

5. Convolutional Neural Networks (CNNs)

הרשתות שתוארו עד כה הינן Fully-Connected (FC), כלומר, כל נירון מחובר לכל הנירונים בשכבה שלפניו ולכל הנירונים בשכבה שאחריו. גישה זו יקרה מבחינה חישובית, ופעמים רבות אין צורך בכל הקשרים בין הנירונים. לדוגמא, תמונה בגווי אפור (grayscale) בעלת $256 \times 256 \times 1$ פיקסלים, המוזנת לרשת FC עם $N = 1000$ קטגוריות במוצא, מכילה יותר מ-65 מיליון קשרים בין נירונים, כאשר כל קשר הינו משקל המתעדכן במהלך הלמידה. אם יש מספר שכבות רב המספר נהיה עצום ממש, ולכן כמות הקשרים והפרמטרים גדלה, באופן כזה שבלתי מעשי לתחזק את הרשת. מלבד בעיית הגודל, בפועל לא תמיד יש צורך בכל הקשרים, כיוון שלא תמיד יש קשר בין כל איברי הכניסה. למשל, עבור תמונה המוזנת לרשת, במשימות רבות קשר בין פיקסלים רחוקים בתמונה איננו משמעותי, לכן אין חשיבות לחבר את הכניסה לכל הנירונים בשכבה הראשונה ולקשר בין כל שתי שכבות סמוכות באופן מלא. כדי להימנע מבעיות אלו, לרוב יהיה כדאי להשתמש ברשתות או שכבות קונבולוציה, שאינן מקשרות בין כל שני נירונים, אלא רק בין איברים קרובים, כפי שיפורט. רשתות מודרניות רבות מבוססות על שכבות קונבולוציה, כאשר על גבי המבנה הבסיסי נבנו 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 בכדי לקבל ייצוג אחר ופשוט יותר שלו. לרוב, שכבת קונבולוציה מבצעת פעולת קרוס-קורלציה בין וקטור המשקלים לבין input מסוים (וקטור הכניסה או וקטור היוצא משכבה חבויה). וקטור המשקלים נקרא גרעין הקונבולוציה (convolution kernel) או מסנן (filter), ובעזרתו מבוצעת פעולת הקרוס-קורלציה הבאה:

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

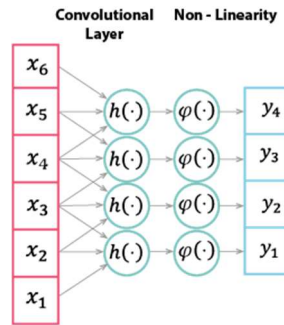
כאשר $x \in \mathbb{R}^n$ הוא וקטור הכניסה ואילו $w \in \mathbb{R}^K$ הוא וקטור המשקלים אשר נלמדים במהלך האימון. וקטור המשקלים w זהה לכל הכניסות בשכבה ולכן מספר הפרמטרים הנלמדים לעומת שכבת FC הינו קטן בהרבה – שכבת FC מכילה $N_{inputs} \times N_{outputs}$ משקלים ואילו שכבת קונבולוציה מכילה K משקלים בלבד (לרוב מתקיים $K \ll N_{inputs} \times N_{outputs}$).

מלבד הקטנת כמות המשקלים, השימוש בגרעין קונבולוציה מסייע לזיהוי דפוסים ולמציאת מאפיינים. יכולות אלו נובעות מאופי פעולת הקונבולוציה, הבודקת חפיפה בין חלקים מווקטור הכניסה לבין גרעין הקונבולוציה. הקונבולוציה יכולה למצוא מאפיינים 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 מאפיינים – 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 ממדית למציאת קווי מתאר של בתוך תמונה.

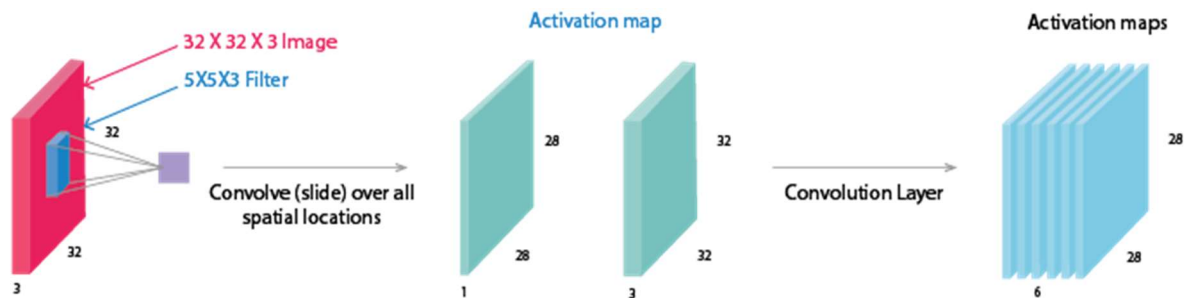
המוצא של שכבת הקונבולוציה עובר בפונקציית הפעלה לא לינארית (בדרך כלל tanh או ReLU), והוא מכונה מפת הפעלה (activation map) או מפת מאפיינים (feature map). הקונבולוציה יחד עם האקטיבציה נראות כך:



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

לרוב בכל שכבת קונבולוציה יהיו כמה מסננים, אשר כל אחד מהם אמור ללמוד מאפיין אחר בתמונה. ככל שהרשת הולכת ומעמיקה, כך המאפיינים בתמונה אמורים להיות מובחנים באופן חד יותר אחד מהשני, ולכן המסננים בשכבות העמוקות אמורים להבדיל בין דברים מורכבים יותר. למשל, פעמים רבות ניתן להבחין כי המסננים בשכבות הראשונות יזהו את שפות האלמנטים שבתמונה או בצורות אבסטרקטיות, ואילו מסננים בשכבות העמוקות יותר יזהו אלמנטים מורכבים יותר כמו איברים או חפצים שלמים בעלי צורה ידועה ומוגדרת.

הקלט של שכבת הקונבולוציה יכול להיות רב ערוצי (למשל, תמונה צבעונית המיוצגת לרוב בעזרת ערכי RGB). 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. גרעין הקונבולוציה יכול להיות חד ממדי, כלומר וקטור שפועל על קלט מסוים, אך הוא יכול להיות גם מממד גבוה יותר. לרוב, מסננים הפועלים על תמונות הינם דו ממדיים, ופעולת הקונבולוציה מבצעת בכל שלב כפל בין המסנן לבין אזור דו ממדי אחר בתמונה.



איור 5.3 מסנן $F \in \mathbb{R}^{5 \times 5 \times 3}$ פועל על קלט $x \in \mathbb{R}^{32 \times 32 \times 3}$ ומתקבלת מפת אקטיביציה $y \in \mathbb{R}^{28 \times 28}$ (שמאל). הקלט יכול לעבור דרך מספר מסננים ולייצר מפת אקטיביציה עם מספר שכבות – עבור שישה מסננים הממד של המפה יהיו $y \in \mathbb{R}^{28 \times 28 \times 6}$ (ימין).

5.1.2 Padding, Stride and Dilation

כמו ברשת FC, גם ברשת קונבולוציה יש היפר-פרמטרים הנקבעים מראש וקובעים את אופן פעולת הרשת. ישנם שני פרמטרים של שכבת הקונבולוציה – גודל המסנן ומספר ערוצי הקלט וכן שלושה פרמטרים עיקריים נוספים הקובעים את אופן פעולת הקונבולוציה:

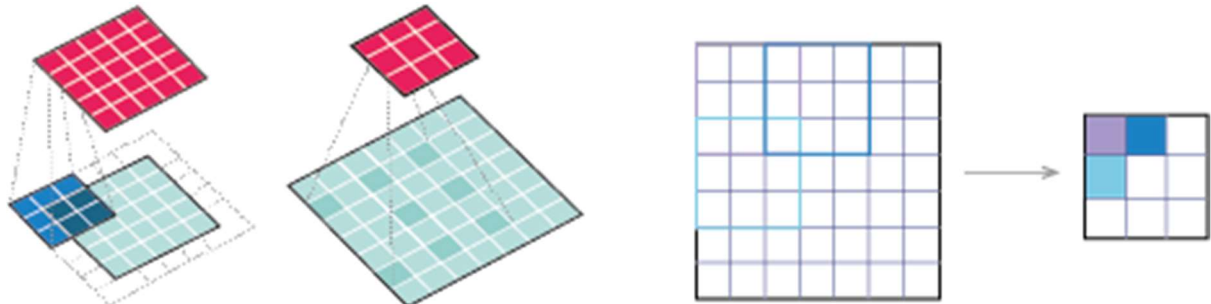
ריפוד (Padding): פעולת הקונבולוציה המוגדרת המסנן הינה פעולה מרחבית, כלומר, המסנן פועל על מספר איברים בכל פעולה. בנוסף, נשים לב כי פעולת הקונבולוציה לא מוגדרת על איברי הקצוות לכן לא נוכל להפעיל את המסנן במקומות אלו. באיור 5.2 ניתן לראות כיצד פעולה על תמונה בממד של 32×32 מקטינה את ממד הפלט ל- 28×28 , דבר הנובע מכך שהקונבולוציה לא מוגדרת על הפיקסלים בקצוות התמונה ולכן לא מופעלת עליהם. אם רוצים לבצע את הקונבולוציה גם על הקצוות, ניתן לרפד את שולי הקלט (באפסים או שכפול של ערכי הקצה). עבור

$$\text{Padding} = \frac{K-1}{2}, K \times K \text{ גודל הריפוד הנדרש הינו:}$$

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 הקונבולוציה תוך השמטה של ערכים קרובים. התרחבות טיפוסית הינה בעלת פרמטר $d = 2$.

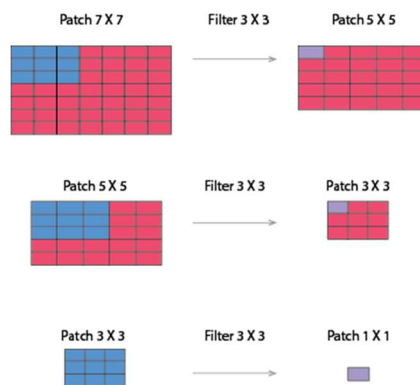
גודל צעד (Stride): ניתן להניח שלרוב הקשר המרחבי נשמר באזורים קרובים, לכן על מנת להקטין בחישוביות ניתן לדלג על הפלט ולהפעיל את פעולת הקונבולוציה באופן יותר דליל. כלומר, אין צורך להטיל את המסנן על כל האזורים האפשריים ברשת, אלא ניתן לבצע דילוגים, כך שלאחר כל חישוב קונבולוציה יבוצע דילוג בגודל הצעד לפני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

גודל שכבת הפלט לאחר ביצוע הקונבולוציה תלוי בגדלים של הכניסה והמסנן, בריפוד באפסים ובגודל הצעד. באופן פורמלי ניתן לחשב את גודל שכבת הפלט לפי הנוסחה: $O = \frac{W-K+2P}{S} + 1$, כאשר W הוא גודל הכניסה, K הוא גודל המסנן, P זה הריפוד באפסים ו- S זה גודל הצעד. מספר שכבות הפלט הינו כמספר המסננים (כאשר שכבת פלט יכולה להיות רב ערוצית). יש לשים לב שערכי ההיפר-פרמטרים (padding, dilation and stride) וכן גודל הגרעין נדרשים להיות מספרים טבעיים אשר מקיימים את נוסחת גודל שכבת הפלט (O) הנ"ל, כך שגם O הינו מספר טבעי.



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 המסנן בצעד של $s = 2$.

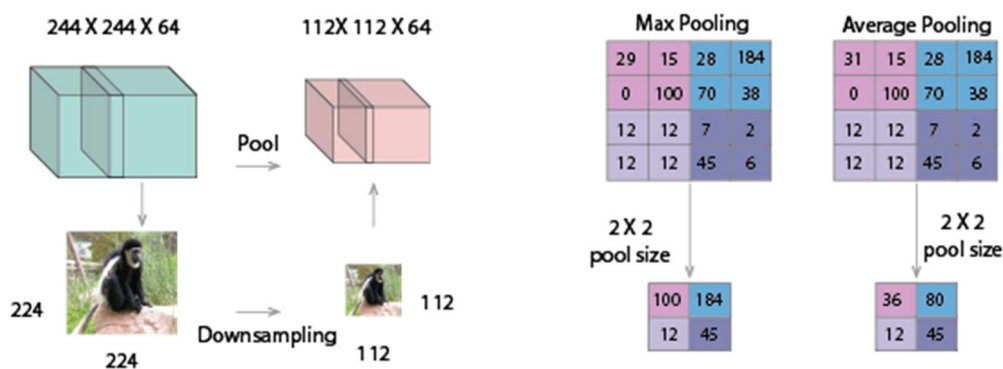
תמך (Receptive field) של איבר ברשת מוגדר להיות כל התחום בכניסה אשר משפיע על אותו איבר לאורך השכבות.



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

5.1.3 Pooling

פעמים רבות 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) או הרחבה (dilation) כפי שתואר לעיל. שיטה אחרת לניצול עובדה זו היא לבצע Pooling – אחרי כל ביצוע קונבולוציה, דגימת ערך יחיד מאזור בעל ערכים מרובים, המייצג את האזור. את צורת חישוב הערך של תוצאת ה-pooling ניתן לבחור בכמה דרכים, כאשר המקובלות הן בחירת האיבר הגדול ביותר באזור שלו (max pooling) או את הממוצע של האיברים (average pooling).



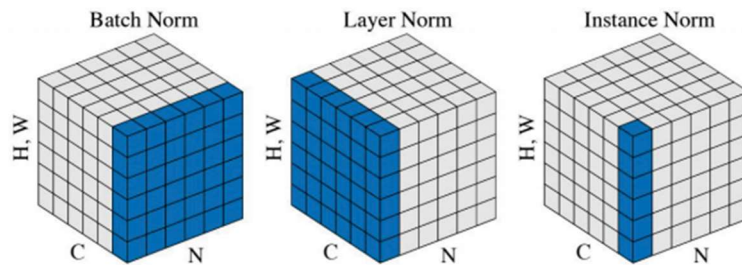
איור 5.6 הקטנת הממד של הדאטה בעזרת Pooling (שמאל), והמחשה מספרית של ביצוע max/average pooling בגודל של 2×2 .

5.1.4 Training

ככלל, תהליך האימון של רשת קונבולוציה זהה לאימון של רשת FC, כאשר ההבדל היחיד הוא בארכיטקטורה של הרשת. יש לשים לב שהמסגרים מופעלים על הרבה אזורים שונים, כאשר המשקלים של המסגרים בכל צעד שווים, ולכן אותם משקלים פועלים על אזורים שונים. לשם הפשטות נניח ויש מסגן יחיד, כלומר מטריצה אחת נלמדת של משקלים. מטריצה זו מוכפלת בכל אחד מהאזורים השונים של הדאטה, וכדי לבצע עדכון למשקלים שלה יש לשקלל את הגרדיאנטים של כל האזורים השונים. בפועל, הגרדיאנט בכל צעד יהיה הסכום של הגרדיאנטים על פני כל הדאטה, ועבור המקרה הכללי בו יש N אזורים שונים עליהם מופעל המסגן הגרדיאנט יהיה:

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

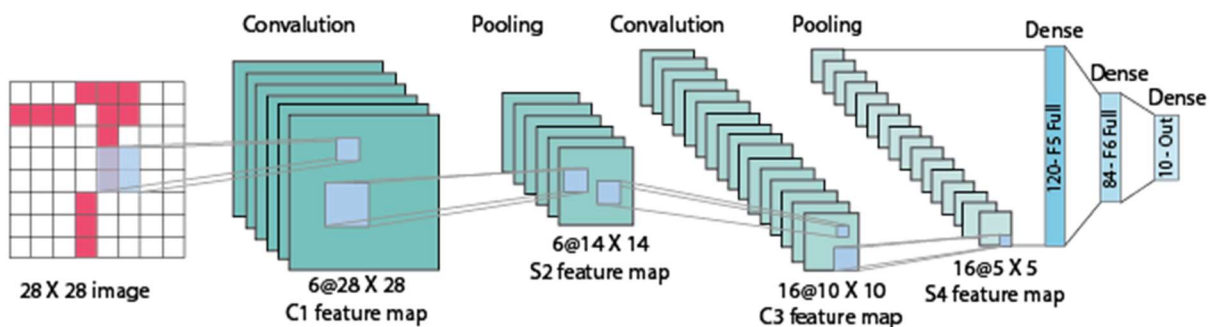
בדומה ל-FC, גם ב-CNN ניתן לבצע Mini-Batch Normalization, כאשר יש כמה אפשרויות לבצע את הנרמול על סט של וקטורים מסוימים (לשם הנוחות נתייחס לווקטורים של הדאטה כתמונות). אפשרות פשוטה ונפוצה היא לנרמל כל מסגן בפני עצמו על פני כמה תמונות (Batch Norm), כלומר לקחת את כל הפיקסלים בסט של תמונות ולנרמל בתוחלת ובשונות שלהם. אפשרות נוספת היא לקחת חלק מהמידע של סט תמונות, אך לנרמל אותו ביחס לאותו מידע על פני מסגרים אחרים (Layer Norm). יש וריאציות של הנרמולים האלה, כמו למשל Instance Norm, הלווק מסגן 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 שרשרת 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 במאפיינים שנאספו במהלך שכבות הקונבולוציה. ניתן להסתכל על הרשת הכוללת כשני שלבים – בשלב הראשון מבצעים קונבולוציה עם מסגרים שונים, שכל אחד מהם נועד לזהות מאפיין אחר, ובשלב השני מחברים חזרה את כל המידע שנאסף על ידי חיבור כל הנירונים באמצעות FC. לראשונה השתמשו בארכיטקטורה זו בשנת 1998, ברשת הנקראת LeNet (על שם Yann LeCun), ומוצגת באיור 5.8. 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.

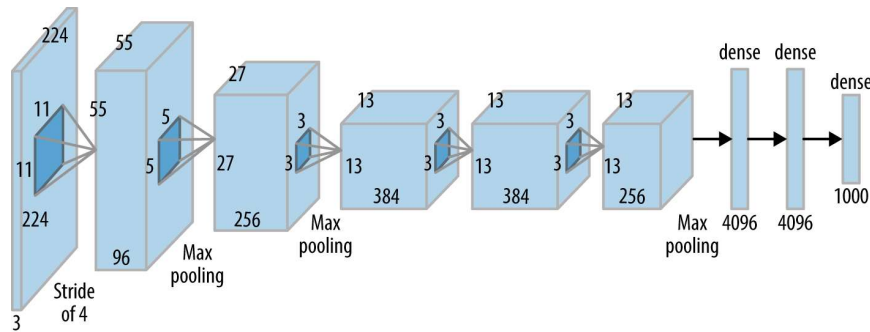
5.2 CNN Architectures

בשנים שלאחר LeNet העיסוק ברשתות נוירוניות 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 משמעותי 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 שאבה את ההשראה למבנה שלה מארכיטקטורת 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 מממד $224 \times 224 \times 3$, ומופעלים עליו 96 מסננים בגודל 11×11 , עם גודל צעד $s = 4$ וללא ריפוד באפסים. לכן המוצא של הקונבולוציה הינו מממד $55 \times 55 \times 96$. לאחר מכן מתבצע max-pooling שמפחית את שני הממדים הראשונים, ומתקבלת שכבה בממד $27 \times 27 \times 96$. בשכבת הקונבולוציה השנייה יש 256 מסננים בגודל 5×5 עם גודל צעד $s = 1$ וריפוד באפסים $p = 2$, לכן במוצא הממד הוא $27 \times 27 \times 256$, ואחרי max-pooling מתקבלת שכבה בממד $13 \times 13 \times 256$. לאחר מכן יש עוד 2 שכבות של קונבולוציה עם מסננים בממד $3 \times 3 \times 384$, גודל צעד $s = 1$ וריפוד $p = 1$, ואז שכבת קונבולוציה אחרונה עם 256 מסננים בממד 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: $\text{Dropout}=0.5$, $\text{batch size}=128$, $\text{SGD+momentum}=0.9$, $\text{lr}=1e-2$. בסך הכל מספר הפרמטרים ברשת הינו בערך 60 מיליון.

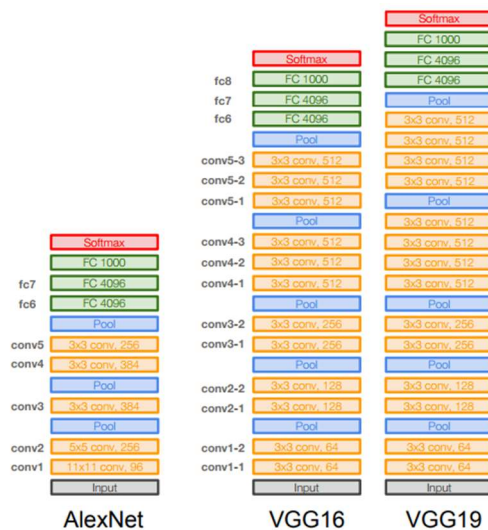


איור 5.9 ארכיטקטורת AlexNet.

שנה לאחר הפרסום של רשת AlexNet, פותחה רשת דומה בשם ZFNet, הבנויה באותה ארכיטקטורה עם הבדלים קטנים בהיפר-פרמטרים ובמספר הפילטרים: השכבה הראשונה של הקונבולוציה 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. הממד של השכבות בשתי הארכיטקטורות אינו נובע מסיבה מסוימת אלא מניסוי וטעיה –נוסו תצורות רבות ומתוכן נבחרה זו בעלת הביצועים הטובים ביותר. לאחר שהרשתות מבוססות קונבולוציה הוכיחו את כוחן, השלב הבא היה לבנות רשתות עמוקות, ובעלות ארכיטקטורה הנשענת לא רק על ניסויים אלא גם על היגיון מסוים.

5.2.2 VGG

שנה לאחר ZFNet הוצגה בתחרות רשת עמוקה – בעלת 19 שכבות, המנצלת יותר טוב את שכבות הקונבולוציה. מפתחי הרשת הראו כי ניתן להחליף שכבת פילטרים של 7×7 בשלוש שכבות של 3×3 ולקבל את אותו תמך (receptive field), כאשר מרוויחים חסכון משמעותי במספר הפרמטרים הנלמדים. לפילטר בגודל $d \times d$ הפועל על c ערוצי קלט ופלט יש $d^2 c^2$ פרמטרים נלמדים, לכן לפילטר של 7×7 יש $49c^2$ פרמטרים נלמדים ואילו לשלוש שכבות של 3×3 יש $27c^2 = 3 \cdot 3^2 \cdot c^2$ פרמטרים נלמדים – חיסכון של 45%. הרשת המקורית שפיתחו נקראת VGG16 והיא מכילה 138 מיליון פרמטרים, ויש לה וריאציה המוסיפה עוד שתי שכבות קונבולוציה ומכונה VGG19.



איור 5.10 ארכיטקטורת VGG (ימין) ביחס לארכיטקטורת AlexNet (שמאל).

5.2.3 GoogleNet

המודלים הקודמים היו יקרים חישובית עקב מספר הפרמטרים הגדול. בכדי להצליח להגיע לאותם ביצועים עם אותו עומק אבל עם הרבה פחות פרמטרים, קבוצת מפתחים מגוגל הציגו קונספט שנקרא inception module. בלוק המבצע הרבה פעולות פשוטות במקביל, במקום לבצע פעולה אחת מורכבת. כל בלוק מקבל input ומבצע עליו ארבעה חישובים במקביל, כאשר הממדים 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 padding 1 ולאחריה 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 וחשובים 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 והממדים בכל שכבה המביאים לביצועים אופטימליים.



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

5.2.4 Residual Networks (ResNet)

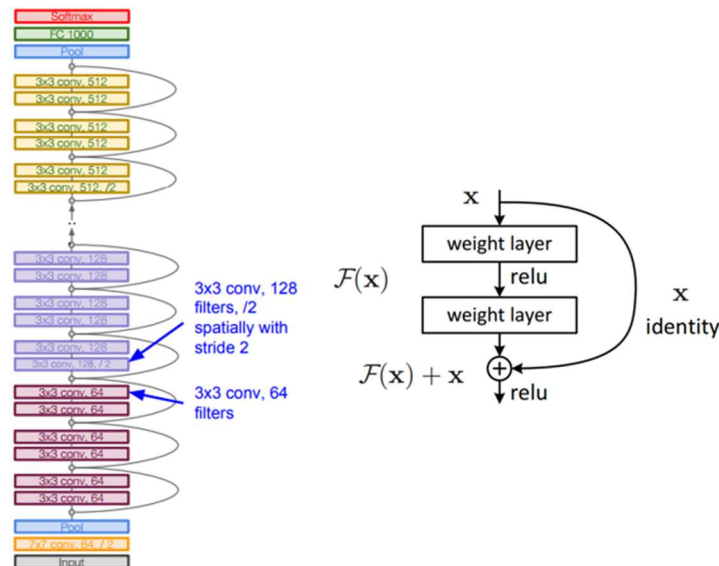
לאחר שראו שכל שהרשת עמוקה יותר כך היא משיגה תוצאות טובות יותר, ניסו לבנות רשתות עם מאות שכבות, אך הן השיגו תוצאות פחות טובות מהרשתות הקודמות שהיו בעלות סדר גודל של 20 שכבות. הבעיה המרכזית של

הרשתות העמוקות נבעה מכך שלאחר מספר שכבות מסוים התקבל ייצוג מספיק טוב, ולכן 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 המרכזית 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 פותחה רשת בעלת 152 שכבות אשר הציגה 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 המסננים והורדה של הממד פי שניים בעזרת pooling. ההיפר-פרמטרים הם: Batch normalization, lr=0.1, SGD+momentum=0.9, Xavier initialization, אחריו כל שכבת קונבולוציה, אתחול משקלים בשיטת validation error-שה מתיישר, weight decay=1e-5 ו-batch size=256.

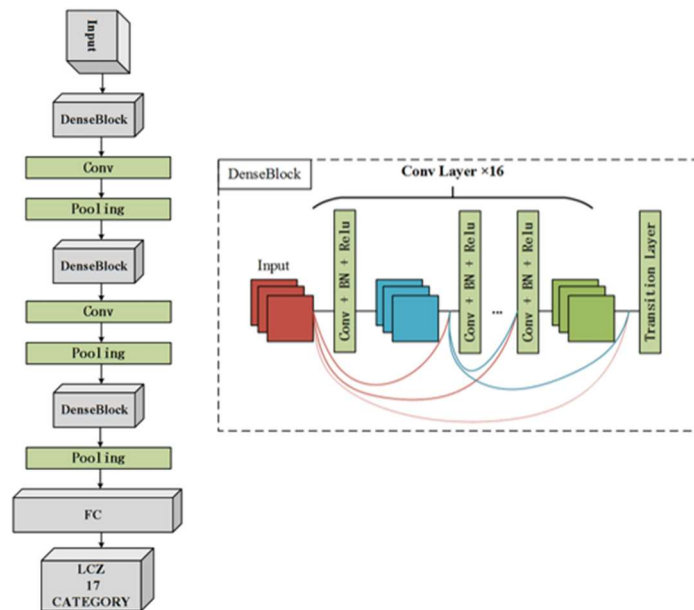
רשתות מתקדמות יותר שילבו את גישת ה-inception עם ResNet על מנת לשלב בין היתרונות של שתי השיטות.



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

5.2.5 Densely Connected Networks (DenseNet)

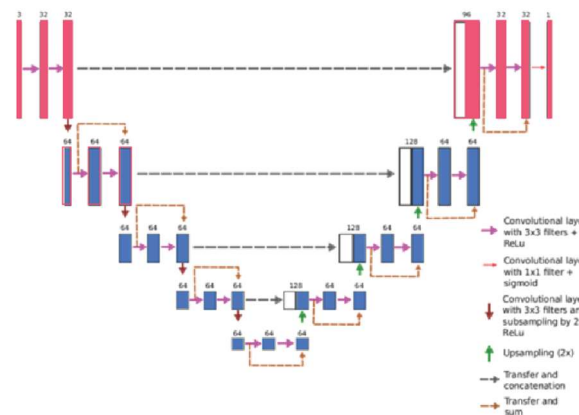
ניתן להרחיב את הרעיון של Residual Block כך שלא רק מחברים את הכניסה של כל בלוק למוצא שלו, אלא גם שומרים את הכניסה בפני עצמה, ובודקים את היחס שלה לשכבות יותר עמוקות. Dense block הוא בלוק בעל כמה שכבות, הבנוי כך שכניסה של כל שכבה מחוברת לכל הכניסות של השכבות אחריה. ניתן כמובן לשרשר כמה בלוקים כאלה יחד ולבצע ביניהם כל מיני פעולות כמו pooling או אפילו שכבת קונבולוציה עצמאית. כיוון שמשלבים כמה כניסות של בלוקים שונים, יש בעיה של התאמת ממדים, משום שכל בלוק מגדיל את מספר הערוצים, חיבור של כמה בלוקים יכולים ליצור מודל מורכב מדי. כדי להתגבר על בעיה זו הוספו שכבות transition בסוף כל dense block המבצעות קונבולוציות 1×1 עם רוחב צעד $s = 2$, ובכך מספר הערוצים נותר סביר והמודל לא נעשה 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 המאפיינים שבתמונה ועל פיהם לקבוע מה יש בתמונה, אלא צריך גם לשחזר את מיקומי הפיקסלים והתיגים שלהם ביחס לתמונה 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 הווקטור שהתקבל במוצא צוואר הבקבוק יחד עם המידע שנשאר מהחלק הראשון של התהליך. פונקציית המחיר שמשמשים ברשת זו נקראת pixel-wise cross entropy loss, הבודקת כל פיקסל ביחס ל-label האמיתי אליו הוא שייך.

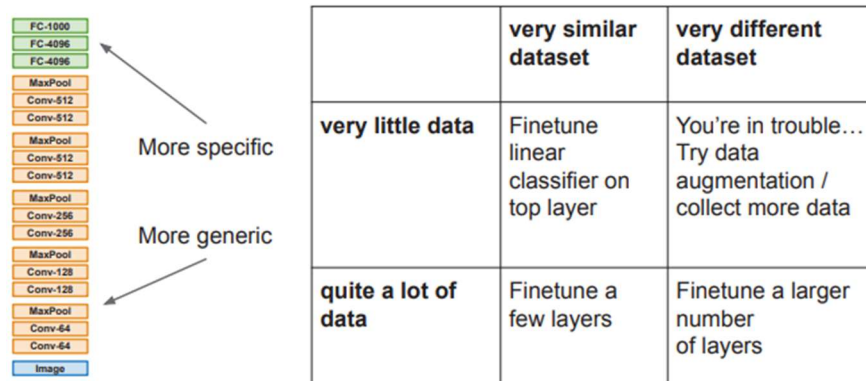


איור 5.14 ארכיטקטורת U-Net.

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 טוען שעבור כמעט כל סוג דאטה השכבות הראשונות לומדות אותו דבר (זיהוי שפות, קווים וצורות כלליות, מאפיינים כלליים וכו') ולכן ניתן להשתמש בהן פעמים רבות ללא שינוי כלל. משום כך, בפועל בדרך כלל לוקחים רשת קיימת ומחליפים בה את השכבות האחרונות או מוסיפים לה עוד שכבות בסופה, ואז מאמנים את השכבות החדשות על הדאטה החדש כך שהן תהיינה

מכוונות 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. כמו כן, משום שבשיטה זו נדרשים לאמן מספר שכבות קטן יותר, קטן ה-overfitting הנובע ממחוסר בדאטה.



איור 5.15 Transfer Learning.

5. References

Convolutional:

<https://github.com/technion046195/technion046195>

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>