שיעור 5

14 במאי 2019

1 רשתות קונבולוציה

בשיעור זה נעסוק בהתמודדות עם בעיות ממשפחת הראייה הממוחשבת ועיבוד התמונה: אלו בעיות בהן לפתרונות מבוססי למידת מכונה בכלל ורשתות נוירונים בפרט יש יתרון משמעותי - אכן, בעולם בעיה זה האלגוריתמים שמגיעים למדדים הטובים ביותר מכילים שימוש נרחב ברשתות נוירונים, כמעט באופן בלעדי.
נניח שאנו רוצים לחזות, עבור תמונה, האם היא תמונה של חתול. בשיעור הקודם למדנו על הארכיטקטורה הבסיסית ביותר של רשתות נוירונים - רשת קשירה לחלוטין (fully connected). כיצד היינו יכולים לעשות זאת באמצעות הארכיטקטורה שתיארנו? היינו מתייחסים לתמונה כאל וקטור ארוך, כאשר כל איבר בו הוא ערך התמונה בפיקסל. היינו מעבירים את הוקטור דרך הרשת לקבלת חיזוי. אילו בעיות עשויות להיווצר כתוצאה מכך?
לתמונות כ-data יש כמה מאפיינים יחודיים שאיננו מתחשבים בהם בפתרון זה. למשל, הפתרון אינו אינווריאנטי בכלל להזזות של התמונה: אם החתול שבתמונה יוזז בפיקסל אחד ימינה, נקבל וקטור שונה לגמרי, אף שמהותית ה"שינוי" בתמונה הוא קטן מאוד. באותו האופן, הפתרון שלנו אינו רובוסטי לשינויים נוספים במרחב (סיבוב, ניפוח) או לשינויים בצבעים (תאורה, צל).
בנוסף, בפתרון שלנו לא ניצלנו ידע רב שיש לנו על עולם התמונות באופן כללי: אנו יודעים כי לתמונה יש תכונות מקומיות משמעויות - למשל, עבור פיקסל מסוים, סביר שהפיקסלים הסמוכים אליו יהיו בצבעים דומים. אנו מצפים, בתמונות טבעיות, למעין חלקות למקוטעין בצבע הפיקסלים. אם נניח אותה, באופן כלשהו, נוכל לחסוך בכמות הפרמטרים ברשת. אכן, שימו לב כי ברשת fully connected של תמונות, שבהחלט עשויות להיות בגודל $\sim 10^4$ פיקסלים, כמות המשקולות גדלה במהירות.

1.1 משימות מרכזיות בעיבוד תמונה

המשימות בעיבוד תמונה הן מגוונות, ובאופן כללי נוכל לחלק אותן לכמה משפחות:

- משימות סיווג - לכל תמונה נרצה לבחור קטגוריה אליה היא שייכת, ולאפיין מה יש בתמונה.
- משימות איתור אובייקטים - במשימות אלו נרצה למצוא אובייקטים בתמונה.
- משימות סגמנטציה - במשימות אלו נרצה חתחום במדויק אובייקט בתמונה.
- משימות זיהוי והתאמה - במשימות אלו נרצה לדעת האם שתי תמונות מתארות אותו האובייקט. למשל, בזיהוי פנים של אדם מתוך מאגר, או בקישור בין פנים של אדם בכמה תמונות.

1.2 קונבולוציה

1.2.1 קונבולוציה דיסקרטית חד־מימדית

נזכר (?) באופרטור אהוב וחשוב: הקונבולוציה. ניתן להגדיר קונבולוציה במגוון הקשרים, ואנו נעסוק כאן בקונבולוציה דיסקרטית, בין שני וקטורים או בין שתי מטריצות. עבור וקטורים u, v באורכים n, m בהתאמה נגדיר את

$$(u * v)_i = \sum_j u_j \cdot v_{i-j}$$

כאשר עבור j או $i - j$ החורגים מגודל הוקטורים נציב 0. כיצד נחשוב על $(u * v)_i$? נחשוב על כל הדרכים לקבל את האינדקס i כסכום של שני אינדקסים מהוקטורים u, v, ונסכום את המכפלות בין ערכי האינדקסים הללו.
נחשוב על הקונבולוציה באופן נוסף: עבור הוקטורים u, v, נהפוך את הוקטור v,

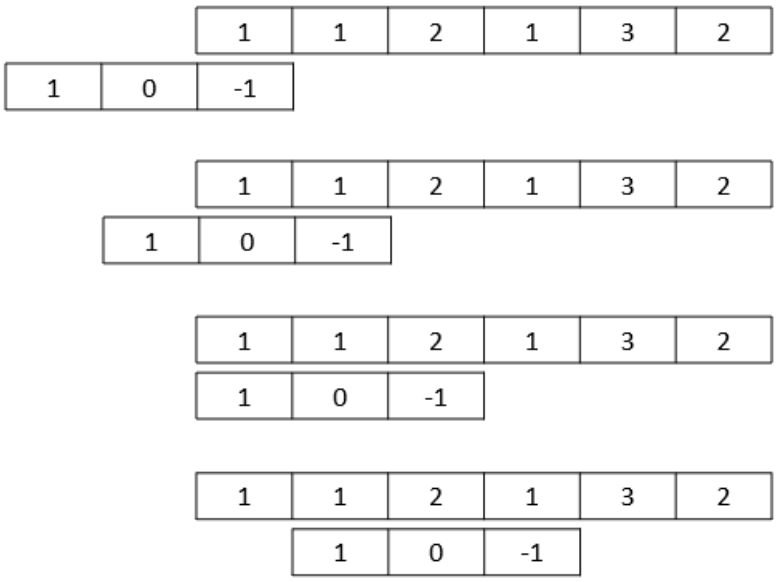
$$\tilde{v} = (v_m, ..., v_0)$$

ו"נעביר" את הוקטור על־פני הוקטור u, ונחשב את המכפלה הפנימית ביניהם:

$$(u * v)_i =< u[i - m : i], \tilde{v}[0 : m] >$$

כאשר אם האינדקס לא מוגדר נגדיר 0.
נראה דוגמא: עבור הוקטור $u = (1, 1, 2, 1, 3, 2)$ והוקטור $v = (1, 0, -1)$ נקבל כי $\tilde{v} = (-1, 0, 1)$ והקונבולוציה נתונה כך:

$$\begin{aligned}
(u * v)_0 &= 0 \cdot (-1) + 0 \cdot 0 + 1 \cdot 1 = 1 \\
(u * v)_1 &= 0 \cdot (-1) + 1 \cdot 0 + 1 \cdot 1 = 1 \\
(u * v)_2 &= 1 \cdot (-1) + 1 \cdot 0 + 2 \cdot 1 = 1 \\
(u * v)_3 &= 1 \cdot (-1) + 2 \cdot 0 + 1 \cdot 1 = 0 \\
(u * v)_4 &= 2 \cdot (-1) + 1 \cdot 0 + 3 \cdot 1 = 1 \\
&\vdots
\end{aligned}$$



איור 1:

1.2.2 קונבולוציה רב מימדית

כזכור, אנו מתעניינים בתמונות. לרוע המזל, לא נוכל לחשוב עליהן כעל וקטורים חד־מימדיים, אלא כעל מטריצות, או אף כעל "תיבות", ונרצה להיות מסוגלים לדבר על קונבולוציה רב מימדית. עבור שתי מטריצות, A, B, נגדיר

$$(A * B)_{i,j} = \sum_k \sum_l A_{k,l} B_{i-k,j-l}$$

כפי שאתם רואים, ההגדרה דומה מאוד לזו החד־מימדית. אנו עוברים על כל האפשרויות לבחירה של אינדקסים כך שסכומי האינדקסים יתנו את (i, j). באותו האופן נוכל להגדיר קונבולוציה עבור "תיבות" ("טנזורים"). יהיו A, B מערכים n־מימדיים. נגדיר

$$(A * B)_{i_1,\ldots,i_n} = \sum_{j_1,\ldots,j_n} A_{j_1,\ldots,j_n} \cdot B_{i_1-j_1,\ldots,i_n-j_n}$$

הרעיון, כאמור, דומה מאוד לקונבולוציה חד־מימדית, כאשר אנו "מעבירים" תיבה אחת על־פני השנייה ומחשבים מכפלות פנימיות.

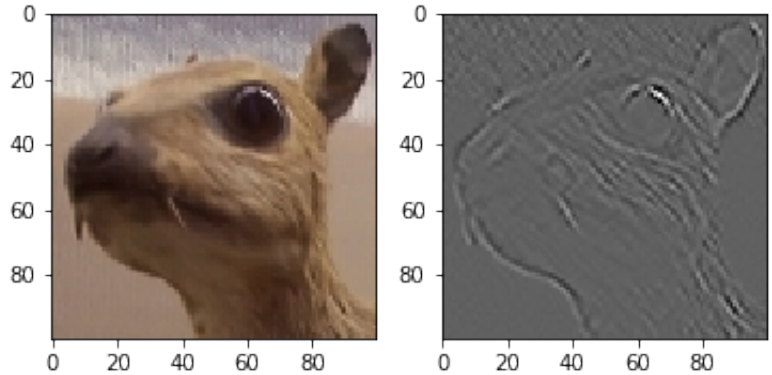
1.3 שיטות בעיבוד תמונה קלאסי

הקונבולוציה היא כלי שימושי ביותר, כבר בעיבוד תמונה קלאסי. טכניקות רבות בעיבוד תמונה מערבות קונבולוציה של התמונה עם "פילטרים" שונים. ראשית נראה דוגמא. בתמונות

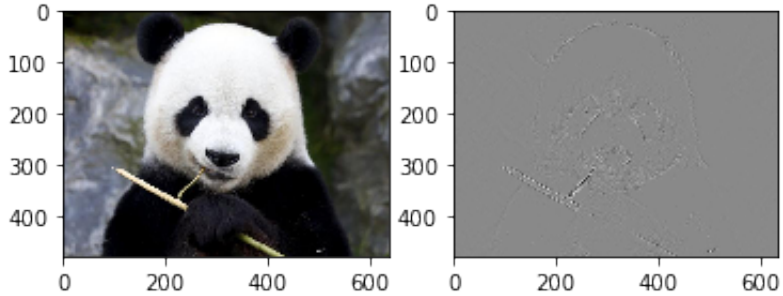
הבאות, ביצענו קובנולוציה של התמונה עם ה"פילטר", כלומר המטריצה,

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

פעולה זו משמשת, באופן מסורתי, ל־edge detection. למעשה "החלקנו" את הפילטר על התמונה, ובכל מקום, כפלנו אותו בסביבה אותה בחנו. קיבלנו כי סכומים גדולים התקבלו כאשר ההפרש בין הערך במרכז הפילטר לערכים בקצותיו היו גדולים. כלומר, באזורים של שינויים חדים של צבע בתמונה - אלו בדיוק ה"קווים המפרידים" בין אובייקטים בתמונה.



איור 2:



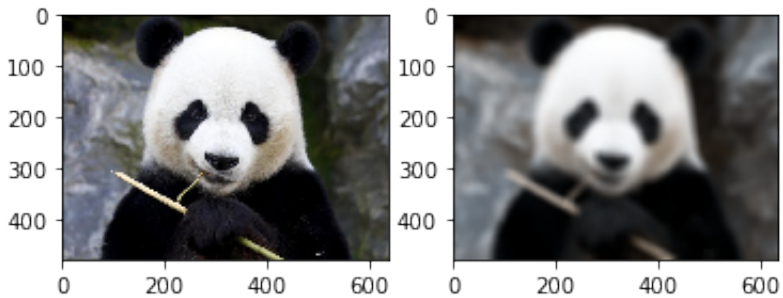
איור 3:

קיבלנו פירוש חדש לפעולת הקונבולוציה: בקונבולוציה אנו עוברים על התמונה המקורית עם פילטר, ובתמונה החדשה, ערך כל פיקסל יתאר לנו כמה הסביבה של הפיקסל מתאימה לפילטר. ככל שהסביבה דומה יותר לפילטר, כך ערכו יהיה גבוה יותר. תחת הפילטר בו השתמשנו קיבלנו תמונה חדשה, בה כל פיקסל מתאר עד כמה הפיקסל המקורי היה בסביבה של "קו".

נראה דוגמא לפילטר נוסף: פילטר "החלקה":

$$\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

זהו פילטר עבורו כל פיקסל יהפוך להיות ממוצע הפיקסלים שסביבו. פעולתו נראית כך:



איור 4:

1.4 רשתות קונבולוציה

ברשתות קונבולוציה נעשה התאמות לרעיון של "פילטרים", אלא שהפעם נרצה ללמוד את הפילטרים המתאימים לבצע איתם קונבולוציה. רשת קונבולוציה בנויה משכבות קונבולוציה. ראשית נבין כיצד פועלת שכבת קונבולוציה מאומנת. שכבת קונבולוציה מקבלת כקלט טנזור תלת־מימדי בגודל $n_1 \times m_1 \times c_1$ ומחזירה טנזור תלת־מימדי בגודל $n_2 \times m_2 \times c_2$. אל c_1 נתייחס כאל מספר ערוצי הקלט, ואל c_2 נתייחס כאל מספר הפילטרים: בקרוב נבין מדוע. שכבת קונבולוציה מורכבת מפילטרים, טנזורים איתם אנו מבצעים קונבולוציה. נהוג שהאורך והרוחב של כל הפילטרים בשכבה יהיה קבוע - למשל, 3×3. השכבה תכיל c_2 פילטרים, טנזורים, מהצורה

$$3 \times 3 \times c_1$$

השכבה פועלת כך: לכל פילטר שמוגדר בה, היא מחשבת את הקונבולוציה של הקלט איתו. כל קונבולוציה שכזו תהיה בגודל

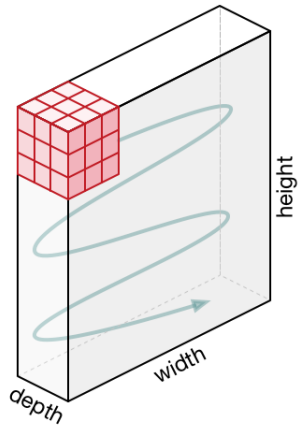
$$n_2 \times m_2 \times 1$$

הפלט של כל קונבולוציה שכזו הוא תמונה חדשה, ואנו מקבלים c_2 תמונות שכאלו. השכבה תיקח את כל התמונות שתתקבלנה ותחזיר טנזור המכיל את כולן, וגודלו

$$n_2 \times m_2 \times c_2$$

הביטוי המתמטי מתקבל כך: בהינתן טנזור הקלט A והפילטרים B_l, הפלט C הוא

$$C_{i,j,l} = (A * B_l)_{i,j}$$



איור 5:

בשלב זה מומלץ לעבור ולהתבונן באנימציות המצורפות.
זהו הבסיס לשכבת קונבולוציה. נהוג להוסיף עבור כל פילטר בשכבה גם איבר bias - כך שהביטוי שלנו יהפוך ל-

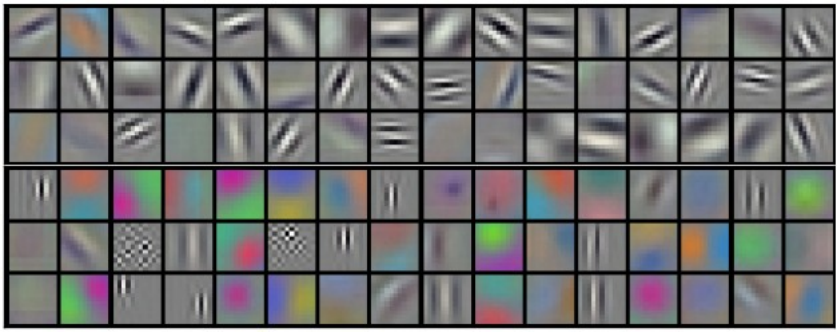
$$C_{i,j,l} = (A * B_l)_{i,j} + b_l$$

כמובן כי בשלב זה, הביטוי שאנו מתבוננים בו, עם כל מורכבותו, הוא עודנו אפיני (ליניארי עם מקדם חופשי) בקלט. נוסיף פונקציית אקטיבציה לקבלת הביטוי הסופי

$$C_{i,j,k,l} = \sigma((A * B_l)_{i,j,k} + b_l)$$

זו, אם-כן, היחידה הבסיסית ברשת קונבולוציה. המשקולות אותן אנו מתאימים הן אוסף המשקולות המרכיבות את הפילטרים ואת ה-bias. כיצד נבצע back propagation? ובכן, ניתן לפתוח את הביטוי בקונבולוציה ולקבל צירוף ליניארי של איברי הקלט A. אותו נוכל לגזור ממש כשם שגזרנו את הביטוי במקרה של fully connected networks.
מה קיבלנו? ראשית, שימו לב שאנו מקבלים טנזור "גדול" (עם אלפי פיצ'רים) ומחזירים טנזור גדול, אך משתמשים בכמות קטנה מאוד של משקולות - כמות המשקולות תלויה בעיקר בכמות הפילטרים ובגודלם, וכמעט ואינה תלויה בגודל הקלט. בנוסף, אנו משתמשים בתכונות מקומיות בתמונה, ולכל פיקסל בתמונה אנו מתחשבים בשכנים שלו בחישובים שאנו מבצעים עליו.
שימו לב כי שכבת הקבונולוציה שהגדרנו מקבלת טנזור, תיבה תלת-מימדית, ומחזירה תיבה תלת-מימדית. באופן כללי, העבודה עם תמונות היא, בניגוד לאינטואיציה, עבודה עם טנזורים שיכולים להיות גם מאוד "עמוקים" - אין זה נדיר לקבל בעומק הרשת טנזור עם ~ 100 ערוצים.
רשתות קונבולוציה שואבות השראה באופן ישיר מהפילטרים בעיבוד תמונה קלאסי, אלא שכאן אנו לומדים את הפילטרים הטובים ביותר על-סמך סט האימון, במקום להנדס אותם באופן ידני וקפדני. פעמים רבות הפילטרים שהרשת לומדת מזכירים פילטרים מעיבוד תמונה קלאסי.

איור 6:



מה הם n_2, m_2? ובכן, ישנן כמה דרכים להגדיר קונבולוציות ולקבל גדלים שונים. בהגדרה שראינו, n_2, m_2 גדולים מ־n_1, m_1. לרוב נהוג לקחת קונבולוציה כך שב"מכפלות הפנימיות" אין צורך להוסיף אפסים, כפי שעשינו, כך שמקבלים כי n_2, m_2 קטנים יותר מ־n_1, m_1.

רשת קונבולוציה, כפי שניתן היה לנחש, מורכבת משכבות קונבולוציה, כאשר אחרי כל אחת מהן מופיעה אקטיבציה לא ליניארית. ההערמה של שכבות קונבולוציה האחת על השנייה מאפשרת יצירה של תבניות מורכבות יותר ויותר. כך למשל, השכבות הראשונות לומדות פילטרים פשוטים, של גילוי קצוות או תבניות בסיסיות. השכבות הבאות משתמשות באינפורמציה "איכותית יותר". אם השכבה הראשונה משתמשת בצבע של פיקסלים, הרי שהשכבה השנייה משתמשת במידע כמו, באיזו מידה פיקסל הוא "קצה של אויבקט" או באיזו מידה הוא דומה לסביבה שלו. שכבות עמוקות יותר ברשת יכולות לאפיין מעגלים, עיניים או את הזנב של החתול מהדוגמא הראשונה שלנו.

הערה אחרונה: כשהגדרנו את פעולת הקונבולוציה, הדבר עירב "היפוך" של הפילטר, ואז החלקה שלו לאורך התמונה, עם מכפלות פנימיות בכל צעד. כיוון שהפילטרים ברשת קונבולוציה נלמדים, יהיה קל יותר "להשמיט" את ההיפוך כשנחשוב על קונבולוציות: אכן, הדבר לא משפיע על הלמידה כלל, נוכל ללמוד את הפילטר "ההפוך" באותה קלות בדיוק בה נלמד את הפילטר לפני "הפיכה".

1.4.1 אקטיבציה ברשתות קונבולוציה ומנהגים נוספים

הזכרנו אקטיבציה לא ליניארית ברשתות קונבולוציה. ברשתות אלו משתמשים כמעט תמיד באקטיבציית Relu:

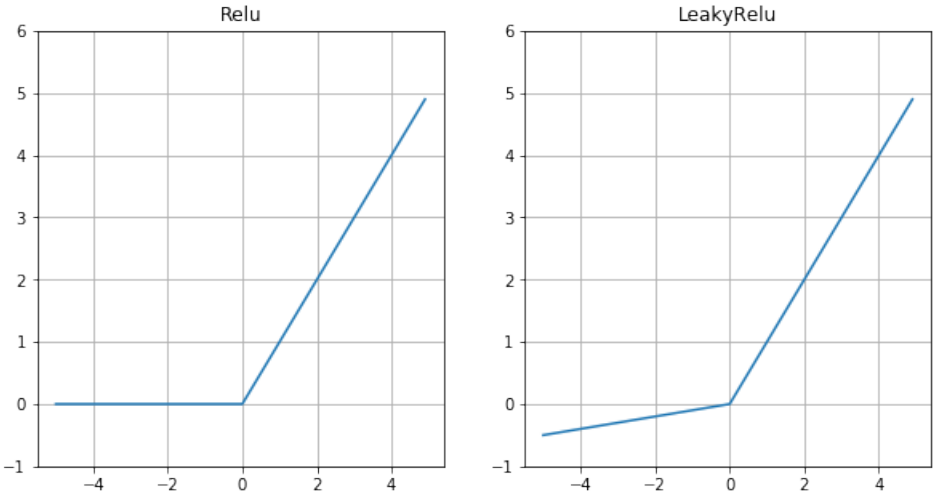
$$Relu(x) = max(0, x)$$

המוטיבציה אליה מגיעה מהאופן בו אנו חושבים על תמונה, כאשר פיקסלים לא מקבלים ערכים שליליים. שימו לב ש־Relu, אף שהיא ליניארית למקוטעין, איננה פונקציה ליניארית בשום אופן, והרכבות של פונקציות Rele בתוך הרשת יכולות ליצור פונקציות מורכבות מאוד. ל־Relu יש מאפיין שאנו עשויים שלא לאהוב - יש בה תחום שלם בו הגרדיאנט מתאפס. פירוש הדבר הוא שלגרדיאנטים "קשה יותר" לעבור דרך הרשת, הם עשויים להתאפס בתהליך ה־back propagation. לכן יש המשתמשים בפונקציית ה־LeakyRelu:

$$LeakyRelu(x) = \begin{cases} x & x \geq 0 \\ 0.1 \cdot x & x < 0 \end{cases}$$

שמדמה את ההתנהגות של Relu אך לא מאפסת גרדיאנטים.

איור 7:



באימון של רשתות קונבולוציה משתמשים לרוב בפילטרים קטנים - 3×3, או 5×5. ככלל אצבע, נהוג להגדיל את כמות הפילטרים עם כמות השכבות ברשת. מובן שיש לכללים מהסוג הזה יוצאים מן הכלל רבים. ברשתות עמוקות מגיעים, כאמור, גם למאות פילטרים בשכבה. ברשתות שכאלו משתמשים בעיקר למשימות סיווג תמונות מורכבות. אחת מהמשימות לפיהן נהוג למדוד ארכיטקטורות בראייה ממוחשבת היא משימת סיווג תמונות ממאגר ImageNet - זהו מאגר גדול של תמונות המחולקות ל-1000 קטגוריות שונות.

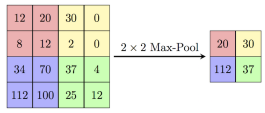
1.4.2 שדה ראייה

ברשתות קונבולוציה יש טעם לדבר על שדה הראייה של נוירונים: החלק מהתמונה המקורית שנוירון מסוים מסוגל לקבל ממנו מידע. ב"שכבה 0", בקלט, כל נוירון "רואה" נוירון יחיד - את עצמו. אחרי קונבולוציה עם פילטר בגודל 3×3, כל נוירון רואה "טלאי" בגודל 3×3 נוירונים. אם נוסיף שכבה נוספת של קונבולוציה עם פילטרים באותו הגודל, כל נוירון בשכבה זו יראה טלאי בגודל 4×4 מהתמונה המקורית (למה?). לכן, ככל שאנו מעמיקים ברשת נוירונים הם בעלי "שדה ראייה", receptive field, גדול יותר. בשכבות עמוקות אנו עשויים להגיע למצב בו כל נוירון "רואה" את כל התמונה המקורית. שימו לב כי בשימוש בשכבת fully connected, מלכתחילה כל נוירון "ראה" את כל הנוירונים האחרים.
ככלל, ככל ששדה הראייה של הנוירונים גדול יותר, הם מסוגלים לבטא תבניות מורכבות יותר, או גלובליות יותר, בתמונה. בפרט, אם נוירון מסויים מחפש מעגלים ברדיוס 20 פיקסלים, נוכל לקבל חסם מלמטה לשכבה בה הוא יהיה מסוגל לבצע פעולה שכזו.

1.4.3 Max Pooling

שכבת קונבולוציה היא אחת משתי שכבות שנהוג למצוא בשכבות קונבולוציה. השנייה היא שכבת pooling, ואופן פעולתה פשוט למדי. שכבת ה-pooling הפשוטה ביותר מחלקת את

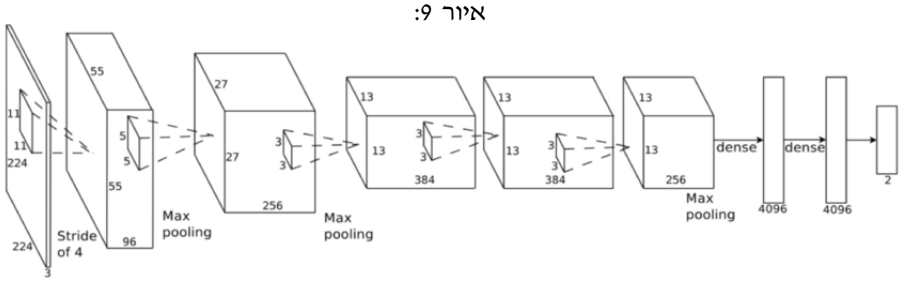
התמונה לקבוצות של $m \times m$ פיקסלים צמודים (למשל, $m = 2$) ולוקחת מכל קוביה שכזו את הפיקסל בעל הערך הגבוה ביותר. כלומר, זו מעין דגימה של התמונה ברזולוציה יותר נמוכה.



איור 8:

הדגימה הזו שימושית ומעשית מאוד שכן היא מקטינה משמעותית את גודל התמונה, ובכך מאפשרת לנו לעבוד בנוחות עם כמות פילטרים הולכת וגדלה. ה"פילוסופיה" בשימוש בשכבת pooling היא שהמידע "המעניין" שמתקבל אחרי רשת קונבולוציה מקיים תכונות של מקומיות, לוקאליות, וכי אין צורך בכל המידע שחושב כדי לאמן רשת טובה.
רשת קונבולוציה טיפוסית היא הרכבה של שכבות קונבולוציה עם שכבות pooling ביניהן. ה"בלוקים" מביאים טנזורים עם פחות ופחות נוירונים, או פיקסלים. לקראת סוף הרשת נהוג "לשטח" את הטנזור ולהשתמש בשכבות fully connected לסיום פעולת הרשת - למשל, כדי לחזות לאיזו מבין 1000 הקטגוריות האפשריות התמונה שייכת. שימו לב כי אף שאלו השכבות האחרונות, פעמים רבות כמות המשקולות הגדולה ביותר תהיה דווקא בשכבות ה־fully connected הללו. יש בכך כדי להדגים עד כמה חסכוניות שכבות הקונבולוציה.
את הקונבולוציות, כמו גם את ה־pooling, אין הכרח לעשות ב־stride=1. כלומר, ניתן להזיז את ה"פילטר" יותר מפיקסל אחד בכל תנועה, דבר המאפשר גם כן להקטין את מימדי הטנזורים איתם עובדים במהירות.
בתמונה ניתן לראות ארכיטקטורה מאוד בסיסית בשם AlexNet, שנוסחה באוניברסיטת טורונוטו ב־2012 והגיעה להישגים של state of the art בשעתה (אגב, תופתעו לשמוע כמה חזקה אוניברסיטת טורונטו בלמידת מכונה).

איור 9:



ברשתות מודרניות יש כבר מאות שכבות עם מיליוני פרמטרים (ומגוון דרכים להקטין את כמות הפרמטרים ברשת ולאפשר אימון פשוט יותר).

1.5 Transfer learning

כפי שראינו, רשתות קונבולוציה עשויות לגדול לגדלים מפלצתיים, ובפרט הרשתות המודרניות והחזקות שבהן. לרוב יש לכך מחיר - האימון שלהן מצריך כמויות עצומות של מידע, משאבים

וזמן. פעמים רבות לא יהיו ברשותנו אף אחד מהם. בנוסף, במקרים רבים נרצה לאמן רשת המתאימה למידע מסוג מסוים - למשל, נרצה לסווג מכוניות בתמונות שנלקחו ממל"טים. מצד אחד, ברור כי יש בתמונות הללו מאפיינים יחודיים, ולא נוכל למצוא "רשת מדף" שתפתור לנו את הבעיה מבלי שהיא התאמנה על המידע הייחודי שלנו. מצד שני, ברור כי יש מאפיינים משותפים לתמונות בסט האימון שלנו ולתמונות של אובייקטים מציאותיים אחרים. במקרים כאלו ניתן להשתמש בטכניקה של transfer learning: בטכניקה זו אנו לוקחים רשת מאומנת, שאומנה מראש על כמויות רבו של מידע, אך לאו דווקא המידע שלנו. למשל, ניקח רשת שאומנה לסיווג תמונות ממאגר ImageNet. נפעל תחת ההנחה כי בשלבים הראשונים של הרשת, בהם היא לומדת פילטרים בסיסיים שמתארים תמונות ואובייקטים מציאותיים באופן כללי, נוכל לעשות שימוש גם במקרה שלנו. נניח כי בשלבים הראשונים נלמדים פילטרים וטרנספורמציות "גנריים", ובשלבים האחרונים ברשת נלמדות טרנספורמציות "ספציפיות", הקשורות קשר הדוק יותר לעולם הבעיה. אכן, הנחה זו נתמכת בבדיקות אמפיריות רבות.

נוכל לעשות שימוש כמעט בכל הרשת בשלמותה, ללא אימון של המשקולות שלה, בתרחיש שלנו. כדי להתאים את הרשת באופן ספציפי לסיווג מכוניות מהאוויר נוכל לחתוך את השכבות האחרונות שלה ולהחליף אותן בשכבות חדשות. כעת נאמן אותן בלבד. בכך אנו מנצלים את האימון הארוך של הרשת המקורית בשכבות הראשונות והמקוריות - השכבות שמנתחות תמונה באופן כללי, ומאמנים מחדש שכבות יותר "ספציפיות", שמותאמות יותר למשימה מסוימת. הדבר מאפשר לנו, במובן מסוים, "לרקוד על שתי החתונות" - אנו מנצלים רשת גדולה ובעלת פרמטרים רבים, אך בפועל מאמנים רק חלק קטן שלה, ומנצלים ידע "גנרי" לגבי ניתוח של תמונות.