רכיבים נוספים של שכבות קונבולוציה

לאחר שהוספנו ערוצי פלט וקלט מרובים לשכבות הקונבולוציה בפרק הקודם, נסקור כעת את הרכיבים האחרונים הדרושים בכדי להשתמש בהן במשימות למידה.

ריפוד (Padding)

בפרקים הקודמים נתקלנו בתכונה יסודית של פעולת הקונבולוציה: גודל תמונת הפלט קטן מתמונת הקלט במימדי האורך והרוחב וזאת בהתאם לגודל גרעין הקונבולוציה. לעתים הקטנת מימד זו אינה רצויה, ובכדי לשלוט על גודל הפלט אפשר להגדיל באופן מלאכותי את הקלט של השכבה על ידי הוספת שורות ועמודות בקצוות התמונה. שיטה זו נקראת ריפוד (padding) וישנם מספר מימושים שלה ב-PyTorch, בהתאם לערכים אשר מוצבים בעמודות/שורות החדשות כאשר הבחירה הנפוצה ביותר היא ריפוד עם אפסים. ראו להלן כיצד אנו מרפדים טנזור קלט לדוגמה בשורת ועמודת אפסים אחת נוספת מכל צד.

```
pad_layer = nn.ZeroPad2d(1)
A = torch.arange(1,10).reshape(3,3)
print(A,pad_layer(A),sep='\n')
```

פלט:

```
tensor([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
tensor([[0, 0, 0, 0, 0],
        [0, 1, 2, 3, 0],
        [0, 4, 5, 6, 0],
        [0, 7, 8, 9, 0],
        [0, 0, 0, 0, 0]])
```

טרנספורמציה זו ממומשת גם בתוך שכבת הקונבולוציה הסטנדרטית של PyTorch: בשורת הקוד הבאה אנו מגדירים שכבת קונבולוציה המקבלת minibatch של תמונות קלט בעלות ערוץ מידע יחיד, טנזור בגודל $N\times1\times H\times W$, ומחזירה פלט בעל אותו גודל בדיוק. בהתאם לגודל הגרעין, השכבה בוחרת באופן אוטומטי את מספר שורות ועמודות הריפוד הדרושות.

```
conv_layer = nn.Conv2d(in_channels=1, out_channels=1,
                       kernel_size=(3, 5), padding="same")
```

פרמטר ה-bias

לשכבת הקונבולוציה המובנית ב-Pytorch שני רכיבים נוספים שעוד לא דנו בהם. בראשון ניתקל תוך התבוננות ברשימת הפרמטרים בשכבה שזה עתה הגדרנו.

```
for params in conv_layer.named_parameters():
  print(params)
```

פלט:

```
('weight', Parameter containing:
tensor([[[[-0.2093, -0.2208,  0.0425,  0.1462, -0.1257],
          [ 0.0535, -0.0787, -0.0518,  0.1329,  0.2383],
          [ 0.0557,  0.1856, -0.1075, -0.2001,  0.1463]]]],
requires_grad=True))
('bias', Parameter containing:
tensor([-0.2201], requires_grad=True))
```

ראו שהפרמטר `weight` מכיל את גרעין הקונבולוציה מגודל 3X5 (עם שני מימדים מנוונים נוספים – עבור ערוצי הקלט והפלט), כצפוי. בנוסף יש לשכבה פרמטר נוסף, `bias`, המכיל ערך נפרד לכל

ערוץ פלט (אחד במקרה שלנו), ואותו מוסיפים לתוצאת הקונבולוציה בערוץ זה. הצורך בערך זה נובע ממבנה הרשת בה שכבת הקונבולוציה תמצא שימוש: לרוב, פלט השכבה יעבור באופן מיידי לאקטיבציה לא ליניארית כגון ReLU, עבורה ה-bias דרוש על מנת לווסת את הסף בו הנוירון מופעל. אין אנו מחברים שכבות קונבולוציה אחת לשניה ללא אקטיבציה לא ליניארית ביניהן, שהרי הקונבולוציה בבסיסה היא פעולה ליניארית, ועל כן שרשור קונבולוציות אחת לאחר השניה שקול לביצוע פעולה ליניארית יחידה.

גודל הפסיעה (Stride)

הרכיב הנוסף השני של שכבות קונבולוציה, בו נדון כעת, הוא פרמטר גודל הפסיעה. בעוד שבריפוד הקלט השתמשנו למטרת הגדלת הפלט, לעתים נרצה דווקא את התוצאה ההפוכה – לצמצם את גודל התמונה לאחר המעבר בשכבה, למשל כאשר גודל הקלט גדול מדי לאימון הרשת ביעילות, או כאשר אנו מעוניינים לקבל לאחר מספר מועט של שכבות מאפיינים היוצרים קשר בין אזורים מרוחקים בתמונת הקלט.

אם בעבר עברנו על **כל פיקסל** בקלט (אשר מימינו ומתחתיו ישנם מספיק איברים כגודל הגרעין) וחתכנו מהקלט מטריצה בגודל הגרעין לטובת חישוב פיקסל יחיד בפלט, כעת נעשה זאת רק עבור חלק מהפיקסלים לפי חוקיות פשוטה, אותה נבין מהדוגמה הבאה:

```
downsample = nn.Conv2d(in_channels=1, out_channels=1,
                       bias=False, kernel_size=(1, 1),
                       stride=(2,3))
downsample.weight=nn.Parameter(
    torch.tensor(1.).reshape(1,1,1,1))
A=torch.arange(25,dtype=torch.float).reshape(1,1,5,5)
print(A,downsample(A),sep='\n')
```

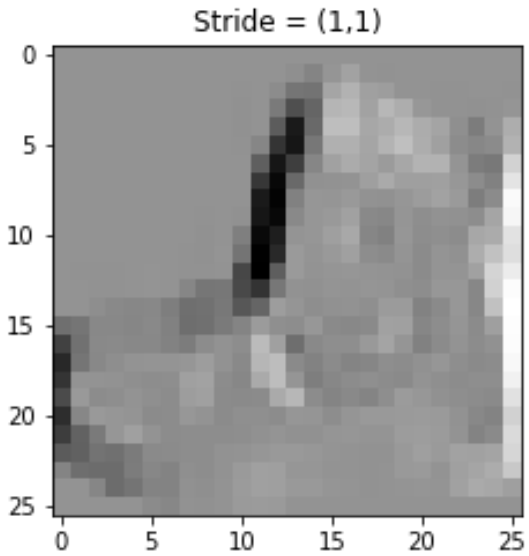
פלט:

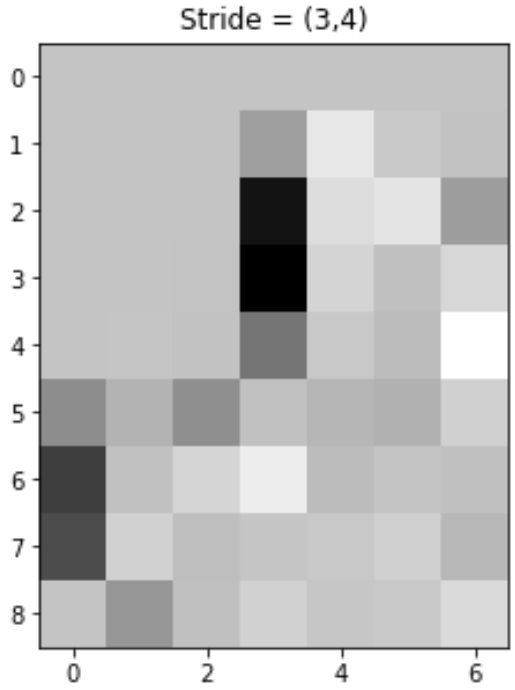
```
tensor([[[[ 0.,  1.,  2.,  3.,  4.],
          [ 5.,  6.,  7.,  8.,  9.],
          [10., 11., 12., 13., 14.],
          [15., 16., 17., 18., 19.],
          [20., 21., 22., 23., 24.]]]])
tensor([[[[ 0.,  3.],
          [10., 13.],
          [20., 23.]]]], grad_fn=<ThnnConv2DBackward0>)
```

בדוגמה זו הגדרנו שכבת קונבולוציה ללא bias עם גרעין בגודל 1X1, ואת ערך הגרעין קבענו להיות 1. קונבולוציה עם גרעין שכזה, עבור קלט בעל ערוץ יחיד אמורה להניב את פונקציית הזהות, אך שימו לב שהעברנו לבנאי השכבה גם פרמטר `stride`, זהו גודל הפסיעה. משמעות הזוג הסדור שהועבר בפרמטר זה היא שבעת חישוב הקונבולוציה יש לחשב ולשמור את ערך מכפלת תת המטריצה בגרעין עבור אחד מכל שלושה איברים בשורה (כל עוד ניתן לבצע חישוב זה, כאמור), ואחת מכל שתי שורות. התוצאה המתקבלת עבור הגרעין המסויים שבחרנו היא שרק האיברים מהעמודה הראשונה או השלישית הנמצאים בשורות האי זוגיות נשמרים בפלט.

כמובן שניתן לשלב את פרמטר הפסיעה עם כל אחד מהרכיבים האחרים של שכבת הקונבולוציה. למשל, נגדיר מחדש את מזהה הקצוות האופקיים ששימש כדוגמה בפרק הקודם עם פסיעות של שורה אחת מתוך כל שלוש ועמודה אחת מתוך כל ארבע בקטע הקוד הבא, ונשווה את תוצאות הפילטר החדש למקורי באיור העוקב.

```
edge_detector2 = nn.Conv2d(in_channels=1, out_channels=1,
                        bias=False, kernel_size=(3, 3),
                        stride=(3,4))
edge_detector2.weight=nn.Parameter(torch.tensor(
    [[-1.,0,1],[-1,0,1],[-1,0,1]]).reshape(1,1,3,3))
```





איגום (Pooling)

שיטה נוספת לצמצום מימד הפלט היא העברת הפלט של שכבת הקונבולוציה (לאחר אקטיבציה) לשכבת איגום (pooling layer). שכבה זו תשקלל את הנתונים של מספר פיקסלים קרובים זה לזה, למשל על ידי חישוב הממוצע שלהם, ליצירת מאפיין יחיד בפלט. התוצאה שתתקבל היא מאפיינים אשר אינם רגישים לתזוזות קטנות של הקלט, דבר הרצוי לרוב: מעניין אותנו האם קיימים בתמונת הקלט שרוכי נעליים או לא, ולאו דווקא המקום המדויק בו הם נמצאים.

ראו בדוגמה הבאה כיצד אנו מגדירים שכבת איגום אשר מחלקת את הקלט לתת-מטריצות **זרות** בגודל 2X3 ומחזירה מאפיין **יחיד** לכל תת מטריצה – הערך המקסימלי בה.

```
pool= nn.MaxPool2d(kernel_size=(2,3))
A=torch.arange(40,dtype=torch.float).reshape(1,1,5,8)
print(A,pool(A),sep='\n')
```

פלט:

```
tensor([[[[ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11., 12., 13., 14., 15.],
          [16., 17., 18., 19., 20., 21., 22., 23.],
          [24., 25., 26., 27., 28., 29., 30., 31.],
          [32., 33., 34., 35., 36., 37., 38., 39.]]]])
tensor([[[[10., 13.],
          [26., 29.]]]]
```

בדוגמה זו שתי העמודות האחרונות והשורה האחרונה בקלט "אבדו" – הן לא השתתפו בתהליך האיגום שכן מימדי הקלט לא תואמים בדיוק את גודל המטריצה המשמשת לאיגום. כדי להימנע מאובדן זה ניתן להעביר לשכבה פרמטר הדורש להשתמש בכל המאפיינים בקלט, במחיר של מטריצת איגום קטנה יותר בקצוות. למשל בקטע הקוד הבא, המאפיין השלישי בשורה הראשונה מחושב כערך המקסימלי של מטריצה בגודל 2X2.

```
pool= nn.MaxPool2d(kernel_size=(2,3),ceil_mode=True)
print(A,pool(A),sep='\n')
```

פלט:

```
tensor([[[[ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.],
          [ 8.,  9., 10., 11., 12., 13., 14., 15.],
          [16., 17., 18., 19., 20., 21., 22., 23.],
          [24., 25., 26., 27., 28., 29., 30., 31.],
          [32., 33., 34., 35., 36., 37., 38., 39.]]]])
tensor([[[[10., 13., 15.],
          [26., 29., 31.],
          [34., 37., 39.]]]])
```

בדומה לשכבת קונבולוציה, גם לשכבת איגום יש פרמטר `stride`, המאפשר לקבוע ברזולוציה גבוהה את החפיפה בין תתי המטריצות אשר ישמשו ליצירת מאפייני הפלט. ברירת המחדל המייצרת מטריצות זרות היא `stride=kernel_size`. לבסוף נציין כי עבור קלט רב ערוצי שכבת האיגום מחשבת פלט לכל ערוץ בנפרד.

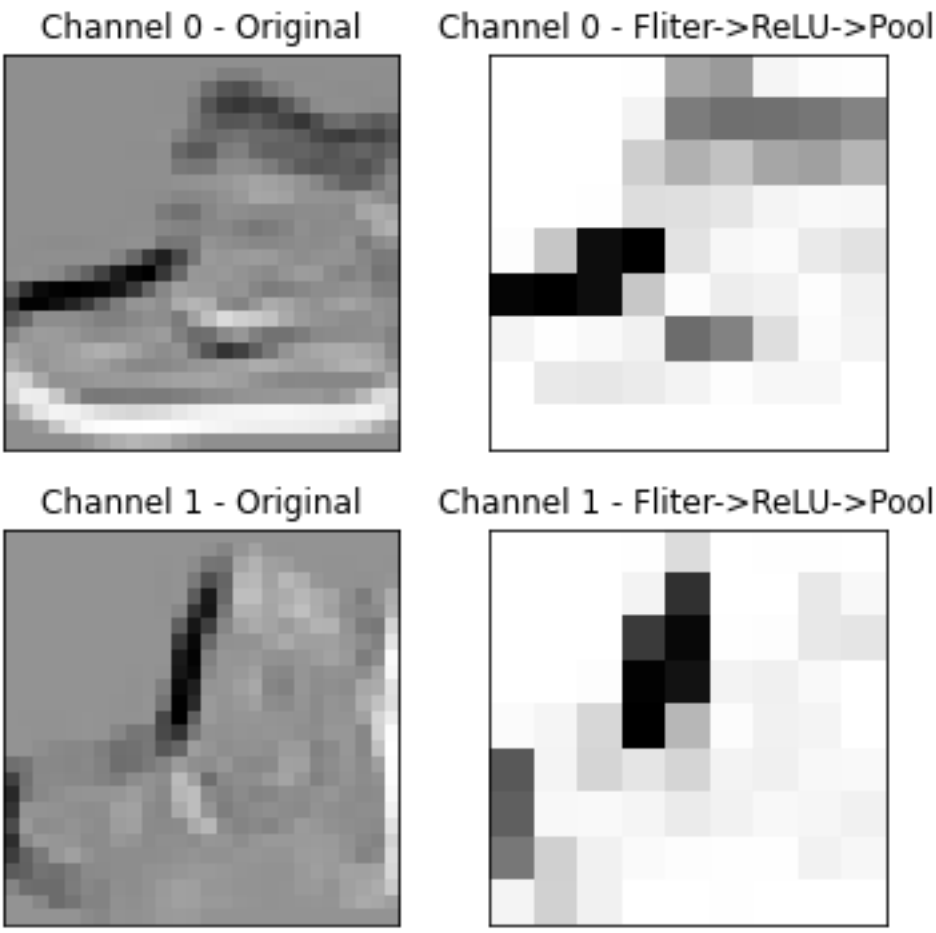
לסיכום, ראו בקטע הקוד הבא כיצד אנו מגדירים בלוק של שלוש שכבות המשרשר את פעולת הקונבולוציה, האקטיבציה והאיגום. זהו הרצף בו נשתמש לרוב בתוך רשת נוירונים, עם הציפייה שהרשת תלמד בתהליך האימון את גרעין הקונבולוציה ופרמטר ה-bias. בדוגמה זו אנו מגדירים באופן ידני את הגרעין, מזהה הקצוות האופקיים והאנכיים בו השתמשנו בעבר.

```
edge_detector2 = nn.Conv2d(in_channels=1, out_channels=2,
                           bias=False, kernel_size=(3, 3),
                           stride=(1,1))
edge_detector2.weight=nn.Parameter(torch.tensor(
    [[[-1.,-1,-1],
      [0,0,0],
      [1,1,1]],

     [[-1,0,1],
      [-1,0,1],
      [-1,0,1]]]).reshape(2,1,3,3))

block = nn.Sequential(edge_detector2,
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=(3,3),
                                   ceil_mode=True))
```

פלט הבלוק (בגודל 9X9 פיקסלים) עבור תמונה אחת מאוייר להלן, לצד הפלט המקורי של מזהה הקצוות לבדו.



שאלות לתרגול

1. עבור גרעין קונבולוציה בגודל $p \times q$, בכמה שורות ועמודות ריפוד יש להשתמש כדי לשמור על מימדיה המקוריים של תמונת הקלט לאחר הפעלת הקונבולוציה?
2.
   א. הגדירו שכבת קונבולוציה עם גרעין בגודל 7×5 ללא פרמטר bias. בשכבה יהיה ערוץ קלט יחיד וערוץ פלט יחיד.
   ב. בחרו תמונת קלט אקראית מהאוסף Fashion-MNIST והעבירו אותה דרך השכבה.
   ג. הוסיפו לתמונת הקלט שורות ועמודות אפסים מלמעלה ומשמאל בלבד, כך שהפלט יהיה בעל אותו מימד כמו הקלט. והעבירו את התוצאה דרך שכבת הקונבולוציה.
   ד. האם תוכלו לשער מדוע נהוג למקם את האפסים הנוספים מסביב לתמונת המקור, ולא רק בצד אחד?
3. הציגו את הקונבולוציה של תמונת קלט X בגודל $1 \times 5 \times 5$ (בעלת ערוץ יחיד) עם גרעין בגודל 2×2 כהעתקה ליניארית.

   הנחיות:
   - מצאו את מימד פלט הקונבולוציה ראשית.
   - שטחו את תמונת הקלט והציגו אותה כוקטור X בגודל $HW \times 1$.
   - כעת מצאו מטריצה A כך ש- AX היא תוצאת הקונבולוציה, גם היא לאחר שיטוח.
     - מצאו ראשית את מימדי A בעזרת ההנחיות הקודמות.
     - רוב ערכיה של A יהיו אפסים.
4. העבירו לבנאי של `nn.Conv2d` בו זמנית פרמטר גודל פסיעה גדול מ-1 ואת הפרמטר `padding="same"`. נסו להסביר מדוע מתקבלת שגיאה.

5. צרו שכבת קונבולוציה בעזרת המחלקה `nn.Conv2d`, והגדירו את הגרעין שלה באופן ידני כך שהיא תבצע פעולה זהה לשכבת איגום הנוצרת כך:

```
pool=nn.AvgPool2d(kernel_size=(2,2))
```

הערה: שימו לב ששכבת איגום זה מחשבת את הממוצע של כל תת מטריצה 2X2.