1. (3 Punkte) Berechne die Faltung (ohne Padding, Stride 1) des Bildes B mit dem Filter K, wie sie ein CNN rechnet: Der Filter wird über das Bild geschoben, an jeder Stelle werden die übereinander liegenden Zahlen multipliziert und addiert.

$$B = \begin{pmatrix} 1 & 2 & 0 & 1 & 3 \\ 0 & 1 & 2 & 3 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 2 & 0 \\ 3 & 1 & 2 & 0 & 1 \end{pmatrix} \qquad K = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

Welche Größe hat das Ergebnis? Berechne alle Einträge.

Lösung: Das Ergebnis ist 3×3, denn $5 - 3 + 1 = 3$.

$$\begin{pmatrix} -1 & -3 & -3 \\ 2 & -4 & -2 \\ 3 & -3 & 1 \end{pmatrix}$$

Oben links: $1 \cdot 1 + 0 \cdot 2 + (-1) \cdot 0 + 2 \cdot 0 + 0 \cdot 1 + (-2) \cdot 2 + 1 \cdot 2 + 0 \cdot 1 + (-1) \cdot 0 = -1$

2. (3 Punkte) Gegeben sind das Bild B und die Filter K_1 und K_2.

$$B = \begin{pmatrix} 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad K_1 = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \qquad K_2 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

   (a) Berechne die Faltungen von B mit K_1 und mit K_2.

   (b) Was erkennen die beiden Filter? Erkläre die Ergebnisse.

Lösung:

(a)

$$B * K_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad B * K_2 = \begin{pmatrix} 0 & 0 & 0 \\ 27 & 27 & 27 \\ 27 & 27 & 27 \end{pmatrix}$$

(b) K_1 zieht die rechte Spalte von der linken ab und erkennt senkrechte Kanten. Das Bild hat keine, das Ergebnis ist 0. K_2 zieht die untere Zeile von der oberen ab und erkennt waagrechte Kanten. Die Kante zwischen Zeile 3 und 4 ergibt große Werte in den Zeilen des Ergebnisses, bei denen der Filter über der Kante liegt.

3. (3 Punkte) Ein $n \times n$-Bild wird mit einem $f \times f$-Filter, Padding p und Stride s gefaltet. Das Ergebnis hat die Größe $\frac{n+2p-f}{s} + 1$ (abgerundet). Berechne die Größe für
   (a) $n = 28$, $f = 5$, $p = 2$, $s = 1$
   (b) $n = 32$, $f = 3$, $p = 0$, $s = 1$
   (c) $n = 39$, $f = 5$, $p = 0$, $s = 2$
   (d) $n = 7$, $f = 3$, $p = 1$, $s = 2$
   (e) $n = 224$, $f = 7$, $p = 3$, $s = 2$

   Welches Padding braucht man bei $f = 3$ und $s = 1$, damit das Bild gleich groß bleibt?

Lösung:
(a) $\frac{28+4-5}{1} + 1 = 28$
(b) $\frac{32+0-3}{1} + 1 = 30$
(c) $\frac{39+0-5}{2} + 1 = 18$
(d) $\frac{7+2-3}{2} + 1 = 4$
(e) $\frac{224+6-7}{2} + 1 = 112$
$p = 1$, allgemein $p = \frac{f-1}{2}$.

4. (2 Punkte) Berechne das Ergebnis von Max-Pooling mit Größe 2 und Stride 2 für

$$\begin{pmatrix} 1 & 4 & 2 & 0 \\ 3 & 2 & 1 & 5 \\ 0 & 1 & 7 & 2 \\ 6 & 2 & 3 & 1 \end{pmatrix}$$

   Wie viele Parameter hat eine Pooling-Schicht?

Lösung:

$$\begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

Keine, sie nimmt nur das Maximum jedes 2×2-Blocks.

5. (3 Punkte) Eine Faltungsschicht `nn.Conv2d(3, 10, kernel_size=3)` macht aus einem Farbbild mit 32×32 Pixeln 10 Feature Maps mit 30×30 Pixeln.
   (a) Wie viele Parameter hat die Schicht?
   (b) Eine vollständig verbundene Schicht `nn.Linear` soll aus denselben Inputs dieselbe Zahl an Outputs berechnen. Wie viele Parameter hätte sie?
   (c) Erkläre den Unterschied mit den Begriffen Parameter Sharing und lokale Verbindungen.

Lösung:
(a) $(3 \cdot 3 \cdot 3 + 1) \cdot 10 = 280$
(b) Inputs $32 \cdot 32 \cdot 3 = 3072$, Outputs $30 \cdot 30 \cdot 10 = 9000$, also $3072 \cdot 9000 + 9000 = 27657000$
(c) Parameter Sharing: Jeder Filter wird an allen Stellen des Bildes mit denselben Gewichten benutzt. Lokale Verbindungen: Jeder Output hängt nur von $3 \cdot 3 \cdot 3$ Inputs ab, nicht von allen 3072.

6. (4 Punkte) Ein Batch mit der Gestalt `(20, 3, 39, 39)` geht durch drei Faltungsschichten ohne Padding:

```
conv1 = nn.Conv2d(3, 10, kernel_size=3)
conv2 = nn.Conv2d(10, 20, kernel_size=5, stride=2)
conv3 = nn.Conv2d(20, 40, kernel_size=5, stride=2)
```

   (a) Welche Gestalt hat der Tensor nach jeder Schicht?

   (b) Wie viele Parameter hat jede Schicht?

Lösung:

(a) conv1: `(20, 10, 37, 37)`, conv2: `(20, 20, 17, 17)`, conv3: `(20, 40, 7, 7)`

(b) conv1: $(3 \cdot 3 \cdot 3 + 1) \cdot 10 = 280$, conv2: $(5 \cdot 5 \cdot 10 + 1) \cdot 20 = 5020$, conv3: $(5 \cdot 5 \cdot 20 + 1) \cdot 40 = 20040$

7. (4 Punkte) Gegeben ist das folgende CNN für Fashion-MNIST.

```
class Netz(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 8, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(8, 16, kernel_size=3, padding=1)
        self.fc = nn.Linear(16 * 7 * 7, 10)

    def forward(self, t):
        t = F.max_pool2d(torch.relu(self.conv1(t)), 2)
        t = F.max_pool2d(torch.relu(self.conv2(t)), 2)
        return self.fc(t.reshape(-1, 16 * 7 * 7))
```

   (a) Welche Gestalt hat ein Batch von 50 Fashion-MNIST-Bildern `(50, 1, 28, 28)` nach jeder Zeile von `forward`?

   (b) Wie viele Parameter hat das Netz?

Lösung:

(a) nach der ersten Zeile `(50, 8, 14, 14)`, nach der zweiten `(50, 16, 7, 7)`, Ausgabe `(50, 10)`. Die Faltungen ändern wegen Padding 1 die Größe nicht, das Pooling halbiert sie.

(b) conv1: $(3 \cdot 3 \cdot 1 + 1) \cdot 8 = 80$, conv2: $(3 \cdot 3 \cdot 8 + 1) \cdot 16 = 1168$, fc: $784 \cdot 10 + 10 = 7850$, zusammen 9098.

8. (4 Punkte) (a) Warum eignen sich vollständig verbundene Schichten schlecht für große Bilder?
   (b) Was lernen die Filter in den ersten Schichten eines CNN typischerweise, was in späteren?
   (c) Welches Problem lösen die Skip Connections von ResNet?
   (d) Was ist Transfer Learning und wann ist es besonders nützlich?

Lösung:

(a) Jeder Pixel ist mit jedem Neuron verbunden, das ergibt sehr viele Parameter. Außerdem nutzen sie nicht aus, dass benachbarte Pixel zusammengehören, und müssen jedes Merkmal an jeder Stelle des Bildes neu lernen.

(b) Frühe Schichten lernen einfache Merkmale wie Kanten in verschiedenen Richtungen, spätere setzen daraus kompliziertere Formen und Teile von Objekten zusammen.

(c) In sehr tiefen Netzen werden die Fehlersignale auf dem Weg zurück immer kleiner, die vorderen Schichten lernen kaum. Bei $a = f(x) + x$ kommt das Fehlersignal über den Summanden x ungeschwächt zurück.

(d) Man nimmt ein Netz, das auf einer großen Datenmenge trainiert wurde, behält seine Schichten und trainiert nur eine neue letzte Schicht für die eigenen Klassen. Es ist besonders nützlich, wenn man nur wenige eigene Beispiele hat.

9. (4 Punkte) Ändere in Colab das CNN aus dem Notebook so, dass es zwei Faltungsschichten mit 3×3-Filtern und Padding 1 hat (16 und 32 Filter), jeweils mit ReLU und Max-Pooling, und danach eine vollständig verbundene Schicht. Gib die Zahl der Parameter aus, prüfe die Ausgabegestalt mit `torch.randn(5, 1, 28, 28)` und trainiere es drei Epochen wie im Notebook. Vergleiche mit dem CNN aus dem Notebook.

Lösung:

```
class CNN3(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.fc = nn.Linear(32 * 7 * 7, 10)

    def forward(self, t):
        t = F.max_pool2d(torch.relu(self.conv1(t)), 2)
        t = F.max_pool2d(torch.relu(self.conv2(t)), 2)
        return self.fc(t.reshape(-1, 32 * 7 * 7))

cnn3 = CNN3()
print(sum(p.numel() for p in cnn3.parameters()))
print(cnn3(torch.randn(5, 1, 28, 28)).shape)
```

Ergebnis: 20490 Parameter (das CNN im Notebook hat 28938), Gestalt `torch.Size([5, 10])`. Training mit `trainiere` und `genauigkeit` aus dem Notebook. Die Genauigkeit auf den Testdaten liegt bei uns nach drei Epochen bei 0.883, also fast genauso hoch wie mit 5×5-Filtern (0.888), mit weniger Parametern.