Solutions to Simon J.D. Prince's Understanding Deep Learning

Solutions to Simon J.D. Prince's Understanding Deep Learning

by

Asad Nizami

Preface

This book contains my solutions to the exercises/problems presented in the book. These solutions are entirely my own work and have not been reviewed or verified by others. While I have made every effort to ensure accuracy, there may be errors or misunderstandings. Readers are encouraged to verify the solutions independently and use them as a reference only.

Contents

Chapter I

Introduction

Notebook Problems

1. Notebook 1.1 – Background Mathematics

Chapter II

Supervised learning

Notebook Problems

1. Notebook 2.1 – Supervised Learning

Text Book Problems

Problem 2.1

$$L = \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i)^2$$

$$\frac{\partial L}{\partial \phi_0} = 2 \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i) * \frac{\partial}{\phi_0} (\phi_0 + \phi_1 x_1 - y_i)$$

$$= 2 \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i)$$

$$\frac{\partial L}{\partial \phi_1} = 2 \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i) * x_i$$

Problem 2.2

We can minimize a function by setting its derivative to zero. For the above equations, we can directly obtain the optimal value of ϕ_0 and ϕ_1.

$$\frac{\partial L}{\partial \phi_0} = 2 \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i) = 0$$

Expanding the summation:

$$I * \phi_0 + \phi_1 * \sum_{i=1}^{I} x_i - \sum_{i=1}^{I} y_i = 0 \tag{1}$$

Similarly, for the other equation:

$$\frac{\partial L}{\partial \phi_1} = 2 \sum_{i=1}^{I} (\phi_0 + \phi_1 x_1 - y_i) * x_i = 0$$

$$\phi_0 * \sum_{i=1}^{I} x_i + \phi_1 * \sum_{i=1}^{I} x_i^2 - \sum_{i=1}^{I} y_i x_i = 0 \tag{2}$$

Now we can solve for ϕ_0 and ϕ_1 using equations (1) and (2) without iterative optimization.

Problem 2.3

Reformulating the equation of linear regression, we get $x = \phi_0 + \phi_1 y$. This is the inverse of a typical discriminative equation and here we shift the focus to predicting the input (x) for a given output (y). Rearranging the equation, we get $y = \frac{x - \phi_0}{\phi_1}$.

The new loss function becomes:

$$L = \sum_{i=1}^{I} (\phi_0 + \phi_1 y_i - x_i)^2$$

Fitting a line for both the models (discriminative and generative), we see the models do not overlap completely and will give slightly different results as shown in the supplementary notebook.

Chapter III

Shallow neural networks

Notebook Problems

1. Notebook 3.1 – Shallow neural networks I
2. Notebook 3.2 – Shallow neural networks II
3. Notebook 3.3 – Shallow network regions
4. Notebook 3.4 – Activation functions

Text Book Problems

Problem 3.1
We have the equation of linear regression (removed additional parameters for simplicity) $y = \phi_0 + \phi_1 * a[\theta_0 + \theta_1 x]$. Let's assume activation function $a[z] = \psi_0 + \psi_1 z$. On substituting the activation function and expanding the equation, we get $y = \phi_0 + \phi_1\psi_0 + \phi_1\psi_1\theta_0 + \phi_1\psi_1\theta_1 x$, which is linear as well. By removing the activation function, the equation continues to be linear- $y = \phi_0 + \phi_1(\theta_0 + \theta_1 x)$.

Problem 3.2
The hidden units that are active in the figure are- h_1 and h_2.

Problem 3.3
Joints are present in the function for the value of x where the preactivation function

is zero, ie $\theta_0 + \theta_1 x = 0$, so $x = -\theta/\theta_1$. The slopes of each part of the function depend on which activation functions.

Problem 3.5

$$\text{ReLU}[\alpha \cdot z] = \begin{cases} 0 & \text{if } \alpha \cdot z < 0, \\ \alpha \cdot z & \text{if } \alpha \cdot z \geq 0. \end{cases}$$

For $\alpha \cdot$ ReLU, we have two cases- if $z < 0$, the output is 0 and $\alpha \cdot z$ for the other case. Therefore, in both cases, the equality $\text{ReLU}[\alpha \cdot z] = \alpha \cdot \text{ReLU}[z]$ holds true.

Problem 3.6
We have $h_1 = a[\theta_{10} + \theta_{11} x]$ (equation 3.3) and $y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$ (equation 3.4). On multiplying the parameters θ_{10} and θ_{11} by a positive constant α, we get $h_1' = \alpha h_1 = \alpha \cdot a[\theta_{10} + \theta_{11} x]$ (proof in problem 3.5). The new equation for equation 3.4 becomes $y = \phi_0 + \phi_1 \alpha h_1 + \phi_2 h_2 + \phi_3 h_3$. If we divide the slope ϕ_1 by α we get back the original equation.

Problem 3.7
We have $y = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11} x] + \phi_2 a[\theta_{20} + \theta_{21} x] = 0$ (equation 3.1). Since the model has many parameters, this increases the likelihood of multiple parameter sets achieving the same loss and hence, a model does not always have a single "best" set of parameters.

Problem 3.10
The neural network with one input, one output and 3 hidden units can have almost 4 linear regions. The number of linear regions could be less if there are less number of joints i.e. 2 or more activation functions are activated with the same value of x. This can also occur if the activation function always returns 0 or if the activation function is linear.

Problem 3.11
Number of parameters = Weights + Biases = $12 + 6 = 18$.

Problem 3.12
Number of parameters = Weights + Biases = $9 + 4 = 13$.

Problem 3.14
$h_1 = \theta_{10} + \theta_{11} x_1 + \theta_{12} x_2 + \theta_{13} x_3$
$h_2 = \theta_{20} + \theta_{21} x_1 + \theta_{22} x_2 + \theta_{23} x_3$
$h_3 = \theta_{30} + \theta_{31} x_1 + \theta_{32} x_2 + \theta_{33} x_3$

$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}$
$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}$

Problem 3.15
The number of linear region D_i inputs and D hidden units is given by Zaslavsky's formula:

$$N = \sum_{j=0}^{D_i} \binom{D}{j} = \sum_{j=0}^{D_i} \frac{D!}{(D-j)!\,j!}$$

Substituting the value of the number of input and hidden units, we get 8.

Problem 3.16
Equation of hidden unit:

$$\forall i \in \{1, 2, 3, 4\},\ h_i = \sum_{j=1}^{2} \theta_{ij} x_j + \theta_{i0}$$

Equation of output:

$$\forall i \in \{1, 2\},\ y_i = \sum_{k=1}^{4} \phi_{ik} h_k + \phi_{i0}$$

Problem 3.17
The number of weights = $D_i * D + D * D_o$
The number of biases = $D + D_o$.
Total number of parameters = $D(D_i + D_o + 1) + D_o$.

Problem 3.18
Using the Zaslavsky formula for the maximum number of regions with $D_i = 2$, $D = 5$, we get 16.

Chapter IV

Deep neural networks

Notebook Problems

1. Notebook 4.1 – Composing networks
2. Notebook 4.2 – Clipping functions
3. Notebook 4.3 – Deep neural networks

Text Book Problems

Problem 4.1
There is a similar problem in notebook 4.1- composing networks.

Problem 4.2
Number of hidden layers = 3.
Number of hidden units in each layer: h_1 = 4, h_2 = 2, h_3 = 3.

Problem 4.3
Given λ_0 and λ_1 are non-negative scalar,
we have LHS: $ReLU[\beta_1 + \lambda_1\Omega_1 ReLU[\beta_0 + \lambda_0\Omega_0 x]]$
$= ReLU[\beta_1 + \lambda_1\Omega_1\lambda_0 ReLU[\beta_0/\lambda_0 + \Omega_0 x]]$
$= \lambda_0\lambda_1 ReLU[\frac{\beta_1}{\lambda_0\lambda_1} + \lambda_1\Omega_1\lambda_0 ReLU[\beta_0/\lambda_0 + \Omega_0 x]] =$ RHS.

Problem 4.4
General equation of each hidden unit $h_k = a[\beta_{k-1} + \Omega_{k-1}x]$.
General equation of model $y = \beta_4 + \Omega_4 a[\beta_3 + \Omega_3 a[\beta_2 + \Omega_2 a[\beta_1 + \Omega_1 x]]]$

Dimensions: $\beta_0 = 20, \beta_1 = 10, \beta_2 = 7, \beta_3 = 4, \Omega_0 = 20 \times 5, \Omega_1 = 10 \times 20, \Omega_2 = 7 \times 10, \Omega_3 = 4 \times 7$.

Problem 4.5

Depth = 20 (number of hidden layers) and width = 20 (number of hidden units).

Problem 4.6

Number of weights in the current network:
$10 + (10 \times 10 + 10 \times 10 ...9 \text{ times }) + 10 = 920$.
Number of weights if we increase the depth by one (increase one more hidden layer):
$10 + (10 \times 10 + 10 \times 10 ...10 \text{ times }) + 10 = 1020$.
Number of weights if we increase the width by one (increase one more node to each hidden layer):
$11 + (11 \times 11 + 11 \times 11 ...9 \text{ times }) + 11 = 1111$.
By increasing the network's width or depth, the number of weights increases as well.

Problem 4.7

For an identity function $f, f[x] = x, \ for \ x \in [a, b]$.
Equation 3.1: $\phi_0 + \phi_1 a[\theta_{10} + \theta_{11} x] + \phi_2 a[\theta_{20} + \theta_{21} x] + \phi_3 a[\theta_{30} + \theta_{31} x]$. We can get an identity function for $\phi_0 = a, \phi_1 = 1, \theta_{10} = -a$. keeping the rest of the parameters zero.

Problem 4.8

From the given values of theta, we have the equations:
$h_1 = -1/6 + x$
$h_2 = -2/6 + x$
$h_3 = 4/6 - x$
In the left-most area, only the third function is active and we want $(0, 0)$ and $(1/6, 1)$ in that linear part, which gives the equation $y = 6x$. By comparing this equation with the third one, we get $\phi_3 = -6, \phi_0 = 4$.
In the second linear part, the equation moves from $(1/6, 1)$ to $(2/6, 0)$ and the first and third functions are active. From these points, we get the slope = -6. Therefore, $\phi_1\theta_1 + \phi_3\theta_3 = -6$, so $\phi_1 = -12$.
In the third region, the equation moves from $(2/6, 0)$ to $(4/6, 1)$, which gives slope = 3 and all three functions are active in that region. Therefore, $\phi_1\theta_1 + \phi_2\theta_2 + \phi_3\theta_3 = 3$. By substituting the values of phi and theta, we get $\phi_2 = 9$.

Problem 4.9

A shallow network with only 2 hidden units can create a function with at most 2

linear regions. It is possible to have 3 linear regions with 3 hidden units or more.

Problem 4.10

Total number of weights = weights by input layer and first hidden layer + weights by hidden layers + weights by last hidden layer and output layer. $= 1 \cdot D + D^2(K-1) + D \cdot 1$.

Total number of biases = number of neurons except for the input layer $= D \cdot K + 1$.

Total number of parameters= $3D + 1 + (K-1)D(D+1)$.

Problem 4.11

The number of parameters in the shallow network = 286, the deep network also has 286 parameters.

The maximum number of linear regions by the shallow network is 96 and that by the deep network is $6^9 \cdot 6 = 6^{10}$.

Chapter V

Loss functions

Notebook Problems

1. Notebook 5.1: Least Squares Loss
2. Notebook 5.2 Binary Cross-Entropy Loss
3. Notebook 5.3 Multiclass Cross-Entropy Loss

Text Book Problems

Problem 5.1

$$\begin{aligned}
\text{sig}[z] &= \frac{1}{1+e^{-z}} \\
\text{Case 1: } z \rightarrow \infty & \\
&= \frac{1}{1+0} \quad (e^{-\infty} = 0) \\
&= 1 \\
\text{Case 2: } z = 0 & \\
&= \frac{1}{1+1} \quad (e^{0} = 1) \\
&= 0.5 \\
\text{Case 3: } z \rightarrow \infty & \\
&= \frac{1}{\infty} \quad (e^{\infty} = \infty) \\
&= 0
\end{aligned}$$

Problem 5.3

Step 1- Choose a suitable probability distribution:
It's given in the question to use the von Mises distribution for this problem.

Step 2- Set the model to predict the parameters:
So θ = f[$\mathbf{x}$, ϕ] and $Pr(\mathbf{y}|\theta) = Pr(\mathbf{y}|f[\mathbf{x}, \phi]])$. We can predict the mean of the distribution so:

$$Pr(y \mid f[\mathbf{x}, \phi], \kappa) = \frac{\exp\left[\kappa \cos(y - f[\mathbf{x}, \phi])\right]}{2\pi \cdot \text{Bessel}_0[\kappa]}$$

Step 3- Choose a suitable loss function to find the optimal $\hat{\theta}$:
Since the distribution is exponential, we can choose the negative log-likelihood as the loss function:

$$L[\phi] = -\sum_{i=1}^{I} \log\left[\frac{\exp\left[\kappa \cos(y - f[\mathbf{x}, \phi])\right]}{2\pi \cdot \text{Bessel}_0[\kappa]}\right]$$

We aim to find a distribution which minimizes the loss:

$$\begin{aligned}\hat{\phi} &= \underset{\phi}{\text{argmin}}\left[-\sum_{\mathbf{x}} \frac{\kappa \cos(y - f[\mathbf{x}, \phi])}{2\pi \cdot \text{Bessel}_0[\kappa]}\right] \\ &= \underset{\phi}{\text{argmin}}\left[-\sum_{\mathbf{x}} \cos(y - f[\mathbf{x}, \phi])\right]\end{aligned}$$

Step 4- To perform inference, return either the full distribution or the value where this distribution is maximised:

$$\hat{y} = \underset{y}{\text{argmax}}\left[Pr(y|f[\mathbf{x}, \hat{\phi}])\right]$$

Problem 5.4

$$\begin{aligned}L = \sum_{i=1}^{I} \log\Bigg[&\frac{\text{sig}[f_1[\mathbf{x_i}, \phi]]}{\sqrt{2\pi f_3[\mathbf{x_i}, \phi]^2}} \exp\left[-\frac{(y_i - f_2[\mathbf{x_i}, \phi])^2}{2f_3[\mathbf{x_i}, \phi]^2}\right] \\ &+ \frac{1 - \text{sig}[f_1[\mathbf{x_i}, \phi]]}{\sqrt{2\pi f_5[\mathbf{x_i}, \phi]^2}} \exp\left[-\frac{(y_i - f_4[\mathbf{x_i}, \phi])^2}{2f_5[\mathbf{x_i}, \phi]^2}\right]\Bigg]\end{aligned}$$

f_1, f_2, f_3, f_4, f_5 are the output of the function for $\lambda, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2$ respectively.
It's difficult to find the optimal values for the weights as this loss cannot be solved in a closed form. Hence, it would require optimization techniques like gradient descent

to find the parameters' values that minimises the loss.

Problem 5.5

$$Pr(y \mid \mu_1, \kappa_1, \mu_2, \kappa_2) = \frac{\exp\left[f_2[\mathbf{x}, \phi] \cos(y - f_1[\mathbf{x}, \phi])\right]}{2\pi \cdot \text{Bessel}_0[f_2[\mathbf{x}, \phi]]} + \frac{\exp\left[f_4[\mathbf{x}, \phi] \cos(y - f_3[\mathbf{x}, \phi])\right]}{2\pi \cdot \text{Bessel}_0[f_4[\mathbf{x}, \phi]]}$$

We are optimizing the model's parameters to find the distribution's parameters- $\mu_1, \kappa_1, \mu_2, \kappa_2$ given by the model's output f_1, f_2, f_3, f_4 respectively. Therefore, there are 4 outputs of the model.

Problem 5.6

The goal of the model would be to predict the parameter λ.

$$L = \sum_{i}^{I} \log \left[\frac{(\lambda - f[\mathbf{x}_i, \phi])^{k_i} e^{-(\lambda - f[\mathbf{x}_i, \phi])}}{k_i!} \right]$$

Simplifying and removing the constants, we get:

$$L = \sum_{i}^{I} \left[k_i \cdot \log\left[\lambda - f[\mathbf{x}_i, \phi]\right] + f[\mathbf{x}_i, \phi] - \log[k!] \right]$$

Problem 5.7

Likelihood for the model:

$$\hat{\phi} = \underset{\phi}{\text{argmax}} \prod_{i}^{I} \prod_{d}^{D} Pr(y_{id} \mid \mathbf{f}[\mathbf{x}_i, \phi])$$

Negative log-likelihood of the expression:

$$= \underset{\phi}{\text{argmin}} \left[-\sum_{i=1}^{I} \sum_{d=1}^{D} \log[Pr(y_{id} \mid \mathbf{f}[\mathbf{x}_{id}, \phi])] \right]$$

$$= \underset{\phi}{\text{argmin}} \left[-\sum_{i=1}^{I} \sum_{d=1}^{D} (y_{id} - \mathbf{f}[\mathbf{x}_{id}, \phi])^2 \right]$$

Problem 5.8

$$L = -\sum_{i=1}^{I} \sum_{d}^{D} \log \left[\frac{1}{\sqrt{2\pi f_{2d}[\mathbf{x}_i, \phi]}} \exp\left[-\frac{(y_i - f_{1d}[\mathbf{x}_i, \phi])^2}{2 f_{2d}[\mathbf{x}_i, \phi]} \right] \right]$$

f_{1d} predicts the mean while f_{2d} predicts the variance of the d^{th} dimension.

Problem 5.9

The height of an average human would be around 2 meters whereas the weight can be more than 100 kg. As a result, the model would be more inclined to optimize the parameters for weights as it contributes more to the total loss. So we need to scale these metrics so that the model would treat both the outputs fairly. The other way would be to learn the variance for both outputs.

Problem 5.10

$$L = \sum_{\mathbf{x}} \left[-\omega_1 \cos(y_{dir} - f_{1x}[\mathbf{x}, \phi]) + \omega_2 (y_{speed} - f_{2x}[\mathbf{x}, \phi])^2 \right]$$

f_{1x} predicts the direction while f_{2x} predicts the speed of the wind. ω is the scaling factor.

Chapter VI

Fitting models

Notebook Problems

1. Notebook 6.1: Line search
2. Notebook 6.2 Gradient descent
3. Notebook 6.3: Stochastic gradient descent
4. Notebook 6.4: Momentum
5. Notebook 6.5: Adam

Text Book Problems

Problem 6.1
Least square loss function:

$$\begin{aligned} L &= (\phi_0 + \phi_1 x - y)^2 \\ \frac{\partial L}{\partial \phi_0} &= 2(\phi_0 + \phi_1 x - y) \cdot \frac{\partial (\phi_0 + \phi_1 x - y)}{\partial \phi_0} \\ &= 2(\phi_0 + \phi_1 x - y) \\ \frac{\partial L}{\partial \phi_1} &= 2(\phi_0 + \phi_1 x - y) \cdot \frac{\partial (\phi_0 + \phi_1 x - y)}{\partial \phi_1} \\ &= 2x(\phi_0 + \phi_1 x - y) \end{aligned}$$

Problem 6.2

$$L = (\phi_0 + \phi_1 x - y)^2$$

$$\begin{aligned}\mathbf{H}[L] &= \begin{bmatrix} \frac{\partial^2 L}{\partial \phi_0^2} & \frac{\partial^2 L}{\partial \phi_0 \partial \phi_1} \\ \frac{\partial^2 L}{\partial \phi_1 \partial \phi_0} & \frac{\partial^2 L}{\partial \phi_1^2} \end{bmatrix} \\ &= \begin{bmatrix} 2 & 2x \\ 2x & 2x^2 \end{bmatrix}\end{aligned}$$

Trace of $\mathbf{H}[L] = 2 + 2x^2$.
Determinant of $\mathbf{H}[L] = 4x^2 - 4x^2 = 0$.
As the determinant (the product of the eigenvalues) equals 0, one of the eigenvalues must be zero. Conversely, the trace (sum of eigenvalues) remains positive, suggesting that the other eigenvalue is also positive. Hence, the function is convex.

Problem 6.3
Gabor model: $G = \sin\left(\phi_0 + 0.06 \cdot \phi_1 x\right) \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{32.0}\right)$.
Least square loss: $L = (G - y)^2$

$$\begin{aligned}\frac{\partial L}{\partial \phi_0} &= 2(G - y) \cdot \frac{\partial G}{\partial \phi_0} \\ \frac{\partial G}{\partial \phi_0} &= \cos(\phi_0 + 0.06 \cdot \phi_1) \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{32.0}\right) \\ &\quad + \sin\left(\phi_0 + 0.06 \cdot \phi_1 x\right) \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{32.0}\right) \\ &\quad \cdot \left(-2\frac{(\phi_0 + 0.06 \cdot \phi_1 x)}{32.0}\right) \\ \frac{\partial L}{\partial \phi_0} &= 2(G - y) \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{32.0}\right) \\ &\quad \cdot \left[\cos(\phi_0 + 0.06 \cdot \phi_1) - \sin\left(\phi_0 + 0.06 \cdot \phi_1 x\right) \cdot 2\frac{(\phi_0 + 0.06 \cdot \phi_1 x)}{32.0}\right] \\ \frac{\partial L}{\partial \phi_1} &= 2(G - y) \cdot \frac{\partial G}{\partial \phi_1} \\ \frac{\partial L}{\partial \phi_1} &= 2(G - y) x \exp\left(-\frac{(\phi_0 + 0.06\phi_1 x)^2}{32}\right) \cdot \\ &\quad [0.06\cos(\phi_0 + 0.06\phi_1 x) - \\ &\quad \frac{0.12}{32}(\phi_0 + 0.06\phi_1 x)\sin(\phi_0 + 0.06\phi_1 x)\Big]\end{aligned}$$

Problem 6.4
i) ϕ_0 (bias) shifts the graph right or left, or up or down if the ϕ_1 (slope) is nearly zero. ϕ_1 increases the steepness of the graph.

ii) Binary cross-entropy is a suitable loss function because there are only two classes- $\{0, 1\}$.

$$L = \sum_{i=1}^{I} -(1 - y_i) \log[1 - \text{sig}[\phi_0 + \phi_1 x]] - y_i \log[\text{sig}[\phi_0 + \phi_1 x]]$$

iii)

$$\frac{\partial L}{\partial \phi_0} = \sum_{i=1}^{I} \left(\frac{1 - y_i}{1 - \text{sig}(\phi_0 + \phi_1 x_i)} + \frac{y_i}{\text{sig}(\phi_0 + \phi_1 x_i)} \right) \frac{\exp(-\phi_0 - \phi_1 x_i)}{(1 + \exp(-\phi_0 - \phi_1 x_i))^2}$$

$$\frac{\partial L}{\partial \phi_1} = \sum_{i=1}^{I} \left(\frac{1 - y_i}{1 - \text{sig}(\phi_0 + \phi_1 x_i)} + \frac{y_i}{\text{sig}(\phi_0 + \phi_1 x_i)} \right) \frac{x_i \cdot \exp(-\phi_0 - \phi_1 x_i)}{(1 + \exp(-\phi_0 - \phi_1 x_i))^2}$$

Supplementary notebook

Problem 6.5

$$\begin{aligned}
\frac{\partial f(x_i, \phi)}{\partial \phi_0} &= 1 \\
\frac{\partial f(x_i, \phi)}{\partial \phi_1} &= a[\theta_{10} + \theta_{11} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \phi_2} &= a[\theta_{20} + \theta_{21} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \phi_3} &= a[\theta_{30} + \theta_{31} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{10}} &= \phi_1 \mathbb{I}[\theta_{10} + \theta_{11} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{11}} &= \phi_1 x_i \mathbb{I}[\theta_{10} + \theta_{11} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{20}} &= \phi_2 \mathbb{I}[\theta_{20} + \theta_{21} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{21}} &= \phi_2 x_i \mathbb{I}[\theta_{20} + \theta_{21} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{30}} &= \phi_3 \mathbb{I}[\theta_{30} + \theta_{31} x_i] \\
\frac{\partial f(x_i, \phi)}{\partial \theta_{31}} &= \phi_3 x_i \mathbb{I}[\theta_{30} + \theta_{31} x_i]
\end{aligned}$$

$\mathbb{I}$ is the indicator function, defined as:

$$\mathbb{I}[z] = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z <= 0 \end{cases}$$

Problem 6.6
Only the figure 6.11-b represents a convex function. For a function to be convex, any line from one point on the curve to another should lie above the function.
Local minimum: $1, 3$.
Global minimum: $2, 5, 6$.

Problem 6.7
In gradient descent, we move in the direction of the steepest descent and continue in that direction until the slope becomes zero. At this point, we adjust the direction to be perpendicular to the previous trajectory. Each time we update the parameters, we tend to overshoot along the previous trajectory, leading to oscillations in the path. To avoid this behaviour, we add momentum to the trajectory to "remember" the previous path we took, and hence, the curve becomes smoother.

Problem 6.8
Yes, if the learning rate is high enough.

Problem 6.9
200 epochs.

Problem 6.10

$$\begin{aligned}\mathbf{m_t} &= \beta \cdot \mathbf{m_{t-1}} + (1-\beta)\cdot g_t \\ &= \beta \cdot (\beta \cdot \mathbf{m_{t-2}} + (1-\beta)\cdot g_{t-1}) + (1-\beta)\cdot g_t \\ &= \beta \cdot (\beta \cdot (\beta \cdot \mathbf{m_{t-3}} + (1-\beta)\cdot g_t)) + (1-\beta)\cdot g_{t-1}) + (1-\beta)\cdot g_t \\ &= \ \ldots\end{aligned}$$

Problem 6.11
1 millon $\times$ 1 million.

Chapter VII

Gradients and initialization

Notebook Problems

1. Notebook 7.1: Backpropagation in Toy Model
2. Notebook 7.2: Backpropagation
3. Notebook 7.3: Initialization

Text Book Problems

Problem 7.1

$$y = \phi_0 + \phi_1 a[\psi_{01} + \psi_{11} a[\theta_{01} + \theta_{11} x] + \psi_{21} a[\theta_{02} + \theta_{12} x]] + \\ \phi_2 a[\psi_{02} + \psi_{12} a[\theta_{01} + \theta_{11} x] + \psi_{22} a[\theta_{02} + \theta_{12} x]]$$

$$\frac{\partial y}{\partial \phi_0} = 1$$

$$\frac{\partial y}{\partial \phi_1} = a[\psi_{01} + \psi_{11} a[\theta_{01} + \theta_{11} x] + \psi_{21} a[\theta_{02} + \theta_{12} x]]$$

$$\frac{\partial y}{\partial \phi_2} = a[\psi_{02} + \psi_{12} a[\theta_{01} + \theta_{11} x] + \psi_{22} a[\theta_{02} + \theta_{12} x]]$$

$$\frac{\partial y}{\partial \psi_{01}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0]$$

$$\frac{\partial y}{\partial \psi_{02}} = \phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0]$$

$$\frac{\partial y}{\partial \psi_{11}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot a[\theta_{01} + \theta_{11}x]$$

$$\frac{\partial y}{\partial \psi_{12}} = \phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot a[\theta_{01} + \theta_{11}x]$$

$$\frac{\partial y}{\partial \psi_{21}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot a[\theta_{02} + \theta_{12}x]$$

$$\frac{\partial y}{\partial \psi_{22}} = \phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot a[\theta_{02} + \theta_{12}x]$$

$$\frac{\partial y}{\partial \theta_{01}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{11} \cdot \mathbb{I}[\theta_{01} + \theta_{11}x > 0]$$
$$+\phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{12} \cdot \mathbb{I}[\theta_{01} + \theta_{11}x > 0]$$

$$\frac{\partial y}{\partial \theta_{02}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{21} \cdot \mathbb{I}[\theta_{02} + \theta_{12}x > 0]$$
$$+\phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{22} \cdot \mathbb{I}[\theta_{02} + \theta_{12}x > 0]$$

$$\frac{\partial y}{\partial \theta_{11}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{11} \cdot \mathbb{I}[\theta_{01} + \theta_{11}x > 0] \cdot x$$
$$+\phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{12} \cdot \mathbb{I}[\theta_{01} + \theta_{11}x > 0] \cdot x$$

$$\frac{\partial y}{\partial \theta_{12}} = \phi_1 \cdot \mathbb{I}[\psi_{01} + \psi_{11}a[\theta_{01} + \theta_{11}x] + \psi_{21}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{21} \cdot \mathbb{I}[\theta_{02} + \theta_{12}x > 0] \cdot x$$
$$+\phi_2 \cdot \mathbb{I}[\psi_{02} + \psi_{12}a[\theta_{01} + \theta_{11}x] + \psi_{22}a[\theta_{02} + \theta_{12}x] > 0] \cdot \psi_{22} \cdot \mathbb{I}[\theta_{02} + \theta_{12}x > 0] \cdot x$$

Problem 7.2

$$\frac{\partial f_3}{\partial h_3} = \omega_3$$
$$\frac{\partial l_i}{\partial f_3} = 2(f_3 - y_i)$$
$$\frac{\partial l_i}{\partial h_3} = 2\omega_3(\beta_3 + \omega_3 \cdot h_3 - y_i)$$

Problem 7.3

$$\frac{\partial l_i}{\partial \mathbf{f}_1} = D_2 \times 1$$

$$\frac{\partial \mathbf{h}_2}{\partial \mathbf{f}_1} = D_2 \times D_2$$
$$\frac{\partial \mathbf{f}_2}{\partial \mathbf{h}_2} = D_2 \times D_3$$
$$\frac{\partial \mathbf{h}_3}{\partial \mathbf{f}_2} = D_3 \times D_3$$
$$\frac{\partial \mathbf{f}_3}{\partial \mathbf{h}_3} = D_3 \times D_f$$
$$\frac{\partial l_i}{\partial \mathbf{f}_3} = D_f \times 1$$

Problem 7.4

$$l_i = (y_i - \mathrm{f}[x_i, \phi])^2$$
$$\frac{\partial l_i}{\partial \mathrm{f}[x_i, \phi]} = -2(y_i - \mathrm{f}[x_i, \phi])$$

Problem 7.5

$$l_i = -(1 - y_i)\log[1 - \mathrm{sig}[f(x_i, \phi)]] - y_i \log[\mathrm{sig}[f(x_i, \phi)]]$$
$$\frac{\partial \ell_i}{\partial f[x_i, \phi]} = \mathrm{sig}(f[x_i, \phi]) - y_i$$

Problem 7.6

Given in the problem $\frac{\partial z}{\partial h}$, contains the term $\frac{\partial z_i}{\partial h_j}$ in its i^{th} column and j^{th} row. But its derivative is Ω_{ij} is contained at i^{th} row and j^{th} column. Hence the derivative of z w.r.t h would be Ω^T.

Problem 7.7

$$h = \mathrm{sig}[f]$$
$$\frac{\partial h}{\partial f} = \frac{e^{-f}}{(1 + e^{-f})^2}$$

i) If the input to the derivative of the sigmoid function is a large positive value, it returns zero.
ii) For large negative values, it also returns zero.

Problem 7.8

For both functions, the gradient change concerning each activation function would be zero for most of the input and the function is not differentiable at the threshold

points. This would render the network inactive by preventing weight updates.

Problem 7.9

$$\frac{\partial f_i}{\partial \Omega_{ij}} = \mathbf{h_j}$$
$$\frac{\partial l}{\partial \boldsymbol{\Omega}} = \frac{\partial l}{\partial \mathbf{f}} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\Omega}}$$
$$\frac{\partial l}{\partial \boldsymbol{\Omega}} = \frac{\partial l}{\partial \mathbf{f}} \mathbf{h}^T$$

Problem 7.10

The derivative of ReLU:

$$\frac{\partial a}{\partial f} = \begin{cases} 1 & \text{if } x > 0, \\ \alpha & \text{if } x \leq 0. \end{cases}$$

Only the derivative equation $\frac{\partial l}{\partial f}$ will be changed, rest will be updated automatically using the chain rule:

$$\frac{\partial l_i}{\partial \mathbf{f}_{k-1}} = \mathbb{I}[\mathbf{f}_{k-1} > 0] \odot \left(\boldsymbol{\Omega}_k^T \frac{\partial l_i}{\partial \mathbf{f}_k} \right) + \mathbb{I}[\mathbf{f}_{k-1} \leq 0] \odot \alpha \left(\boldsymbol{\Omega}_k^T \frac{\partial l_i}{\partial \mathbf{f}_k} \right)$$

Problem 7.11

During backpropagation, the value of the activation function is required to calculate the gradients of weights and biases. Only after computing the corresponding gradients can the intermediate variable f be removed from memory. However, storing many intermediate variables may lead to memory exhaustion. Therefore, instead of retaining these variables, they are saved at checkpoints, allowing for the next activation function's value to be calculated as needed. A useful illustration of this algorithm is available on GitHub repository[1].

Problem 7.12

$$\frac{\partial y}{\partial f_5} = 1$$
$$\frac{\partial y}{\partial f_4} = 1$$
$$\frac{\partial y}{\partial f_3} = \frac{\partial f_4}{\partial f_3}\frac{\partial y}{\partial f_4} + \frac{\partial f_5}{\partial f_3}\frac{\partial y}{\partial f_5} = \exp[f_3] + \cos[f_3]$$
$$\frac{\partial y}{\partial f_2} = \frac{\partial f_3}{\partial f_2}\frac{\partial y}{\partial f_3} = \exp[f_3] + \cos[f_3]$$
$$\frac{\partial y}{\partial f_1} = \frac{\partial f_2}{\partial f_1}\frac{\partial y}{\partial f_2} + \frac{\partial f_3}{\partial f_1}\frac{\partial y}{\partial f_3} = 2f_1(\exp[f_3] + \cos[f_3]) + \exp[f_3] + \cos[f_3]$$

[1] https://github.com/cybertronai/gradient-checkpointing

$$\frac{\partial y}{\partial x} = \frac{\partial f_1}{\partial x}\frac{\partial y}{\partial f_1} = \exp[x](2f_1(\exp[f_3] + \cos[f_3]) + \exp[f_3] + \cos[f_3])$$

Problem 7.13

$$\begin{aligned}
\frac{\partial f_1}{\partial x} &= \exp[x] \\
\frac{\partial f_2}{\partial x} &= \frac{\partial f_1}{\partial x}\frac{\partial f_2}{\partial f_1} = 2f_1\exp[x] \\
\frac{\partial f_3}{\partial x} &= \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial x} = \exp[x] + 2f_1\exp[x] \\
\frac{\partial f_4}{\partial x} &= \frac{\partial f_1}{\partial x}\frac{\partial f_2}{\partial f_1}\frac{\partial f_3}{\partial f_2}\frac{\partial f_4}{\partial f_3} = (\exp[x] + 2f_1\exp[x])\exp[f_3] \\
\frac{\partial f_5}{\partial x} &= \frac{\partial f_3}{\partial x}\frac{\partial f_5}{\partial f_3} = (\exp[x] + 2f_1\exp[x])\cos[f_3] \\
\frac{\partial y}{\partial x} &= \frac{\partial f_4}{\partial x} + \frac{\partial f_5}{\partial x} = (\exp[x] + 2f_1\exp[x])\exp[f_3] + (\exp[x] + 2f_1\exp[x])\cos[f_3]
\end{aligned}$$

Forward-mode differentiation is not used in practice because it computes derivatives of intermediate values with respect to one input parameter at a time. For each parameter, you need to propagate derivatives through the entire network. While in backpropagation, one forward pass and one backward pass are enough to calculate the gradients.

Problem 7.14

$$\begin{aligned}
\text{Var}[a] &= \mathbb{E}[a^2] - \mathbb{E}[a]^2 \\
&= \mathbb{E}[a^2] \qquad (\mathbb{E}[a] = 0) \\
\mathbb{E}[a^2] &= \sigma^2
\end{aligned}$$

Since the distribution a is symmetrical around 0, $\mathbb{E}[a^2_{a<=0}] = \mathbb{E}[a^2_{a>0}] = \sigma^2/2$

$$\begin{aligned}
\mathbb{E}[b^2] &= \mathbb{E}[\text{ReLU}[a]^2] \\
&= \mathbb{E}[\text{ReLU}[a_{a<=0}]^2] + \mathbb{E}[\text{ReLU}[a_{a>0}]^2] \\
&= 0 + \mathbb{E}[a^2_{a>0}] \\
&= \sigma^2/2
\end{aligned}$$

Problem 7.15

The network will not learn effectively. If all the weights are zero, then the gradient due to the previous hidden layers will also be zero as they are multiplied by the weights. Also, since all the weights and biases are the same, the update will be the same in a particular layer, so the network cannot learn diversified patterns.

Problem 7.16
Supplementary notebook.

Problem 7.17
Supplementary notebook.

Chapter VIII

Measuring performance

Notebook Problems

1. Notebook 8.1: MNIST_1D_Performance
2. Notebook 8.2: Bias-Variance Trade-Off
3. Notebook 8.3: Double Descent
4. Notebook 8.4: High-dimensional spaces

Text Book Problems

Problem 8.1
Yes, the training loss, in this case, can reach zero. For example, if the number of linear regions of the model is equal to the number of data samples in the dataset, the training loss could reach zero as each linear region could represent a data sample.

Problem 8.2
The joints of the model are equally spread across the interval- at 0, $1/3$ and $2/3$, so we can choose all the weights as 1 (as the slope is 1) and biases- 0, $-1/3$ and $-2/3$ ($weight * x + bias >= 0$).

Problem 8.3
Choose the best-fit line for each of the piece-wise linear regions. Linear regression

problems have a definite solution in which the loss is minimal.

Problem 8.4

The training performance will improve and the loss could even reach zero as the number of linear regions will be equal to the number of examples so that each linear region will correspond to one example. As for the test performance, the loss due to bias will decrease but that due to variance will increase. Overall, it will not have much effect on the test performance.

Problem 8.5

The model will overfit and loss due to bias will decrease but the variability of the model output will increase. Increasing the number of training samples or using regularization techniques can help resolve the problem.

Problem 8.7

$$\text{Vol}[r] = \frac{r^D \pi^{D/2}}{\Gamma(D/2+1)}$$

Γ is the Gamma function and is an extension of the factorial function for continuous values. Effectively $\Gamma[x] = (x-1)!$. For simplicity, we assume that D is an even number, so we have:

$$\begin{aligned}\text{Vol}[r] &= \frac{0.5^D \pi^{D/2}}{D/2!} \\ &= \frac{0.5^D \pi^{D/2}}{\sqrt{2\pi D/2}(\frac{D}{2e})^{D/2}} \qquad \text{Using Stirling's for factorial approximation}\end{aligned}$$

As $D \rightarrow \infty$, $0.5^D \rightarrow 0$ and the denominator increases exponentially due to the factorial. Hence, in higher dimensions, the volume of a hyper-sphere becomes zero.

Problem 8.8

$$\begin{aligned}\text{Vol}_{r=1} &= \frac{\pi^{D/2}}{\Gamma(D/2+1)} \\ \text{Vol}_{r=0.99} &= \frac{0.99^D \pi^{D/2}}{\Gamma(D/2+1)} \\ \text{Proportion} &= \frac{\text{Vol}_{r=1} - \text{Vol}_{r=0.99}}{\text{Vol}_{r=1}} \\ &= 1 - 0.99^D\end{aligned}$$

As $D \to \infty$, the proportion reaches 1.

Problem 8.9
Supplementary notebook.

Chapter IX

Regularization

Notebook Problems

1. Notebook 9.1: L2 Regularization
2. Notebook 9.2: Implicit Regularization
3. Notebook 9.3: Ensembling
4. Notebook 9.4: Bayesian approach
5. Notebook 9.5: Augmentation

Text Book Problem

Problem 9.1
Prior distribution:

$$Pr(\phi) = \prod_{j=1}^{J} \text{Norm}_{\phi_j}[0, \sigma_\phi^2]$$

Maximum Likelihood:

$$\hat{\phi} = \underset{\phi}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(y_i|x_i, \phi) \right]$$

Applying prior to the maximum likelihood, we get the posterior:

$$\hat{\phi} = \underset{\phi}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(y_i|x_i, \phi) Pr(\phi) \right]$$

Applying negative log-likelihood to the function, we get:

$$\begin{aligned} L[\phi] &= -\log\left(\prod_{i=1}^{I} Pr(y_i|x_i,\phi)Pr(\phi))\right) \\ &= -\left(\sum_{i=1}^{I} Pr(y_i|x_i,\phi) + \sum_{j=1}^{J}\log(\text{Norm}_{\phi_j}[0,\phi_j])\right) \end{aligned}$$

Comparing it with the equation of L2 regularization:

$$\hat{\phi} = \underset{\phi}{\text{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i,\mathbf{y}_i] + \lambda\sum_{j}\phi_j^2\right],$$

We can see the new loss function is equivalent to the L2 regularization.

Problem 9.2
$2\lambda\sum_j^J \phi_j$ is added to the original loss gradients because the loss function changes with the addition of the regularization term.

Problem 9.3
Let ϵ_i be the noise in the ith input.

$$\begin{aligned} L &= \sum_i (f[x_i+\epsilon_i,\phi] - y_i)^2 \\ &= \sum_i (\phi_0 + \phi_1(x_i+\epsilon_i) - y_i)^2 \\ \mathbb{E}\left[\frac{\partial L}{\partial\phi_0}\right] &= \mathbb{E}\left[2\sum_i(\phi_0+\phi_1(x_i+\epsilon_i)-y_i)\right] \end{aligned}$$

Removing the expected operator as adding Gaussian noise does not affect gradient w.r.t ϕ_0.

$$= 2\sum_i(\phi_0+\phi_1 x_i - y_i) \qquad (\mathbb{E}[\epsilon]=0)$$

Now finding the expected gradient update w.r.t ϕ_1:

$$\begin{aligned} \mathbb{E}\left[\frac{\partial L}{\partial\phi_1}\right] &= \mathbb{E}\left[2\sum_i(\phi_0+\phi_1(x_i+\epsilon_i)-y_i)(x_i+\epsilon_i)\right] \\ &= 2\mathbb{E}\left[\sum_i(\phi_0+\phi_1(x_i+\epsilon_i)-y_i)x_i + \sum_i(\phi_0+\phi_1(x_i+\epsilon_i)-y_i)\epsilon_i\right] \end{aligned}$$

$$
\begin{aligned}
&= 2\mathbb{E}\left[\sum_i(\phi_0+\phi_1 x_i - y_i)x_i + \sum_i \phi_1\epsilon_i^2\right] \qquad (\mathbb{E}[\epsilon]=0)\\
&= 2\sum_i(\phi_0+\phi_1 x_i - y_i)x_i + 2\sum_i \phi_1\sigma_x^2 \qquad (\mathbb{E}[\epsilon^2]=\sigma_x^2)
\end{aligned}
$$

Problem 9.4

$$P_r(y_i|x_i,\phi) = 0.9\cdot\text{softmax}_{y_i}\left[f(x_i,\phi)\right] + \sum_{z\in\{1,\ldots,D_o\}\setminus y_i}\frac{0.1}{D_o-1}\cdot\text{softmax}_z\left[f(x_i,\phi)\right]$$

We get the cross-entropy loss by applying the negative log-likelihood function.

Problem 9.5

$$
\begin{aligned}
\phi &\leftarrow \phi - \alpha\left[\frac{\lambda\phi}{\alpha} + \frac{\partial L}{\partial\phi}\right] \qquad \text{(given)}\\
\tilde{L}[\phi] &= L[\phi] + \frac{\lambda}{2\alpha}\sum_k \phi_k^2 \qquad \text{(given)}\\
\frac{\partial\tilde{L}}{\partial\phi} &= \frac{\partial L}{\partial\phi} + \frac{\lambda}{\alpha}\phi
\end{aligned}
$$

Updating the weights:

$$
\begin{aligned}
\phi &\leftarrow \phi - \alpha\frac{\partial\tilde{L}}{\partial\phi}\\
&= \phi - \alpha\left[\frac{\lambda}{\alpha}\phi + \frac{\partial L}{\partial\phi}\right]
\end{aligned}
$$

Hence, proved.

Problem 9.6
Supplementary notebook.

Chapter X

Convolutional networks

Notebook Problems

1. Notebook 10.1: 1D Convolution
2. Notebook 10.2: Convolution for MNIST-1D
3. Notebook 10.3: 2D Convolution
4. Notebook 10.4: Downsampling and Upsampling
5. Notebook 10.5: Convolution for MNIST

Textbook Problem

Problem 10.1
To prove the operations are equivariant, we can show that they yield the same result irrespective of their applied order:
Case 1:

$$\begin{aligned} \text{Applying translation:} \quad & x_i \rightarrow x_{i+k} \\ \text{Applying convolution:} \quad & z_{i+k} = \omega_1 x_{i+k-1} + \omega_2 x_{i+k} + \omega_3 x_{i+k+1} \end{aligned}$$

Case 2:

$$\text{Applying convolution:} \quad z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$$

Applying translation: $z_{i+k} \rightarrow \omega_1 x_{i+k-1} + \omega_2 x_{i+k} + \omega_3 x_{i+k+1}$

Problem 10.2

$z_i = \omega_1 x_{2i-2} + \omega_2 x_{2i-1} + \omega_3 x_{2i}$

Problem 10.3

$z_i = \omega_1 x_{i-2} + \omega_2 x_i + \omega_3 x_{i+2}$

Problem 10.4

$z_i = \omega_1 x_{3i-11} + \omega_2 x_{3i-8} + \omega_3 x_{3i-5} + \omega_4 x_{3i-2} + \omega_5 x_{3i+1} + \omega_6 x_{3i+4} + \omega_7 x_{3i+7}$

Problem 10.8

Total weights = 236 and total biases = 14.

Problem 10.9

7.

Problem 10.10

19.

Problem 10.11

$R_1 = 5, R_2 = 9, R_3 = 17$.

Problem 10.12

2500.

Problem 10.14

Weights = kernel size * number of kernels, biases = number of kernels.
Weights = $5 * 5 * 3 * 10$, biases = 10.

Problem 10.16

Parameters = Weights + Biases = Kernel Size * Number of kernels + Biases:
$Conv_1 : 11 \times 11 = 11 * 11 * 3 * 96 + 96$
$Conv_2 : 5 \times 5 = 5 * 5 * 96 * 256 + 256$
$Conv_3 : 3 \times 3 = 3 * 3 * 256 * 384 + 384$
$Conv_4 : 3 \times 3 = 3 * 3 * 384 * 384 + 384$
$Conv_5 : 3 \times 3 = 3 * 3 * 384 * 256 + 256$
$FC_1 = 6 * 6 * 256 * 4096 + 4096$
$FC_2 = 4096 * 1000 + 1000$
Total parameters = 62378344.

Chapter XI

Residual networks

Notebook Problems

1. Notebook 11.1: Shattered gradients
2. Notebook 11.2: Residual Networks
3. Notebook 11.3: Batch normalization

Textbook Problem

Problem 11.1

$$\mathbf{h}_1 = \mathbf{x} + \mathbf{f}_1[\mathbf{x}] \tag{1}$$
$$\mathbf{h}_2 = \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1] \tag{2}$$
$$\mathbf{h}_3 = \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2] \tag{3}$$
$$\mathbf{y} = \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3] \tag{4}$$

Sequentially substituting (1) in (2), (2) in (3) and (3) in (4).

$$\begin{aligned}\mathbf{y} = \mathbf{x} &+ \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\Big] + \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\Big]\Big] \\ &+ \mathbf{f}_4\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\Big] + \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\Big]\Big]\Big]\end{aligned}$$

Problem 11.6

Total weights due to hidden layer = $20 + 9 * 20 * 20 + 20$ (Input layer to hidden layer 1 + hidden layer 1 till hidden layer 10 + hidden layer 10 to output layer)

Total biases = $10 * 20 + 1$ (All hidden layer + output layer)
Total weights due to batch normalization = $20 * 10 * 2$ (Total hidden units * 2).
Total parameters = 4241.

Problem 11.7
Weights will decrease in magnitude as $L2$ regularization punishes larger weights and the scaling parameter γ will increase proportionally to compensate. Eventually, the network becomes unstable due to very small weights.

Problem 11.8
Parameters due to batch normalization = $512 * 2$ (channels * 2)
Parameters due to conv layer = $512 * 3 * 3 * 512$ (input channels * kernel size * output size).
Total parameters = $23,60,320$.

Parameters due to the first block = $2 * 512 + 512 * 1 * 1 * 128$ (batch norm + conv layer)
Parameters due to the second block = $128 * 2 + 128 * 3 * 3 * 128$
Parameters due to the third block = $128 * 2 + 128 * 1 * 1 * 512$.
Total parameters = $280,064$.

Problem 11.9
U-Nets are not trained with images of arbitrary size because it makes training difficult as images of different sizes cannot be batched together and since the size it different, they will create feature maps of different sizes.

Chapter XII

Transformers

Notebook Problems

1. Notebook 12.1: Self Attention
2. Notebook 12.2: Multihead Self-Attention
3. Notebook 12.3: Tokenization
4. Notebook 12.4: Decoding strategies

Text Book Problems

Problem 12.1
Biases are a vector of size D, and weights are a matrix of dimension $D \times D$ in the self-attention mechanism. The total number of parameters to compute the queries, keys, and values is $3 \cdot (D + D \times D)$. The number of attention weights is $N \times N$ as attention scores are calculated for each token with every other token by taking the dot product of the query and the key matrix. A fully connected shallow network relating all DN inputs to all DN outputs will have $DN \times DN + DN$ (weights-$DN \times DN$, biases-DN) parameters.

Problem 12.2
Keeping the input and output dimensions the same would help with the model's scalability. The residual connections can then be added from the mechanism's input to its output.

Problem 12.3

$$\begin{aligned}
\textbf{Sa[XP]} &= V\ \text{Softmax}[Q^T K] \\
&= (\beta_v 1^T + \Omega_v X P)\ \text{Softmax}[(\beta_q 1^T + \Omega_q X P)^T (\beta_k 1^T + \Omega_k X P)] \\
&= (\beta_v 1^T + \Omega_v X) P\ \text{Softmax}[(\beta_q 1^T + \Omega_q X)^T P^T (\beta_k 1^T + \Omega_k X)] P \\
&(\beta \text{ is not depended on } X) \\
&= (\beta_v 1^T + \Omega_v X)\ \text{Softmax}[(\beta_q 1^T + \Omega_q X)^T (\beta_k 1^T + \Omega_k X)] P \\
&= \textbf{Sa[X]P}
\end{aligned}$$

Problem 12.4

$$\begin{aligned}
y_i = \text{softmax}[z_i] &= \frac{\exp[z_i]}{\sum_{j=1}^{5} \exp[z_j]} \\
\log[y_i] &= \log\left[\frac{\exp[z_i]}{\sum_{j=1}^{5} \exp[z_j]}\right] \\
&= z_i - \log\left[\sum_{j=1}^{5} \exp[z_j]\right] \\
\frac{\partial y_i}{\partial z_j} \cdot \frac{1}{y_i} &= \frac{\partial z_i}{\partial z_j} - \sum_j \frac{1}{\exp[z_j]} \cdot \exp[z_j] \\
\frac{\partial y_i}{\partial z_j} &= y_i\left[\mathbb{I}[i=j] - y_j\right]
\end{aligned}$$

Supplementary notebook.
We can note that the derivatives are very small numbers because the one element in the array is relatively bigger than the rest.

Problem 12.5
It prevents the overall bad initialization of the weights of the transformer layer and more operations can be done in parallel as compared to having a single head in the self-attention layer.

Problem 12.6
Predicting the masked token is an example of a generative task whereas classifying pairs of sentences as adjacent is contrastive. Using two tasks instead of one makes the model learn a better representation of the given input.
Two novel contrastive tasks are- classifying if two sentences are paraphrased and the tone of two sentences is the same or different.

Problem 12.7

Regular computation will be performed for the query, key, and value matrix. However, updating the query's dot product and the key matrix will only happen between the query of the new token and all other token's keys as the interaction between other's queries and the new token's key is masked.

Problem 12.8

1) Each patch would interact with adjacent patches only.
2) Each path would interact with every alternate patch only.

Problem 12.9

Computation required:
i) between patches, using all channels. The input embedding matrix would have the dimensions- 256×512 (number of patches * number of channels) So, the values, key and query matrix will have the same dimension as the input, but the query and key dot product will have the dimension- 256×256 (square of the number of tokens) to compute the interaction between every patch with every patch.
ii) between channels, using all patches. The input dimension will be reversed and be the following 512×256. The computation will remain the same except for the dot product calculation, which will now be 512×512. Calculating the product of the self-attention weights and values matrix will produce the output with the same dimension in both cases.
Hence, for the dot product calculation, the second case will cost more computation as the number of tokens has increased.

Problem 12.10

The new softmax will become:

$$\text{softmax}_m[\mathbf{k}^\mathbf{T} \cdot \mathbf{q}_n] = \frac{\mathbf{g}[\mathbf{k}_m]^T \mathbf{g}[\mathbf{q}_n]}{\sum_{m'=1}^{N} \mathbf{g}[\mathbf{k}_{m'}]^T \mathbf{g}[\mathbf{q}_n]}$$

$\sum_{m'=1}^{N} \mathbf{g}[\mathbf{k}_{m'}]^T$ can be calculated for all the attention weights and then multiplied, unlike in the case of regular softmax. It reduces the computation of calculating the softmax function from $O(N^2D)$ to $O(ND)$.