1. (3 Punkte) Gegeben ist `t = torch.arange(60)`, also ein Tensor mit den Zahlen 0 bis 59.
   (a) Welchen `shape` hat `t.reshape(3, 4, 5)`? Wie viele Dimensionen hat er?
   (b) Welchen Wert hat `t.reshape(3, 4, 5)[2][1][3]`?
   (c) Welchen Wert hat `t.reshape(6, 10)[4][7]`?
   (d) Welchen `shape` hat `t.reshape(5, -1)`?

2. (4 Punkte) Die vier Inputs von AND stehen als Zeilen in einer Matrix:

$$X = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

   (a) Berechne `z = X @ w + b` für $w = (1, 1)$ und $b = -1.5$. Welche Ausgaben liefert ein Perzeptron, das bei $z \geq 0$ feuert?
   (b) Drei Neuronen haben die Gewichte $(1, -1)$, $(-1, -1)$ und $(2, 1)$ und die Biaswerte 0, 1.5 und -2. Schreibt man die Gewichte als Zeilen einer Matrix W, berechnet `X @ W.T + b` alle Summen auf einmal. Welchen `shape` hat das Ergebnis? Berechne es.

3. (3 Punkte) (a) Wie viele Parameter hat eine Schicht `nn.Linear(784, 120)`?
   (b) Berechne die Zahl der Parameter für ein Netz mit den Schichten 784, 120, 60, 10 (wie im Notebook) und für ein Netz mit den Schichten 784, 256, 10.
   (c) Welche Schicht hat jeweils die meisten Parameter? Warum?

4. (3 Punkte) Die Ausgabeschicht eines Netzes mit drei Klassen liefert die Logits $(2, 1, 0)$.
   (a) Berechne mit der Softmax-Funktion die Wahrscheinlichkeiten der drei Klassen. Welche Klasse sagt das Netz voraus?
   (b) Die richtige Klasse ist die zweite (Nummer 1). Berechne den Fehler mit Cross-Entropy.
   (c) Wie müssten sich die Logits ändern, damit der Fehler kleiner wird?

5. (3 Punkte) Ein Netz hat für zwei Klassen zwei Ausgabeneuronen mit den Logits z und 0.
   (a) Zeige, dass die Softmax-Funktion der Klasse mit dem Logit z die Wahrscheinlichkeit $\sigma(z)$ gibt.
   (b) Was folgt daraus für Cross-Entropy und Binary Cross-Entropy?

6. (4 Punkte) Was gibt das Programm aus? Rechne von Hand mit einem Rechengraphen.

```
a = torch.tensor(2.0, requires_grad=True)
b = torch.tensor(-3.0, requires_grad=True)
c = a * b
d = c + a
e = d * d
e.backward()
print(e.item(), a.grad.item(), b.grad.item())
```

7. (3 Punkte) Gegeben ist das Programm

```
x = torch.tensor(3.0, requires_grad=True)
for i in range(3):
    f = x * x
    f.backward()
    print(x.grad.item())
```

   (a) Was gibt das Programm aus?

   (b) Warum steht in jedem Trainingsschritt `optimizer.zero_grad()`? Was würde ohne passieren?

8. (4 Punkte) So sieht das Training im Notebook aus:

```
for epoche in range(5):
    for bilder, labels in train_loader:
        loss = F.cross_entropy(netz(bilder), labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

   (a) Erkläre die vier Zeilen in der inneren Schleife.

   (b) Es gibt 60000 Trainingsbilder, die Batchgröße ist 100. Wie oft werden die Parameter insgesamt verändert?

   (c) Warum hat die letzte Schicht des Netzes keine Aktivierungsfunktion?

9. (3 Punkte) Ein Netz mit 102070 Parametern wurde mit nur 1000 Trainingsbildern trainiert. Die Genauigkeiten:

Epoche	Trainingsdaten	Testdaten
10	0.845	0.7684
30	0.936	0.7962
60	0.99	0.7993

   (a) Was fällt auf? Wie nennt man das?

   (b) Nenne zwei Maßnahmen dagegen.

   (c) Warum misst man die Genauigkeit überhaupt auf Testdaten?

10. (4 Punkte) Schreibe in Colab eine Klasse `NetzKlein` für Fashion-MNIST mit den Schichten 784, 64, 32, 10. Gib die Zahl der Parameter aus und prüfe mit `torch.randn(5, 1, 28, 28)`, dass das Netz für 5 Bilder die richtige Gestalt liefert. Trainiere es dann mit den Funktionen `trainiere` und `genauigkeit` aus dem Notebook fünf Epochen lang und vergleiche die Genauigkeit auf den Testdaten mit `NetzFashion`.