5. Convolutional Neural Networks

הרשתות שתוארו עד כה הינן Fully-Connected (FC), כלומר, כל נוירון מחובר לכל הנוירונים בשכבה שלפניו ולכל הנוירונים בשכבה שאחריו. גישה זו מאוד יקרה מבחינה חישובית, והרבה פעמים אין צורך בכל הקשרים בין הנוירונים. תמונה בגווני אפור המכילה $256 \times 256 \times 1$ פיקסלים, הנכנסת לרשת FC עם $N = 1000$ קטגוריות במוצא, מכילה יותר מ-65 מיליון קשרים בין נוירונים, כאשר כל קשר הינו משקל שמתעדכן במהלך הלמידה. אם יש מספר שכבות עמוקות, המספר נהיה עצום ממש, ובלתי מעשי לתחזק כזה גודל של פרמטרים נלמדים. מלבד הבעיה של הגודל, בפועל לא תמיד צריך את כל הקשרים, כיוון שלא תמיד יש קשר בין כל האיברים של הכניסה. למשל תמונה שנכנסת לרשת, כנראה אין קשר בין פיקסלים רחוקים, לכן אין טעם לחבר את הכניסה לכל הנוירונים בשכבה הראשונה, ולקשר בין כל שתי שכבות סמוכות בצורה מלאה. כדי להימנע מבעיות אלו, הרבה פעמים כדאי להשתמש ברשתות קונבולוציה, שאינן מקשרות בין כל שני נוירונים, אלא רק בין איברים קרובים, כפי שיפורט. הרבה מהרשתות המודרניות מבוססות על רשתות קונבולוציה, כאשר על גבי המבנה הבסיסי בנו כל מיני ארכיטקטורות מתקדמות.

5.1 Convolutional Layers

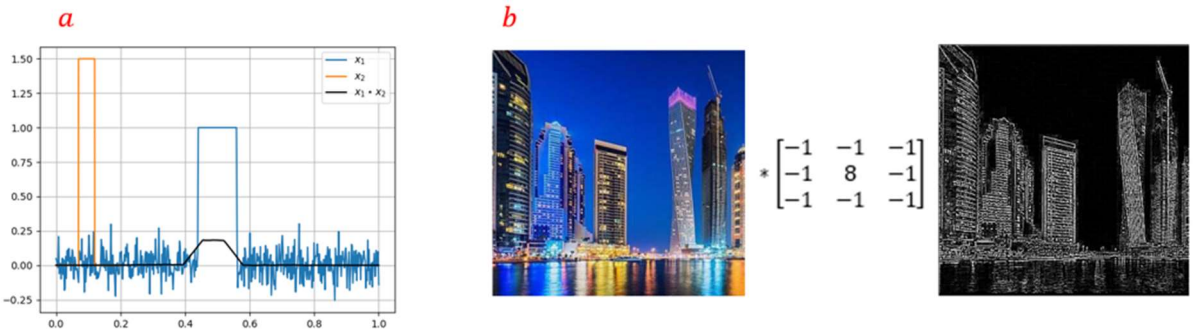
5.1.1 From Fully-Connected Layers to Convolutions

האלמנט הבסיסי ביותר ברשתות קונבולוציה הינו שכבת קונבולוציה, המבצעת קונבולוציה לינארית על פני דאטה מסוים בכדי לקבל ייצוג אחר ופשוט יותר שלו. כל שכבת קונבולוציה הינה למעשה וקטור או מערך של משקלים, המבצעת פעולת קונבולוציה (או ליתר דיוק – קרוס קורלציה) בין וקטור המשקלים לבין input מסוים (זה יכול להיות או וקטור הכניסה, או וקטור היוצא משכבה חבויה). וקטור זה נקרא גרעין הקונבולוציה (convolution kernel) או Filter, והוא מבצע את הפעולה המתמטית הבאה:

$$y[n] = \sum_{m=1}^{K-1} x[n-m]w[m]$$

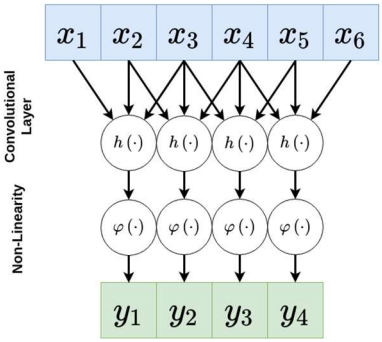
כאשר $x \in \mathbb{R}^n$ הוא וקטור הכניסה, ו-$w \in \mathbb{R}^K$ הוא וקטור המשקלים, והפרמטרים שלו נלמדים בתהליך האימון. בכל שכבה, וקטור המשקלים w זהה לכל הכניסות, ובכך מורידים באופן משמעותי את מספר הפרמטרים הנלמדים לעומת שכבת FC – בשכבת FC יש $N_{inputs} \times N_{outputs}$ משקלים, ואילו בשכבת קונבולוציה יש רק K משקלים.

מלבד הורדת כמות המשקלים, השימוש בגרעין קונבולוציה מסייע בעיקר לזיהוי דפוסים. היכולת לזהות דפוסים נובעת מאופי פעולת הקונבולוציה, הבודקת חפיפה בין חלקים מסוימים של וקטור לבין גרעין הקונבולוציה. הקונבולוציה יכולה למצוא פיצ'רים בסיגנל, וישנם גרעיני קונבולוציה שיכולים לבצע אוסף פעולות שימושיות, כמו למשל החלקה, נגזרת ועוד. אם מטילים על תמונה הרבה גרעינים שונים, ניתן למצוא בה כל מיני פיצ'רים – למשל אם הגרעין הוא בצורה של עין או אף, אז הוא מסוגל למצוא את האזורים בתמונה המקורית הדומים לעין או אף.



איור 5.1 a) קונבולוציה חד ממדית בין שתי פונקציות: x_1 הינו מלבן בגובה 1 עם רעש קטן (כחול), ו-x_2 הינו גרעין קונבולוציה מלבני שרץ על פני כל הישר (כתום). פעולת הקונבולוציה (שחור) בודקת את החפיפה בין הסיגנל לבין הגרעין, וניתן לראות שאכן סביב $x = 0.5 \pm 0.1$ יש אזור עם הרבה חפיפה. b) קונבולוציה דו מימדית למציאת קווי מתאר של בתוך תמונה.

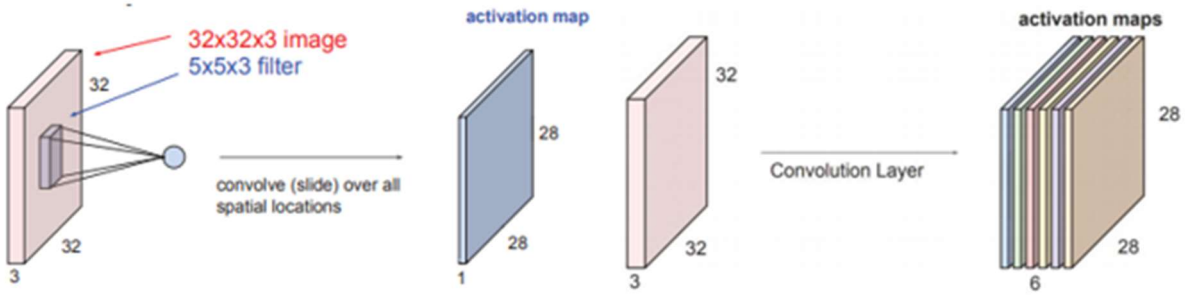
המוצא של שכבת הקונבולוציה עובר בפונקציית הפעלה לא לינארית (בדרך כלל tanh או ReLU), והוא מכונה activation map או feature map. הקונבולוציה יחד עם האקטיבציה נראות כך:



איור 5.2 דאטה x עובר דרך שכבת קונבולוציה ולאחריה פונקציית הפעלה, ובמוצא מתקבלת מפת אקטיבציה y.

לרוב בכל שכבת קונבולוציה יהיו כמה פילטרים, שכל אחד מהם אמור ללמוד פיצ'ר אחר בתמונה. ככל שהרשת הולכת ומעמיקה, כך הפיצ'רים בתמונה אמורים להיות מופרדים בצורה פשוטה יותר אחד מהשני, ולכן הפילטרים בשכבות העמוקות אמורים להבדיל בין דברים מורכבים יותר. למשל – פילטרים בשכבות הראשונות יכולים להבחין בגבולות של אלמנט בתוך תמונה, ואילו פילטרים בשכבות יותר עמוקות אמורים לדעת כבר לזהות מהו אותו אלמנט.

הקלט של שכבת הקונבולוציה יכול להיות רב ערוצי (נפוץ מאוד בתמונה). במקרה זה הקונבולוציה יכולה לבצע פעולה על כל הערוצים יחד ולספק פלט חד ערוצי, והיא יכולה גם לבצע פעולה על כל ערוץ בנפרד ובכך לספק פלט רב ערוצי. כמו כן, הקונבולוציה יכולה להיות דו ממדית, כלומר בכל פעם מטילים את הפילטר על אזור אחר



איור 5.3 פילטר $F \in \mathbb{R}^{5\times5\times3}$ פועל על קלט $x \in \mathbb{R}^{32\times32\times3}$ ומתקבלת מפת אקטיבציה $y \in \mathbb{R}^{28\times28}$ (שמאל). הקלט יכול לעבור דרך מספר פילטרים ולייצר מפת אקטיבציה עם מספר שכבות – עבור שישה פילטרים הממד של המפה הינו $y \in \mathbb{R}^{28\times28\times6}$ (ימין).

5.1.2 Padding, Stride and Dilation

כמו ברשת FC, גם ברשת קונבולוציה יש היפר-פרמטרים הנקבעים מראש וקובעים את אופן הפעולה של הרשת. ישנם שני פרמטרים של שבת הקונבולוציה – גודל הפילטר ומספר ערוצי הקלט, ושלושה פרמטרים מרכזיים של אופן פעולת הקונבולוציה:

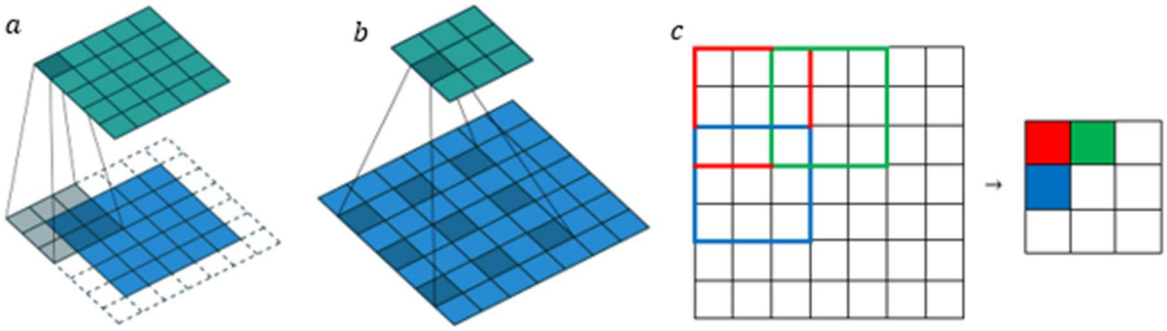
ריפוד (Padding): כיוון שהפילטר הוא מרחבי, כלומר הוא פועל על מספר איברים בכל פעם, לא ניתן לבצע את הקונבולוציה על האיברים בקצוות, כיוון שאז הפילטר יגלוש מעבר לדאטה הנתון. באיור 5.2 ניתן לראות כיצד פעולה על תמונה בממד של 32×32 מקטינה את ממד הפלט ל-28×28, דבר הנובע מכך שהקונבולוציה לא יכולה לפעול על הקצוות. אם רוצים לבצע את הקונבולוציה גם על הקצוות, ניתן לרפד את שולי הקלט (באפסים או שכפול של ערכי הקצה). אם נסמן את גודל הפילטר ב-K, אזי גודל הריפוד הוא: $\text{Zero Padding} = \frac{K+1}{2}$.

התרחבות (Dilation): על מנת לצמצם עוד במספר החישובים, אפשר לפעול על אזורים יותר גדולים מתוך הנחה שערכים קרובים גיאוגרפית הם בעלי ערך זהה. לשם כך ניתן להרחיב את פעולת הקונבלוציה תוך השמטה של ערכים קרובים. התרחבות טיפוסית הינה בעלת פרמטר $d = 2$.

גודל צעד (Stride): ניתן להניח שלרוב הקשר המרחבי נשמר באזורים קרובים, לכן על מנת להקטין בחישוביות ניתן לדלג על הפלט ולהפעיל את פעולת הקונבלוציה באופן יותר דליל. כלומר, אין צורך להטיל את הפילטר על כל האזורים האפשריים ברשת, אלא ניתן לבצע דילוגים, כך שלאחר כל חישוב קונבולוציה יבוצע דילוג בגודל הצעד לפני הקונבולוציה הבאה. גודל צעד טיפוסי הינו $s = 2$.

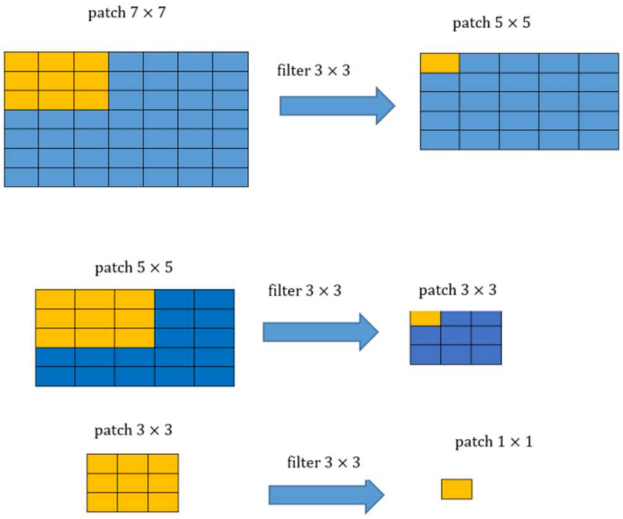
גודל שכבת הפלט לאחר ביצוע הקונבולוציה תלוי בגדלים של הכניסה והפילטר, בריפוד באפסים ובגודל הצעד. באופן פורמלי ניתן לחשב את הגודל לפי הנוסחה: $O = \frac{W-K+2P}{S} + 1$, כאשר W הוא גודל הכניסה, K הוא גודל הפילטר, P

זה הריפוד באפסים ו-S זה גודל הצעד. מספר שכבות הפלט הינו כמספר הפילטרים (כאשר שכבת פלט יכולה להיות רב ערוצית).



איור 5.4 (a) ריפוד באפסים על מנת ביצוע קונבולוציה גם על הקצוות של הדאטה. (b) התרחבות ($d = 2$): ביצוע הקונבולוציה תוך השמטת איברים סמוכים מתוך הנחה שכנראה הם דומים. (c) הזזת הפילטר בצעד של $s = 2$.

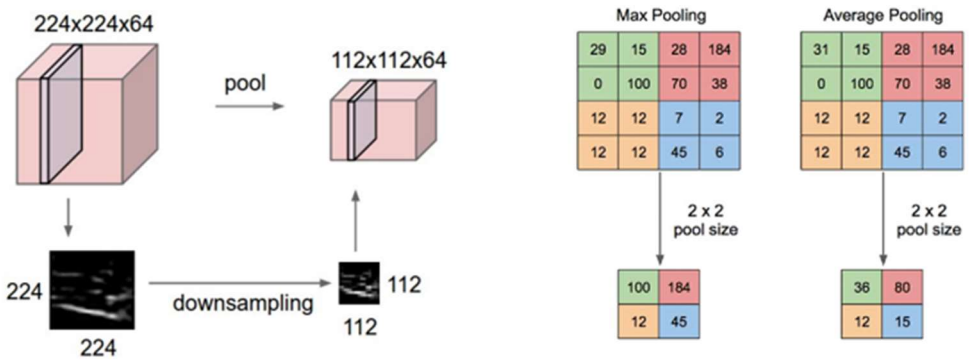
תמך (Receptive field) של איבר ברשת מוגדר להיות כל התחום בכניסה אשר משפיע על אותו איבר לאורך השכבות.



איור 5.5 Receptive field של ערך מסוים במוצא של שלוש שכבות קונבולוציה רצופות עם פילטר בגודל 3×3.

5.1.3 Pooling

הרבה פעמים דאטה מרחבי מאופיין בכך שאיברים קרובים דומים אחד לשני, למשל – פיקסלים סמוכים לרוב יהיו בעלי אותו ערך. ניתן לנצל עובדה זו בכדי להוריד את מספר החישובים הדרוש בעזרת דילוגים (Strides) כפי שתואר לעיל. שיטה אחרת לניצול עובדה זו היא לבצע Pooling\down sampling – אחרי כל ביצוע קונבולוציה, לקחת מכל אזור רק ערך אחד, המייצג את האזור. את הערך של תוצאת ה-pooling ניתן לבחור בכמה דרכים, כאשר המקובלות הן לקחת את האיבר הכי גדול באזור שלו (max pooling) או את הממוצע של האיברים (average pooling).



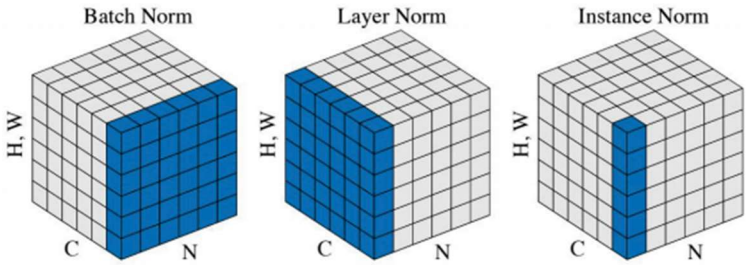
איור 5.6 הקטנת הממד של הדאטה בעזרת Pooling (שמאל), והמחשה מספרית של ביצוע max/average pooling בגודל של 2×2.

5.1.4 Training

תהליך האימון של רשת קונבולוציה זהה לאימון של רשת FC, כאשר ההבדל היחיד הוא בארכיטקטורה של הרשת. יש לשים לב שהפילטרים מופעלים על הרבה אזורים שונים, כאשר המשקלים של הפילטרים בכל צעד שווים, ולכן אותם משקלים פועלים על אזורים שונים. הגרדיאנט בכל צעד יהיה הסכום של הגרדיאנטים על פני כל הדאטה, ועבור המקרה הכללי בו יש N אזורים שונים עליהם מופעל הפילטר הגרדיאנט יהיה:

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^{N} \frac{\partial L}{\partial w_k(i)}$$

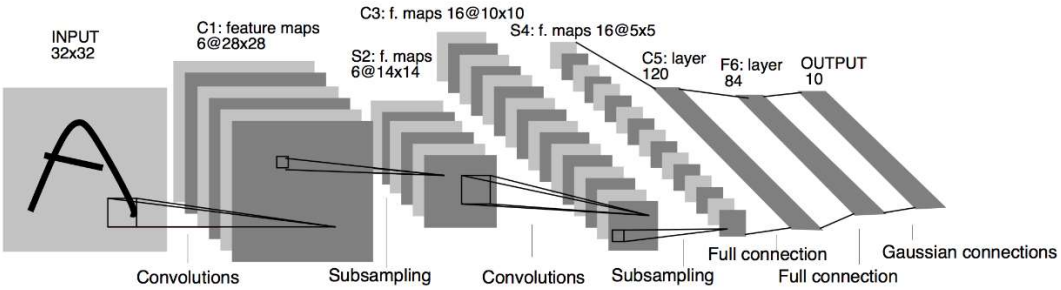
בדומה ל-FC, גם ב-CNN ניתן לבצע Mini-Batch Normalization, כאשר יש כמה אפשרויות לבצע את הנרמול על סט של וקטורים מסוימים (לשם הנוחות נתייחס לווקטורים של הדאטה כתמונות, כיוון שזה הכי נפוץ בהקשר של CNN). האפשרות הפשוטה היא לנרמל כל פילטר בפני עצמו על פני כמה תמונות (Batch Norm), כלומר לקחת את כל הפיקסלים בסט של תמונות ולנרמל בתוחלת ובשונות שלהם. אפשרות נוספת היא לקחת חלק מהמידע של סט תמונות, אך לנרמל אותו ביחס לאותו מידע על פני פילטרים אחרים (Layer Norm). יש וריאציות של הנרמולים האלה, כמו למשל Instance Norm, הלוקח פילטר אחד ותמונה אחת ומנרמל את הפיקסלים של אותה תמונה.



איור 5.7 נרמול שכבות של רשת קונבולוציה.

5.1.5 Convolutional Neural Networks (LeNet)

בעזרת שרשור של שכבות וחיבור כל האלמנטים השייכים לקונבולוציה ניתן לבנות רשת שלמה עבור מגוון משימות שונות. לרוב במוצא שכבות הקונבולוציה יש שכבה אחת או מספר שכבות FC. מטרת ה-FC היא לאפשר חיבור של המידע המוכל בפיצ'רים שנאספו במהלך שכבות הקונבולוציה. ניתן להסתכל על הרשת הכוללת כשני שלבים – בשלב הראשון מבצעים קונבולוציה עם פילטרים שונים, שכל אחד מהם נועד לזהות פיצ'ר אחר, ובשלב השני מחברים חזרה את כל המידע שנאסף על ידי חיבור כל הנוירונים באמצעות FC. לראשונה השתמשו בארכיטקטורה זו בשנת 1998, ברשת הנקראת LeNet (על שם Yann LeCun), ומוצגת באיור 5.7. רשת זו השיגה דיוק של 98.9% בזיהוי ספרות, כאשר המבנה שלה הוא שתי שכבות של קונבולוציה ושלוש שכבות FC, כאשר לאחר כל אחת משכבות הקונבולוציה מבצעים pooling.



איור 5.8 ארכיטקטורת LeNet.

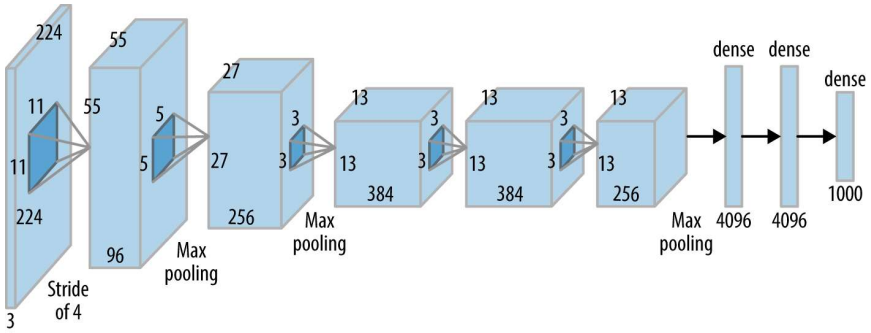
5.2 CNN Architectures

בשנים שלאחר LeNet העיסוק ברשתות נוירונים עמוקות די נזנח, עקב חוסר המשאבים לבצע חישובים רבים ביעילות ובמהירות. בשנת 2012 רשת בשם AlexNet המבוססת על שכבות קונבולוציה ניצחה בתחרות ImageNet (תחרות לזיהוי תמונות), כאשר היא הציגה שיפור של כמעט 10% מהתוצאה הכי טובה בשנה שלפני. יחד עם התפתחות יכולות החישוב, העיסוק ברשתות עמוקות חזר להיות מרכזי ופותחו הרבה מאוד ארכיטקטורות מתקדמות.

5.2.1 AlexNet

רשת AlexNet היא למעשה הרחבה של LeNet, כאשר היכולת שלה להתמודד עם משימות יותר מורכבות מאשר LeNet נובעת מכך שנהיו דאטה סטים גדולים מאוד שניתן לאמן עליהם את הרשת, ובנוסף כבר היה קיים GPU שבעזרתו ניתן לבצע חישובים מורכבים. הארכיטקטורה של הרשת מורכבת מחמש שכבות קונבולוציה ושלוש שכבות FC, כאשר לאחר שתי השכבות הראשונות של הקונבולוציה מתבצע pooling ו-normalization. ה-input הוא ממימד $227 \times 227 \times 3$, ומופעלים עליו 96 פילטרים בגודל 11×11, עם גודל צעד $s = 4$ וללא ריפוד באפסים. לכן המוצא של הקונבולוציה הינו ממימד $55 \times 55 \times 96$. לאחר מכן מתבצע max-pooling שמפחית את שני הממדים הראשונים, ומתקבלת שכבה במימד $27 \times 27 \times 96$. בשכבת הקונבולוציה השנייה יש 256 פילטרים בגודל 5×5 עם גודל צעד $s = 1$ וריפוד באפסים $p = 2$, לכן במוצא המימד הוא $27 \times 27 \times 256$, ואחרי max-pooling מתקבלת שכבה במימד $13 \times 13 \times 256$. לאחר מכן יש עוד 2 שכבות של קונבולוציה עם פילטרים במימד $3 \times 3 \times 384$, גודל צעד $s = 1$ וריפוד $p = 1$, ואז שכבת קונבולוציה אחרונה עם 256 פילטרים במימד 3×3, עם $s = p = 1$. במוצא הקונבולוציות יש עוד max-pooling, ואז שלוש שכבות FC, כאשר המוצא של השכבה האחרונה הוא וקטור באורך 1000, המייצג 1000 קטגוריות שונות שיש בדאטה סט ImageNet.

פונקציית האקטיבציה של הרשת הינה ReLU (בשונה מ-LeNet שהשתמשה ב-tanh), וההיפר פרמטרים הם: Dropout=0.5, batch size=128, SGD+momentum=0.9, lr=1e-2. בסך הכל מספר הפרמטרים ברשת הינו בערך 60 מיליון.

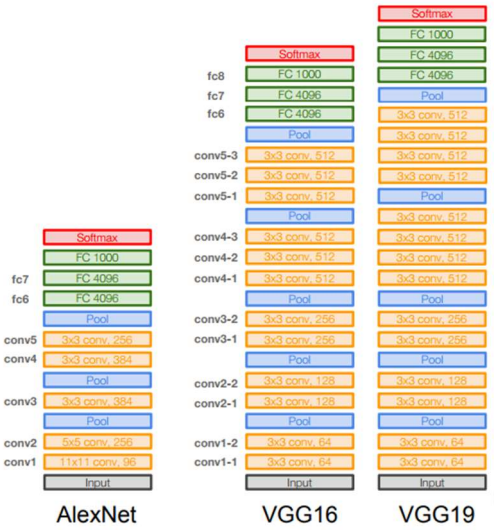


איור 5.9 ארכיטקטורת AlexNet.

שנה לאחר AlexNet פורסמה רשת דומה בשם ZFNet, הבנויה באותה ארכיטקטורה עם הבדלים קטנים בהיפר-פרמטרים ובמספר הפילטרים: השכבה הראשונה של הקונבלוציה הפכה מ: $11 \times 11, s = 4$ ל: $7 \times 7, s = 2$, ובשכבות 3-4-5 מספר הפילטרים הוא 512,1024,512 בהתאמה. הרשת השיגה שיפור של כ-5% על פני AlexNet. המימד של השכבות בשתי הארכיטקטורות אינו נובע מסיבה מסוימת אלא מניסוי וטעיה – ניסו כל מיני קונפיגורציות וראו מה סיפק את הביצועים הכי טובים. לאחר שהרשתות מבוססות קונבולוציה הוכיחו את כוחן, השלב הבא היה לבנות רשתות יותר עמוקות, ובעלות ארכיטקטורה הנשענת לא רק על ניסויים אלא גם על היגיון מסוים.

5.2.2 VGG

שנה לאחר ZFNet הצליחו לבנות רשת יותר עמוקה – בעלת 19 שכבות, על ידי ניצול יותר טוב של שכבות הקונבולוציה. המפתחים של הרשת שמו לב שניתן להחליף שכבת פילטרים של 7×7 בשלוש שכבות של 3×3 ולקבל את אותו receptive field, כאשר מרוויחים חסכון משמעותי במספר הפרמטרים הנלמדים. לפילטר בגודל $d \times d$ הפועל על c ערוצי קלט ופלט יש d^2c^2 פרמטרים נלמדים, לכן לפילטר של 7×7 יש $49c^2$ פרמטרים נלמדים ואילו לשלוש שכבות של 3×3 יש $3 \cdot 3^2c^2 = 27c^2$ פרמטרים נלמדים – חיסכון של 45%. הרשת נקראת VGG16 והיא מכילה 138 מיליון פרמטרים, ויש לה וריאציה המוסיפה עוד שתי שכבות קונבולוציה ומכונה VGG19.

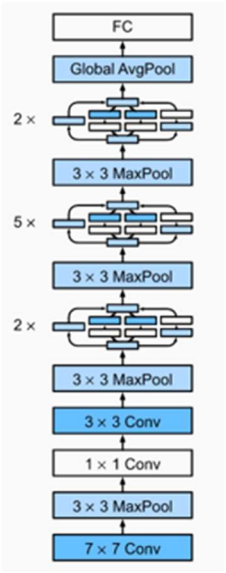


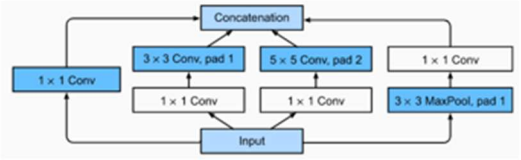
איור 5.10 ארכיטקטורת VGG.

5.2.3 GoogleNet

המודלים הקודמים היו יחסית יקרים מאוד מבחינת מספר פרמטרים. כדי להצליח להגיע לאותם ביצועים עם אותו עומק אבל עם הרבה פחות פרמטרים, גוגל הציעו את הרעיון שנקרא inception module. בלוק כזה מבצע הרבה פעולות פשוטות במקביל, במקום לבצע פעולה אחת מורכבת. כל בלוק מקבל input מסוים, ומבצע עליו ארבעה חישובים במקביל, כאשר המימדים של מוצאי כל הענפים שווים כך שניתן לשרשר אותם יחד. ארבעת הענפים הם: קונבולוציה 1×1, קונבולוציה 1×1 ולאחריה קונבולוציה 3×3 עם padding בגודל 1, קונבולוציה 1×1 ולאחריה קונבולוציה 5×5 עם padding בגודל 2, ו-3×3 max pooling עם 1 padding ולאחריו קונבולוציה 1×1. לבסוף, הפלטים של ארבעת הענפים משורשרים יחד ומהווים את פלט הבלוק.

המבנה הזה שקול למספר רשתות במקביל, כאשר היתרון של המבנה הזה הוא כפול: כמות פרמטרים נמוכה ביחס לרשתות קודמות וחישובים יחסית מהירים כיוון שהם נעשים במקביל. ניתן לחבר שכבות קונבולוציה רגילות עם בלוקים כאלה, ולקבל רשת עמוקה. נעשו הרבה ניסויים כדי למצוא את היחס הנכון בין הרכיבים והמימדים בכל שכבה המביאים לביצועים אופטימליים.





איור 5.11 Inception Block יחיד (ימין), וארכיטקטורת GoogleNet מלאה (שמאל).

5.2.4 Residual Networks (ResNet)

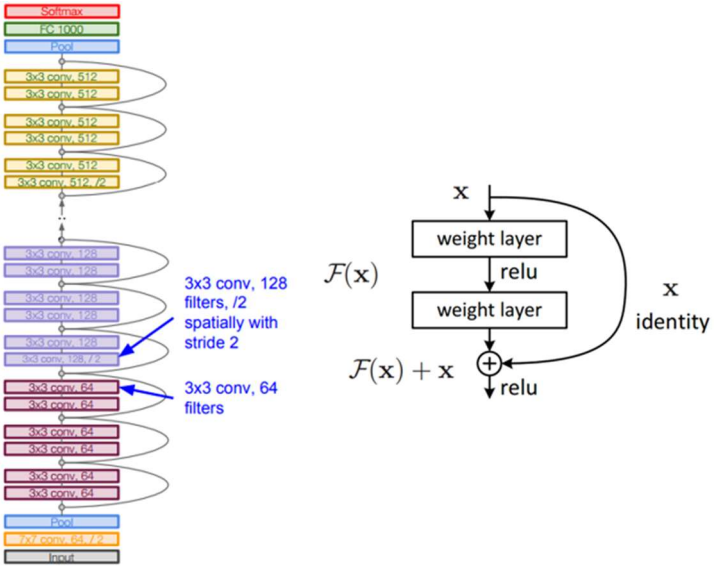
לאחר שראו שככל שהרשת עמוקה יותר כך היא משיגה תוצאות טובות יותר, ניסו לבנות רשתות עם מאות שכבות, אך הן השיגו תוצאות פחות טובות מהרשתות הקודמות שהיו בעלות סדר גודל של 20 שכבות. הבעיה המרכזית של

הרשתות העמוקות נבעה מכך שלאחר מספר שכבות מסוים התקבל ייצוג מספיק טוב, וכעת השכבות היו צריכות לא לשנות את הקלט אלא להעביר את הייצוג כמו שהוא. בשביל לבצע זאת המשקלים בשכבות אלו צריכים להיות 1. הסתבר שלשכבות קשה ללמוד את פונקציית הזהות והן למעשה פגעו בתוצאה. אתגר נוסף ברשתות עמוקות נבע מהקושי לבצע אופטימיזציה כמו שצריך למשקלים בשכבות עמוקות.

ניתן לנסח את הבעיה במרכזית באופן מעט שונה – בהינתן רשת עם N שכבות, יש טעם להוסיף שכבה נוספת רק אם היא תוסיף מידע שלא קיים עד עכשיו. כדי להבטיח ששכבה תוסיף מידע, או לכל הפחות לא תפגע במידע הקיים, בנו רשת חדשה בעזרת Residual Blocks – יצירת בלוקים של שכבות קונבולוציה, כאשר בנוסף למעבר של המידע בתוך הבלוק, מחברים גם בין הכניסה למוצא שלו. כעת אם בלוק מבצע פונקציה מסוימת $\mathcal{F}(x)$, אזי המוצא הינו $\mathcal{F}(x) + x$. באופן הזה כל בלוק ממוקד בללמוד משהו שונה ממה שנלמד עד עכשיו, ואם אין מה להוסיף – הפונקציה $\mathcal{F}(x)$ פשוט נשארת 0. בנוסף, המבנה של הבלוקים מונע מהגרדיאנט בשכבות העמוקות להתבדר או להתאפס, והאימון מצליח להתכנס.

באופן הזה בנו רשת בעלת 152 שכבות שהציגה ביצועים מעולים ביחס לכל שאר הרשתות באותה תקופה. השכבות היו מורכבות משלשות של בלוקים, כאשר בכל בלוק יש שתי שכבות קונבולוציה. בין כל שלשה יש הכפלה של מספר הפילטרים והורדה של המימד פי שניים בעזרת pooling. ההיפר-פרמטרים הם: Batch normalization אחרי כל שכבת קונבולוציה, אתחול משקלים בשיטת Xavier initialization, SGD+momentum=0.9, lr=0.1 ומחולק ב-10 בכל פעם שה-validation error מתיישר, batch size=256 ו-weight decay=1e-5.

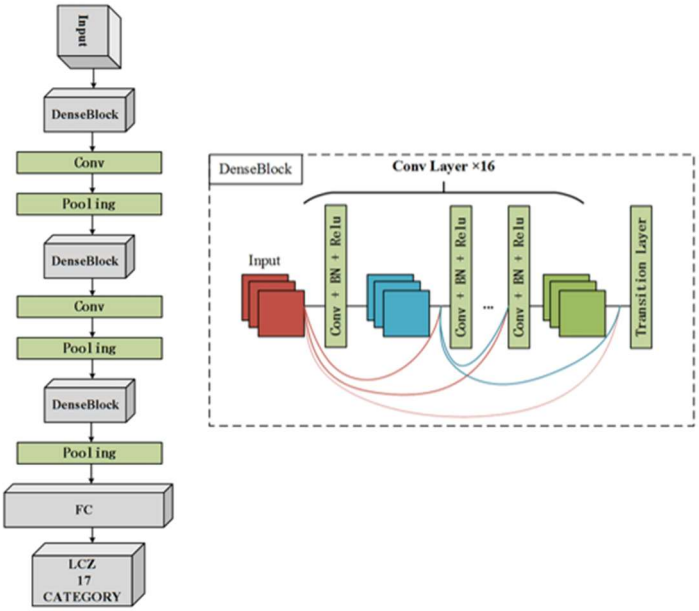
רשתות יותר מתקדמות שילבו את גישת ה-inception יחד עם ResNet על מנת לשלב בין היתרונות של שתי השיטות.



איור 5.12 Residual Block יחיד (ימין), וארכיטקטורת ResNet מלאה (שמאל).

5.2.5 Densely Connected Networks (DenseNet)

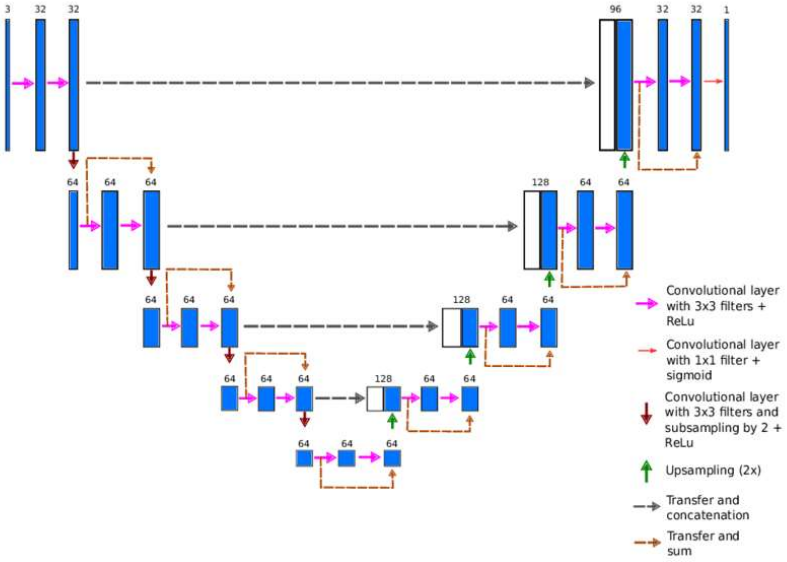
ניתן להרחיב את הרעיון של Residual Block כך שלא רק מחברים את הכניסה של כל בלוק למוצא שלו, אלא גם שומרים את הכניסה בפני עצמה, ובודקים את היחס שלה לשכבות יותר עמוקות. Dense block הוא בלוק בעל כמה שכבות, כך שכניסה של כל שכבה מחוברת לכל הכניסות של השכבות אחריה. ניתן כמובן לשרשר כמה בלוקים כאלה יחד, ולבצע ביניהם כל מיני פעולות כמו pooling או אפילו שכבת קונבולוציה עצמאית. כיוון שמשלבים כמה כניסות של בלוקים שונים, יש בעיה של התאמת מימדים, כיוון שכל בלוק מגדיל את מספר הערוצים, חיבור של כמה בלוקים יכולים ליצור מודל מורכב מדי. כדי להתגבר על בעיה זו מוסיפים transition layers המבצעים קונבולוציות 1×1 עם רוחב צעד $s = 2$, ובכך מספר הערוצים סביר והמודל לא נהיה מורכב מדי.



איור 5.13 Dense Block יחיד (ימין), וארכיטקטורת DenseNet מלאה (שמאל).

5.2.6 U-Net

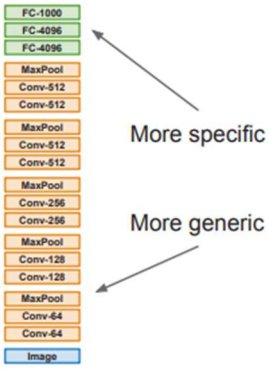
ברשתות קונבולוציה המיועדות לסיווג, בסוף התהליך מתקבל וקטור של הסתברויות, כאשר כל איבר הוא הסתברות של label מסוים. במשימת סגמנטציה זה בעייתי, כיוון שצריך בסוף התהליך לא רק ללמוד את הפיצ'רים שבתמונה ועל פיהם לקבוע מה יש בתמונה, אלא צריך גם לשחזר את התמונה המקורית עם הסגמנטציה המתאימה. כדי להתמודד עם בעיה זו הציעו את ארכיטקטורת U-Net, המכילה שלושה חלקים עיקריים: כיווץ, צוואר בקבוק והרחבה (contraction, bottleneck, and expansion section). כפי שניתן לראות באיור, בחלק הראשון יש טופולוגיה רגילה של רשת קונבולוציה, המבוצעת בעזרת שכבות קונבולוציה וביצוע pooling. השוני בין השלב הזה לבין רשת קונבולוציה קלאסית הוא החיבור שיש בין כל שלב בתהליך לבין חלקים בהמשך התהליך. לאחר המעבר בצוואר הבקבוק יש למעשה שחזור של התמונה עם הסגמנטציה. השחזור נעשה בעזרת up-sampling על הוקטור שהתקבל במוצא צוואר הבקבוק יחד עם המידע שנשאר מהחלק הראשון של התהליך. פונקציית המחיר שמשתמשים ברשת זו נקראת pixel-wise cross entropy loss, הבודקת כל פיקסל ביחס ל-label האמיתי אליו הוא שייך.



איור 5.14 ארכיטקטורת U-Net.

5.2.7 Transfer Learning

כאשר נתקלים במשימה חדשה, אפשר לתכנן עבורה ארכיטקטורה מסוימת ולאמן רשת עמוקה. בפועל זה יקר ומסובך להתאים רשת מיוחדת לכל בעיה ולאמן אותה מהתחלה, ולכן ניתן להשתמש ברשתות הקיימות שאומנו כבר ולהתאים אותן לבעיות אחרות. גישה זו נקראת Transfer Learning, וההיגיון מאחוריה הוא שאם יש רשת מוכנה שהוכיחה את עצמה בתחום מסוים, ניתן לבצע בה שינויים קלים כך שתוכל להתמודד גם עם בעיות אחרות. למעשה בדרך כלל לוקחים רשת קיימת ומוסיפים לה עוד שכבות בסוף ומאמנים אותן על הדאטה החדש, כך שהשכבות יהיו מוכוונות לדאטה הספציפי של המשימה החדשה. ככל שיש יותר דאטה חדש ניתן להוסיף יותר שכבות ולקבל דיוק יותר טוב, וככל שהמשימה החדשה דומה יותר למשימה המקורית של הרשת כך יש צורך בפחות שכבות חדשות. אגב, שיטה זו יכולה לסייע גם במניעת overfitting הנובע מחוסר בדאטה.



	very similar dataset	very different dataset
very little data	Finetune linear classifier on top layer	You're in trouble… Try data augmentation / collect more data
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

איור 5.15 Transfer Learning.

References

Convolutional:

https://github.com/technion046195/technion046195

מצגות מהקורס של פרופ' יעקב גולדברגר

AlexNet:

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

VGG

https://arxiv.org/abs/1409.1556

GoogleNet

http://d2l.ai/chapter_convolutional-modern/vgg.html

ResNet

https://arxiv.org/abs/1512.03385

DenseNet

https://arxiv.org/abs/1608.06993

https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

U-Net

https://arxiv.org/abs/1505.04597