מבנה שכבת קונבולוציה ברשתות נוירונים

את פעולת הקונבולוציה אשר הכרנו בפרק הקודם נרתום כעת עבור רשתות נוירונים בעלות קלט ויזואלי. הצעד הראשון למטרה זו הוא הפיכת גרעין הקונבולוציה **לפרמטרים נלמדים** של רשת הנוירונים: בעוד שלמטרות עיבוד תמונה אנו מגדירים את גרעין הקונבולוציה מראש עבור שימוש ידוע, כאשר נשלב פעולה זו בלמידה עמוקה נניח לערכי מטריצת הגרעין להיקבע לפי אלגוריתם האופטימיזציה. משמעות הדבר היא שהרשת תלמד בעצמה גרעינים המחלצים מאפיינים רלוונטיים למטרותיה. אם כן, נתחיל כמנהגנו בהגדרה ראשונית של שכבת קונבולוציה, אותה נמשיך ונשפר בהמשך היחידה.

נניח כי הקלט לשכבה הוא minibatch בעל N דגימות של תמונות בגודל $H\times W$, להן ערוץ צבע יחיד. בהתאם גודל טנזור הקלט יהיה $N\times H\times W$ כאשר הקונבולוציה תופעל בנפרד על כל תמונה ב-minibatch.

```
class ConvLayer_1(nn.Module):
    def __init__(self, kernel_size):
        super().__init__()
        self.kern = nn.Parameter(torch.rand(kernel_size))
        self.p , self.q = kernel_size
    def forward(self, X):
      output = torch.empty(X.size(0),
                           X.size(1)-self.p+1,
                           X.size(2)-self.q+1)
      for i in range(output.size(1)):
        for j in range(output.size(2)):
          sub_img=X[:,i:(i+self.p),j:(j+self.q)]
          output[:,i,j]=(sub_img*self.kern).sum(dim=(1,2))
      return output
```

במימוש זה בקוד מופיעים מספר פרטים מעניינים:

- ראשית שימו לב שגרעין הקונבולוציה מאותחל באקראי ומוגדר כאובייקט `nn.Parameter` שכן הוא ישתתף בתהליך האימון ובעת ביצוע צעד של אובייקט האופטימיזציה נרצה שגם הוא יעודכן.
- שנית, ראו שבעת חישוב האיבר ה-`i,j` של הפלט, הגרעין משודר לאורך מימד ה-batch של תת התמונה, ואחרי כן מתודת הסכום מבצעת הפחתה על מימד הגובה והרוחב של התמונה, כך שנשאר סקלר יחיד עבור כל דגימה ב-batch.
- לבסוף שימו לב לגמישות המובנית בתוך השכבה: בניגוד לשכבות ליניאריות, בנאי שכבה זו לא דורש את מימד הקלט והפלט מראש. בעת המעבר קדימה בשכבה ייקבע מימד הפלט לפי חוקיות הקונבולוציה. משמעות הדבר היא שניתן להשתמש בשכבה מסוג זה עבור קלט המשתנה בגודלו.

הפרט המעניין ביותר בשכבת הקונבולוציה הוא שבעת המעבר קדימה אנו מבצעים רק פעולות כפל וחיבור בין הגרעין לבין איברי מטריצת הקלט ולכן בעת פעפוע הגרדיאנט לאחור לא ניתקל בבעיה - הרי אלו פעולות גזירות (הגם שמערכת ה-Autograd תטפל בזאת עבורנו).

נדגים זאת על ידי אתחול שכבת קונבולוציה ואימונה באמצעות SGD לזהות קצוות אנכיים: נגדיר פילטר זיהוי קצוות בעל הגרעין

$$K=\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix}$$

בעזרת המחלקה שהגדרנו זה עתה:

```
edge_detector=ConvLayer_1((3,3))
edge_detector.kern.requires_grad=False
edge_detector.kern[:]=torch.tensor([[-1.,-1,-1],
                                    [0,0,0],
                                    [1,1,1]])
```

ראו כי כיבינו את מעכב מערכת ה-Autograd על גרעין מזהה הקצוות, שכן הוא יהווה את המטרה אליה שכבת הקונבולוציה שנגדיר להלן תתאים את עצמה לאורך תהליך האופטימיזציה.

```
my_conv = ConvLayer_1((3,3));
optimizer = torch.optim.SGD(my_conv.parameters(), lr=0.1)
```

כל שנשאר הוא לכתוב את לולאת האימון, בה נטען minibatch של תמונות מהאוסף Fashion-MNIST, ננרמל אותן למספרים ממשים בטווח 0 עד 1 ונעביר אותן דרך פילטר זיהוי הקצוות ושכבת הקונבולוציה שלנו. המרחק בין שתי התוצאות ישמש כפונקציית ההפסד של האופטימיזציה וכך גרעין השכבה יעודכן לאורך התהליך לייצר פלט דומה לזה של `edge_detector`. לאחר טעינת התמונות לתוך המשתנה `imgs`, והמרתן לטנזורים מנורמלים, ליבת לולאת האימון תיראה כך:

```
optimizer.zero_grad()
my_transformed_imgs = my_conv(imgs)
target_imgs = edge_detector(imgs)
loss = ((my_transformed_imgs-target_imgs)**2).mean()
loss.backward()
optimizer.step()
```

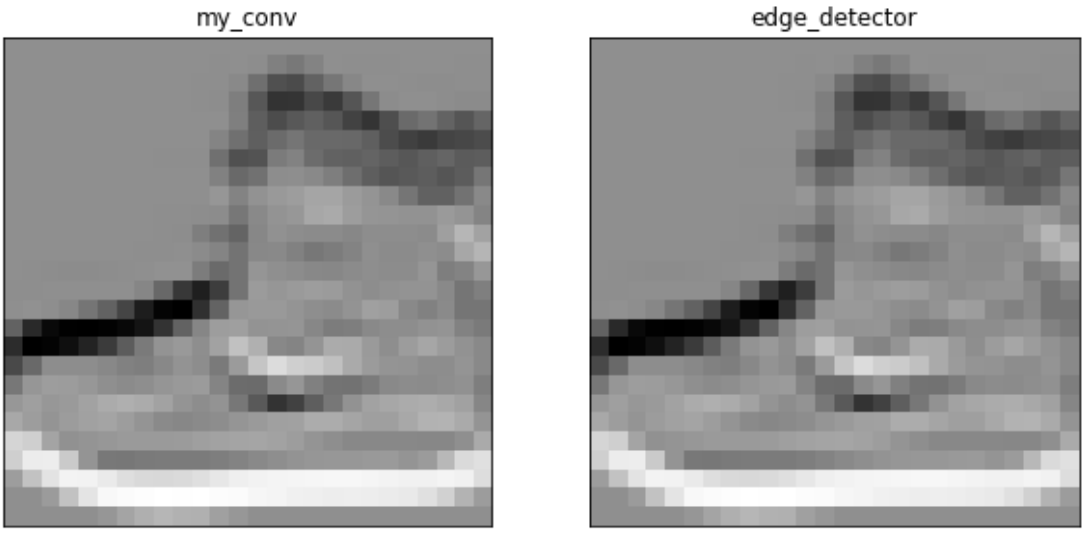
Epoch יחיד של אימון מניב את גרעין הקונבולוציה הבא, וניתן לראות בו את הדימיון לגרעין מזהה הקצוות.

```
print(my_conv.kern)
```

פלט:

```
Parameter containing:
tensor([[-0.8108, -0.7242, -0.6472],
        [-0.0661,  0.0861, -0.0626],
        [ 0.5860,  1.0309,  0.5985]], requires_grad=True)
```

באיור הבא נראה שהפעלת הפילטרים על אותה תמונה מניבה תוצאה כמעט זהה. אם כן, תהליך הלמידה הסתיים בהצלחה.



כבר בשלב זה אנו יכולים להבין את היתרונות של שכבות אלו עבור קלט ויזואלי:

- ראשית, מספר הפרמטרים עבור שכבה נתונה קטן משמעותית מאשר בשכבה בעלת קישוריות מלאה – יש ללמוד את הגרעין בלבד.
- שנית, רק מספר מועט של פיקסלים הממוקמים קרוב זה לזה בתמונת הקלט משפיעים על חישוב פיקסל של הפלט. כך אנו כופים על הרשת ללמוד כללים הלוקחים בחשבון אינטראקציה מקומית בלבד, כללים אשר לצופה האנושי ברור שקיימים במידע ויזואלי: קיומו של שרוול חולצה במקום מסויים בתמונה, למשל, מעיד על כך שזו תמונה של חולצה.
- בנוסף, השימוש החוזר באותו גרעין עבור תתי מטריצות שונות של תמונת הקלט מוביל את הרשת ללמוד כללים שאינם תלויים במיקום ספציפי במרחב (אין זה משנה היכן בתמונה נמצא השרוול הנ"ל).
- לבסוף, פעולת הקונבולוציה זולה משמעותית מחישוב מעבר קדימה בשכבה ליניארית מלאה, שוב עקב כך שכל פיקסל בפלט מחושב על בסיס מספר מעט של מספרים.

שאלות לתרגול

1. תמונות הקלט בצורתן המקורית מכילות ערכים בטווח 0 עד 255. נסו לאמן את שכבת הקונבולוציה ללמוד את גרעין זיהוי הקצוות ללא נרמול לטווח 0 עד 1. אם האימון נכשל, האם תוכלו להסביר למה?
2. אסטרטגיית אתחול פרמטרים מוצלחת תיקח בחשבון את גודל הגרעין. שנו את בנאי שכבת הקונבולוציה כך שאיברי הגרעין יאותחלו על ידי דגימה מהתפלגות $U\left[-\frac{1}{\sqrt{K}},\frac{1}{\sqrt{K}}\right]$ כאשר K הוא מספר האיברים בגרעין. **רמז:** אם $u \sim U[0,1]$ אז $au+b \sim U[b,a+b]$.
3. חשבו באופן אנליטי את הפעפוע לאחור של השגיאה דרך שכבת קונבולוציה בעלת גרעין בגודל 1X2 כאשר ידוע שתמונת הקלט היא בגודל 2X3. כרגיל, הניחו שגרדיאנט המחיר ביחס לפלט השכבה, $\frac{\partial C}{\partial Y}$, חושב זה מכבר, ועליכם לחשב רק:
   - את הגרדיאנט של הפלט ביחס לפרמטרים (גרעין הקונבולוציה), $\frac{\partial Y}{\partial K}$, לצורך עדכונם בעת האופטימיזציה,
   - ואת הגרדיאנט ביחס לקלט השכבה, $\frac{\partial Y}{\partial X}$, לצורך המשך פעפוע השגיאה.

   רמז: הפרידו למקרים לפי מספר הפעמים שפיקסל מסויים של הקלט משתתף בחישוב פיקסל בפלט.