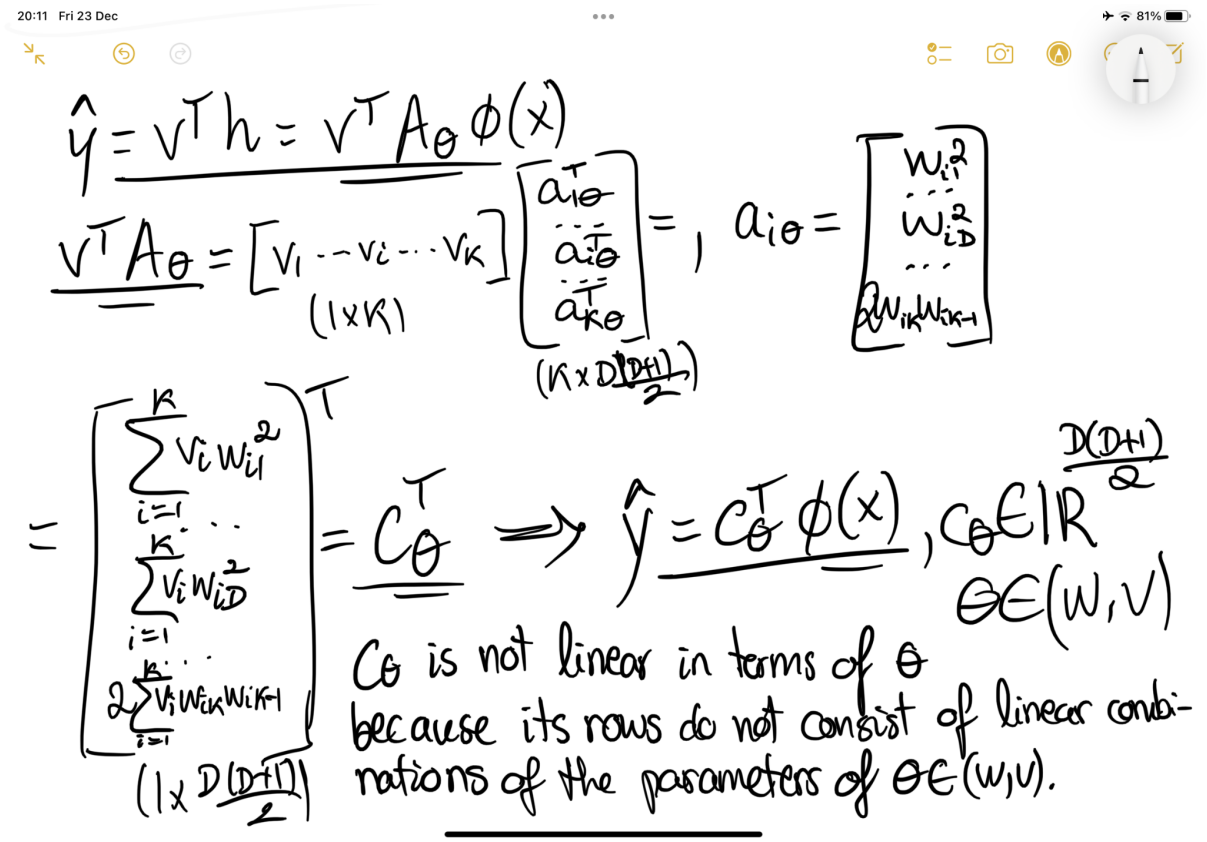
$w_{K-1}w_K = w_K w_{K-1}$ (2x)

By the same logic $\phi(x) \in \mathbb{R}^{\frac{D(D+1)}{2} \times 1}$, $\phi(x) = \begin{bmatrix} \phi_1(x) \\ \dots \\ \phi_i(x) \\ \dots \\ \phi_{\frac{D(D+1)}{2}}(x) \end{bmatrix}$

$\phi_i(x) \in \mathbb{R}$ and corresponds to the i^{th} term of the $\frac{D(D+1)}{2}$ independent terms in $\text{vec}(xx^T)$, where each repeated term is multiplied by 2, as in the case of $a_{i\theta}$.

$$h = A_\theta \phi(x), \quad A_\theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}, \quad \phi(x) \in \mathbb{R}^{\frac{D(D+1)}{2}}$$

2)


20:11 Fri 23 Dec
81%
$\hat{y} = V^T h = V^T A_\theta \phi(x)$
$V^T A_\theta = [V_1 \cdots V_i \cdots V_K]$ $(1 \times K)$
$\begin{bmatrix} a_{1\theta}^T \\ \vdots \\ a_{i\theta}^T \\ \vdots \\ a_{K\theta}^T \end{bmatrix} = ,$ $(K \times D\frac{(D+1)}{2})$
$a_{i\theta} = \begin{bmatrix} w_{i1}^2 \\ \vdots \\ w_{iD}^2 \\ \vdots \\ 2w_{ik}w_{ik-1} \end{bmatrix}$
$= \begin{bmatrix} \sum_{i=1}^{K} V_i w_{i1}^2 \\ \vdots \\ \sum_{i=1}^{K} V_i w_{iD}^2 \\ \vdots \\ 2\sum_{i=1}^{K} V_i w_{ik} w_{ik-1} \end{bmatrix}^T = C_\theta^T \Rightarrow \hat{y} = C_\theta^T \phi(x)$, $C_\theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$
$(1 \times D\frac{(D+1)}{2})$
$\theta \in (W, V)$
C_θ is not linear in terms of θ
because its rows do not consist of linear combinations of the parameters of $\theta \in (W, V)$.

3)
Our neural network structure consists of a single hidden layer which receives $x \in R^D$ as input and has a quadratic activation function $g(z) = z^2$; and an output layer which receives the internal representation of the network (h).

Given model parameters $\Theta = (W, v) \in R^{K \times D} \times R^K$, the matrix $W \in R^{K \times D}$ represents the weights between the input vectors (of Size D) and the hidden layer (of size K), and the vector $v \in R^K$ represents the weights between the hidden layer and the output layer.

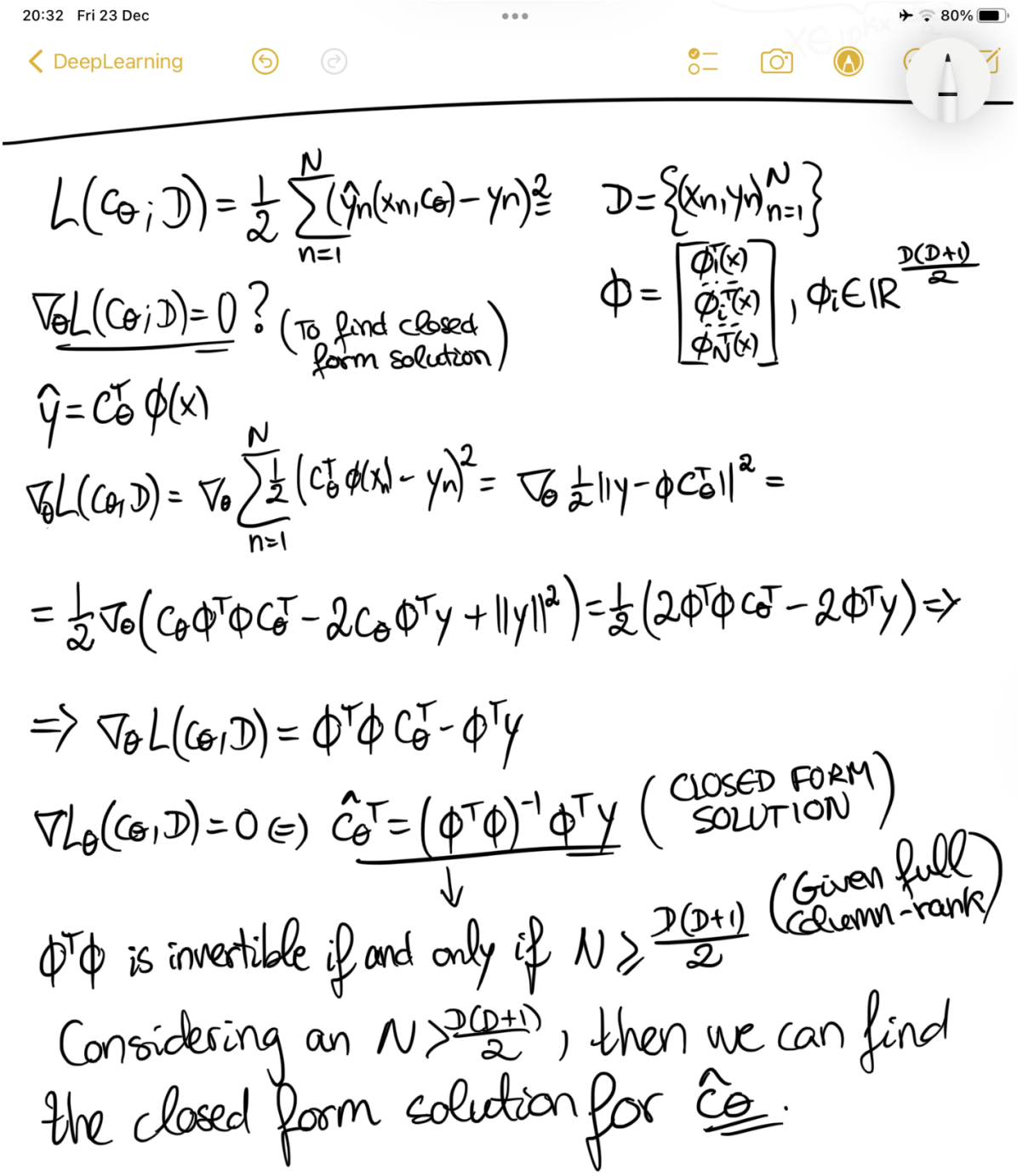
Our model receives the internal representation of the network as input and outputs a prediction, given weights v: $\hat{y} \in R$ is given by $\hat{y} = vTh$, where $h \in R^K$, $h = g(Wx)$. Since $g(z) = z^2$, this means that h is a vector of quadratic functions of the input variables $x \in R^D$.

The feature transformation $\varphi : R^D \rightarrow R^{D(D+1)/2}$ takes $x \in R^D$ as input and maps it to a higher-dimensional space, represented by the vector $\varphi(x) \in R^{D(D+1)/2}$.

We also proved we can write the predicted output $\hat{y}$ as $\hat{y}(x; c\Theta) = cT\Theta\varphi(x)$, where $c\Theta \in R^{D(D+1)/2}$.

Considering $K >= D$, the number of hidden units K is greater than or equal to the number of input variables D, so there must exist a $c \in R^{D(D+1)/2}$ such that $c = c\Theta$ because there are enough hidden units to represent the features of each input variable and their relations in the higher-dimensional space represented by the feature transformation $\varphi(x)$.
This does not mean that it is a linear model in terms of $c\Theta$ given.
On the other side if $K < D$, the number of hidden units K is smaller than the number of input variables, and as such there might not be a choice of the original parameters such that $c = c\Theta$ for $c \in R^{D(D+1)/2}$.

4)



In addition to that, all features must NOT be linear combinations of others, and that is guaranteed, since X has full column-rank.