מבנה שכבת קונבולוציה: ריבוי ערוצים

לשכבת הקונבולוציה אשר כתבנו בפרק הקודם יש להוסיף מספר רכיבים לפני שנוכל להשתמש בהן למשימות למידה מורכבות. בפרק זה נתחיל לסקור רכיבים אלו, ונשלבם בהגדרת השכבה.

ריבוי ערוצי פלט

שכבת הקונבולוציה שלנו בשלב זה תלמד גרעין יחיד בהנתן תמונת הקלט. עם זאת, ייתכן שישנם מספר מאפיינים חשובים שכדאי לרשת לחלץ מהתמונה לצורך המשך הלמידה (זיהוי קצוות אופקיים, זיהוי קצוות אנכיים, זיהוי פינות וכו'). על כן, נרצה לאפשר לשכבה אחת ללמוד **מספר גרעינים שונים** באופן בלתי תלוי זה בזה – כל אחד מהם יחשב קונבולוציה נפרדת של הקלט והפלט שלה יישמר **בערוץ פלט נפרד**. אם כן, עבור טנזור קלט של בעל הגודל $N\times H\times W$ (minibatch של N תמונות בגודל $H\times W$), פלט השכבה יהיה טנזור בעל המימדים $N\times C\times(H-p+1)\times(W-q+1)$ כאשר המימד השני יכיל את תוצאות הקונבולוציות של התמונות עם הגרעינים השונים, כולם ממימד $p\times q$. קוד השכבה החדש הוא

```
class ConvLayer_2(nn.Module):
    def __init__(self, out_channels=1, kernel_size=(1,1)):
        super().__init__()
        self.kern = nn.Parameter(
            torch.rand((out_channels,*kernel_size)))
        self.p , self.q = kernel_size
        self.out_channels=out_channels
    def forward(self, X):
      output = torch.empty(X.size(0),
                           self.out_channels,
                           X.size(1)-self.p+1,
                           X.size(2)-self.q+1)
      for i in range(output.size(2)):
        for j in range(output.size(3)):
          sub_img=X[:,i:(i+self.p),j:(j+self.q)]
          sub_img=sub_img.unsqueeze(1)
          output[:,:,i,j]=(sub_img*self.kern).sum(dim=(2,3))
      return output
```

כאן יש לשים לב למספר פרטים חדשים:

- ראשית, בעת אתחול השכבה יש לציין את מספר ערוצי הפלט המבוקשים, פרט זה הכרחי שכן בעת יצירת אובייקט ממחלקה זו יש לאתחל מספר גרעינים שונים השווה לפרמטר זה.
- שנית, בעת דגימת המ"מ לאתחול גרעיני השכבה, אנו "פותחים" את הזוג הסדור `kernel_size` לשני ערכים נפרדים בעזרת אופרטור ה-`*` ומיד מאגדים אותם לשלשה סדורה עם הפרמטר `out_channels`. אנו עושים זאת שכן גודל הטנזור הנדגם מועבר ל-`torch.rand` בתוך ח-יה סדורה יחידה.
- לבסוף, חישוב כל פיקסל בכל ערוצי הפלט מתבצע בשורת הקוד הקודמת לסוף וזאת על ידי שידור תת התמונה `sub_img` לאורך מימד הערוצים ושידור הגרעינים לאורך מימד ה-batch. מכיוון שלתת התמונה עוד אין מימד המתאים לערוצי הפלט החדשים (גודלה הוא $N\times p\times q$), קודם לשידור אנו מכניסים מימד מנוון לאחר המימד הראשון, בעזרת המתודה `unsqueeze(1)` (גודל תת התמונה אחרי שימוש במתודה זו הוא $N\times 1\times p\times q$). ללא הוספת מימד מלאכותי זה נקבל תוצאה שגויה: חוקיות השידור תשווה את מימד הערוץ של

הגרעינים (טנזור בגודל $C\times p\times q$) למימד ה-batch של תת התמונה, דבר אשר יוביל לרוב לשגיאה, ובמקרים נדירים לשילוב לא רצוי של רכיבים ממימדים שונים.

לצורך הדגמה ניצור שכבת קונבולוציה בעלת שני ערוצי פלט, נגדיר במפורש את הגרעינים שלה ונפעילה על קלט לדוגמה.

```
edge_detector2=ConvLayer_2(2,(3,3))
filters=torch.tensor(
    [[[-1.,-1,-1],
      [0,0,0],
      [1,1,1]],

     [[-1,0,1],
      [-1,0,1],
      [-1,0,1]]])
edge_detector2.kern.requires_grad=False
edge_detector2.kern[:]=filters
```

בקטע קוד זה הגדרנו את גרעיני הקונבולוציה של השכבה להיות

$$\text{kern}[0,\ldots]=\begin{pmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{pmatrix},\quad \text{kern}[1,\ldots]=\begin{pmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{pmatrix}$$

כך שהערוץ הראשון יזהה קצוות אופקיים והשני קצוות אנכיים.

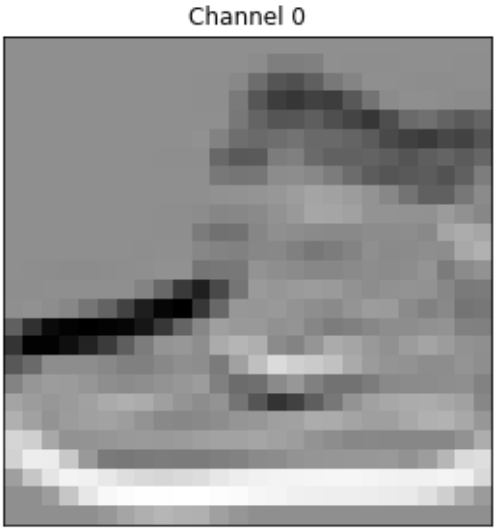
בהנחה שלתוך המשתנה `imgs` טענו minibatch בגודל 512 של תמונות, הפעלת השכבה עליו תניב את הטנזור הבא:

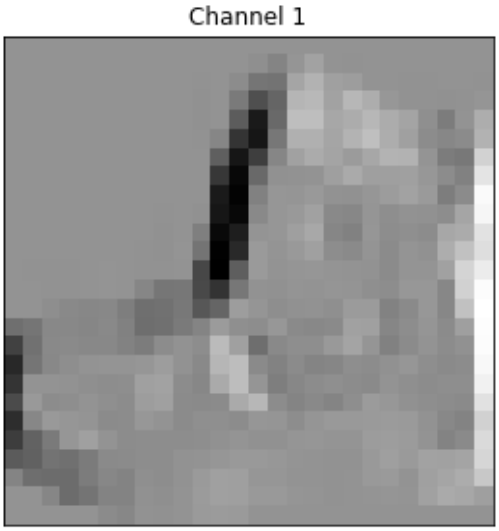
```
edges_detected = edge_detector2(imgs)
print(edges_detected.size())
```

פלט:

```
torch.Size([512, 2, 26, 26])
```

טנזור זה מכיל את תוצאת שתי הקונבולוציות עבור כל תמונה ב-minibatch. באיור הבא ניתן לראות את שני הערוצים עבור התמונה הראשונה:





ריבוי ערוצי קלט

מטרתנו היא להשתמש בשכבות קונבולוציה ברשת עמוקה, בה פלט שכבה אחת מועבר לשכבה הבאה בתור. בהתאם, עלינו לשנות את חישוב הקונבולוציה כך שריבוי הערוצים בקלט יילקח בחשבון, שכן אף אם קלט הרשת היה תמונה בעלת ערוץ צבע יחיד, פלט השכבה הראשונה כבר יכיל מספר ערוצים שונים, והחל מהשכבה השניה פעולת הקונבולוציה בה השתמשנו עד כה לא תהיה מספקת. מובן שהפונקציונליות אותה נוסיף לשכבה כעת תאפשר לנו גם להזין לרשת תמונות צבעוניות.

בעוד שאת המעבר לריבוי ערוצי הפלט עשינו על ידי הגדלת מספר הגרעינים וחישוב מספר קונבולוציות במקביל, שינוי זה אינו מתאים עבור ריבוי ערוצי הקלט. ערוצי הקלט השונים של פיקסל כלשהו בתמונה נתונה מכילים מידע רלוונטי לפיקסל זה וסביבתו: קיומה של פינה באזור מסויים בתמונה יכול לבוא לידי ביטוי בשינויים בשלושת ערוצי הצבע של תמונה צבעונית, למשל. בהתאם, נרצה עתה שפעולת הקונבולוציה תשלב את המידע הקיים בערוצי הקלט השונים, כך שעבור תמונת קלט יחידה בעלת C ערוצים, פלט פעולת הקונבולוציה (עם גרעין נתון) תהיה בעלת ערוץ יחיד.

על כן, עלינו לשנות את הגדרתה של הקונבולוציה עצמה, וכן להגדיל את מימד הגרעין:
עבור טנזור קלט X בגודל $C\times H\times W$ (המייצג תמונה בעלת C ערוצים באורך H ורוחב W), וגרעין קונבולוציה K בגודל $C\times p\times q$ (כאשר, כבעבר, p ו-q הם פרמטרים קטנים), פלט פעולת הקונבולוציה של X עם K יהיה שוב **מטריצה** Y בגודל $(H-p+1)\times(W-q+1)$ בעלת הערכים הבאים:

$$y_{r,s}=\sum_{c=1}^{C}\sum_{i=1}^{p}\sum_{j=1}^{q}x_{c,r+i-1,s+j-1}k_{c,i,j}$$

זו הכללה טבעית של הקונבולוציה עם ערוץ יחיד, שכן כדי לחשב את $y_{r,s}$ יש:

1. לחתוך מ- X תת טנזור בגודל $C\times p\times q$ אשר הפינה השמאלית העליונה שלו במימדי האורך והרוחב היא בעלת האינדקסים r,s.
2. לכפול **כל ערוץ** של תת-מטריצה זו איבר-איבר **בכל ערוץ** במטריצת הגרעין,
3. ולבסוף לסכום את התוצאה.

בקטע הקוד הבא נרחיב את הגדרת שכבת הקונבולוציה המקורית, כך שתקבל טנזור קלט בגודל $N\times C\times H\times W$: minibatch של N תמונות בעלות C ערוצים. פלט השכבה יהיה טנזור בגודל $N\times H\times W$: תוצאת הקונבולוציה עבור כל תמונה בקלט.

```
class ConvLayer_3(nn.Module):
    def __init__(self,in_channels=1, kernel_size=(1,1)):
        super().__init__()
        self.kern = nn.Parameter(
            torch.rand((in_channels,*kernel_size)))
        self.p , self.q  = kernel_size
        self.in_channels = in_channels
    def forward(self, X):
      output = torch.empty(X.size(0),
                           X.size(2)-self.p+1,
                           X.size(3)-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=(1,2,3))
      return output
```

ראו כי שכבה זו בעלת גרעין קונבולוציה יחיד (תלת מימדי), וכן שהפחתת הסכום מתבצעת על כל המימדים מלבד מימד ה-batch.

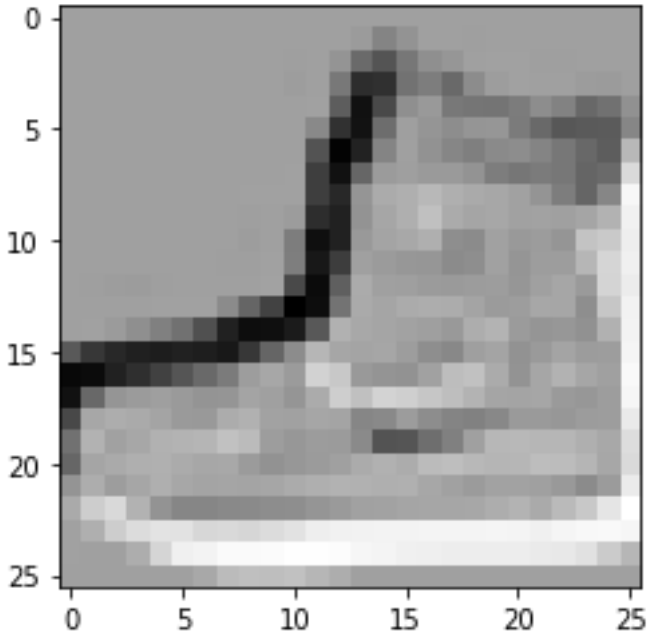
נמשיך עם הדוגמה הקודמת בה יצרנו במשתנה `edges_detected` טנזור בעל שני ערוצים: מזהה קצוות אופקי ומזהה קצוות אנכי. נשלב את המידע מהערוצים השונים בעזרת קונבולוציה עם גרעין 1X1 ושני ערוצי קלט. לאחר הפעלת גרעין זה רוחב ואורך התמונה יישארו זהים, אך בכל פיקסל נקבל צירוף ליניארי של ערוצי הקלט. נבחר למשל גרעין המבצע ממוצע של ערכי הערוצים השונים ונקבל את התוצאה הבאה.

```
aggregate=ConvLayer_3(2,(1,1))
aggregate.kern = nn.Parameter(torch.tensor(0.5).expand(2,1,1))
edges_aggregated=aggregate(edges_detected)

print(aggregate.kern.size())
print(edges_aggregated.size())
plt.imshow(edges_aggregated[0,...].detach(),cmap='Greys');
```

פלט:

```
torch.Size([2, 1, 1])
torch.Size([512, 26, 26])
```



ראו כיצד פלט השכבה מזהה קצוות אופקיים **או** אנכיים.

שאלות לתרגול

1. כתבו שכבת קונבולוציה בעלת מספר ערוצי קלט **ופלט**: הקלט לשכבה יהיה טנזור בגודל $N \times C_{in} \times H \times W$ והפלט טנזור בגודל $N \times C_{out} \times H \times W$. השכבה תחשב C_{out} קונבולוציות שונות (עם גרעינים שונים) על כל אחת מ- N תמונות הקלט.
   חתימת בנאי השכבה תהיה

```
__init__(self, in_channels=1, out_channels=1, kernel_size=(1,1))
```