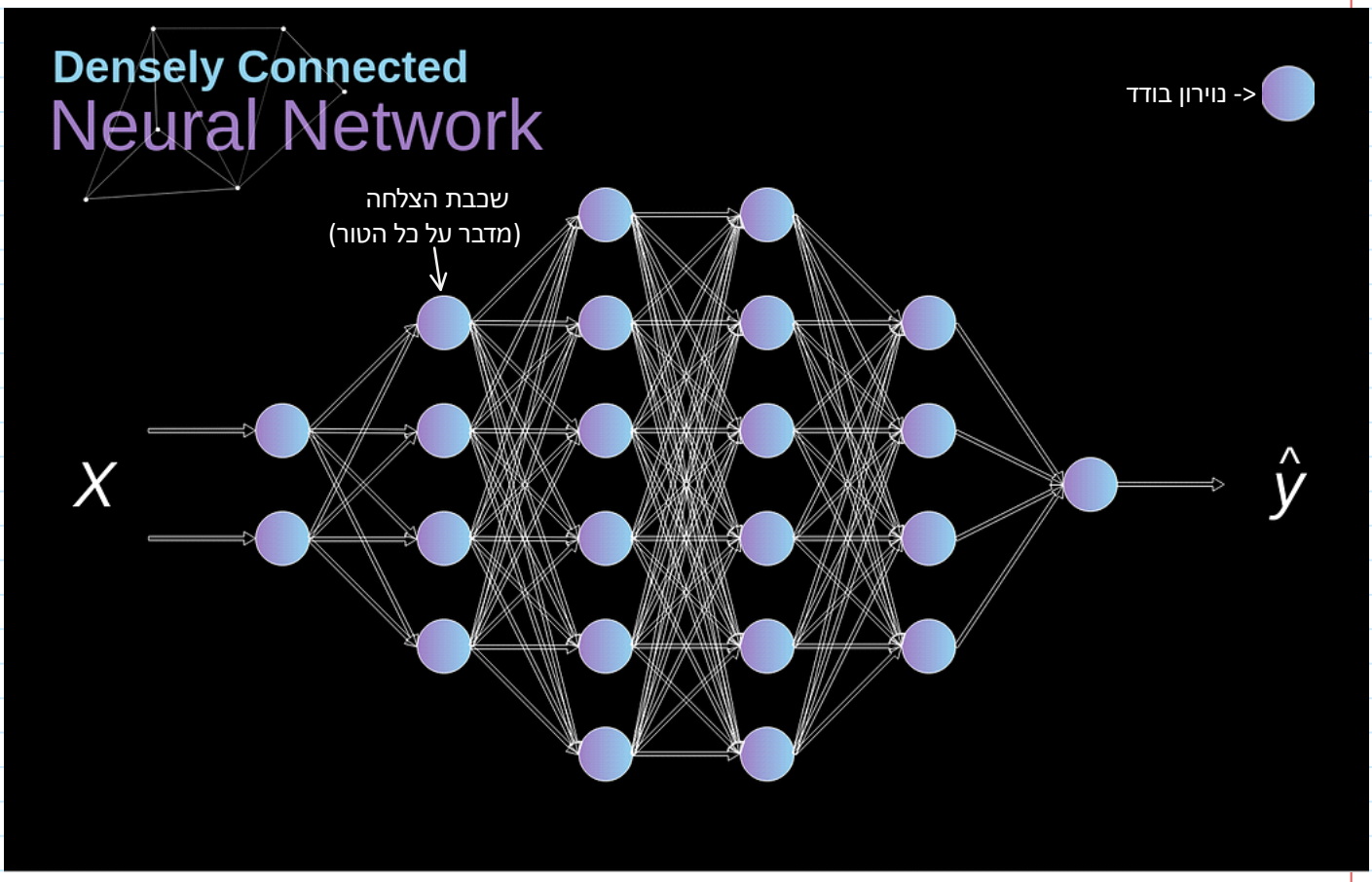
המתמטיקה שמאחורי - Convolutional Neural Networks (CNN)

12:56 PM Monday, September 2, 2024

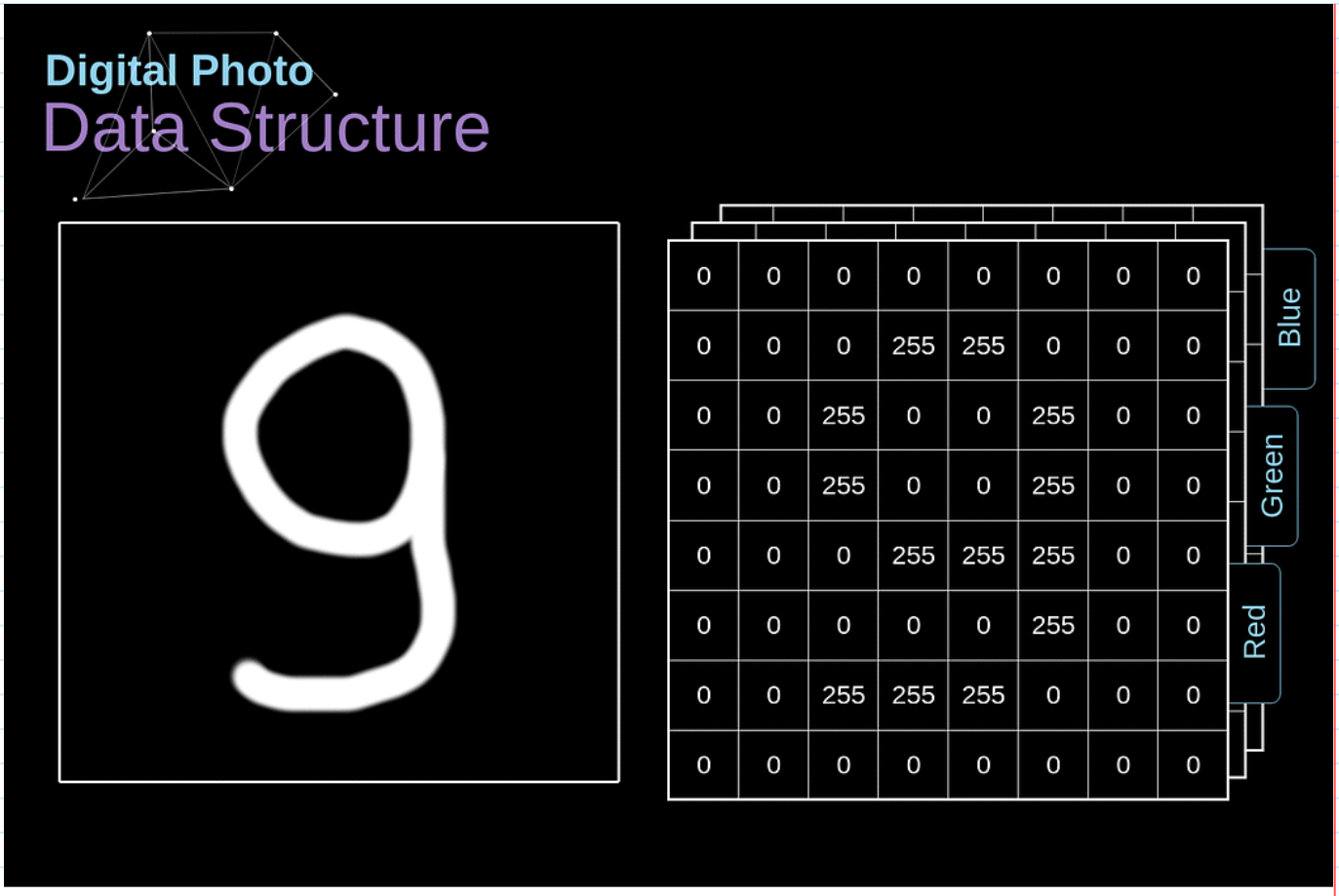
בעולם ה-AI ישנם כמה סוגים של Neural Networks, סוגים אלו התפתחו מהבסיס אשר נקרא Densely Connected Neural Networks.
שהם בעצם רשתות של נוירונים אשר מחולקים לקבוצות של שכבות הצלחה, וכל נוירון אחד בשכבה מחובר לכל אחד מהנוירונים בשכבה שלאחריו, כמי שמופיע בדוגמא הבאה:



גישה זו פועלת היטב כאשר אנו פותרים בעיית סיווג על סמך קבוצה מוגבלת של תכונות מוגדרות, לדוגמא חיזוי של תפקיד של שחקן כדורגל מסוים בהתבסס על הנתונים האישיים שלו. לעומת זאת כאשר עוסקים בעיבוד תמונה המצב מורכב יותר, זה אפשרי להתייחס לבהירות של כל פיקסל בתור Feature נפרד ולהעביר אותו בתור קלט לרשת, אך שיטה זאת תצריך כמה מיליוני נוירונים ברשת שלנו, ובמידה ונרצה להוריד את איכות התמונה נוכל לאבד מידע חשוב שיכול להשפיע על ביצועי המודל.
מכאן אנו מבינים כי ה-Densely Connected Neural Networks יצריך מאיתנו משאבים רבים ועלול לעבוד שונה באיכויות תמונה שונות, ועל כן יש למצוא רשת נוירונים חסכונית ויעילה שתעבוד גם כאשר נרצה לעשות לתמונות Upscale או Downscale, וכאן Convolutional Neural Networks באה לידי ביטוי.

מבנה דיגיטלי של תמונה -

לפני שנתייחס אל Convolutional Neural Networks, נבין תחילה כיצד תמונה מאוחסנת ומוצגת על ידי המחשב.
תמונה היא בעצם מטריצה גדולה מאוד המכילה מספרים כל מספר כזה מתאים לבהירות של פיקסל בודד. במודל RGB, תמונת הצבע מורכבת למעשה משלוש מטריצות כאלה המתאימות לשלושה ערוצי צבע - אדום, ירוק וכחול. בתמונות בשחור-לבן אנחנו צריכים רק מטריצה אחת. כל אחת מהמטריצות הללו מאחסנת ערכים מ-0 עד 255(256 ערכים מתאימים בצורה מושלמת ב-1 בייט) כאשר כל ערך מסמן גוון מסוים בין שחור-ללבן.



מה זה Convolution/Kernel Convolution? -

Kernel Convolution הוא רכיב מפתח באלגוריתמים הקשורים ל"ראיה של המחשב". זה תהליך בו אנו לוקחים מטריצה קטנה של מספרים הנקראת Kernel או Filter, מעבירים אותה על התמונה אותה נרצה לעבד וממירים אותה על סמך הערכים מה-Filter שלנו ל-Feature Map זאת באמצעות הנוסחה הבאה המתבססת על כפל מטריצות. כאשר f הוא תמונת הקלט שלנו ו-h הוא ה-Kernel והמיקומים של השורות והעמודות מסומנים על ידי m, שורות ו-n, עמודות.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

הדגמה לנוסחה נמצאת בסרטון הבא:

Kernel Convoluti...

הסבר לסרטון שמראה את השימוש בקנה מידה קטן: לאחר שממקמים את הפילטר שלנו על פיקסל, אנו לוקחים כל ערך מה-Kernel ומכפילים אותם בזוגות עם ערכים מתאימים מהתמונה. לבסוף אנו מבצעים סכום ומכניסים אותו ל- Feature Map למקום בו הוא אמור להיות.

הדגמה לנוסחה בקנה מידה גדול יותר בסרטון הבא (ביצד ה-Kernels השונים מדגישים את הצורות ההנדסיות בתמונה

בדרכים שונות):

Edge
...Detection

מה זה Valid ומה זה Same ב-Convolution -

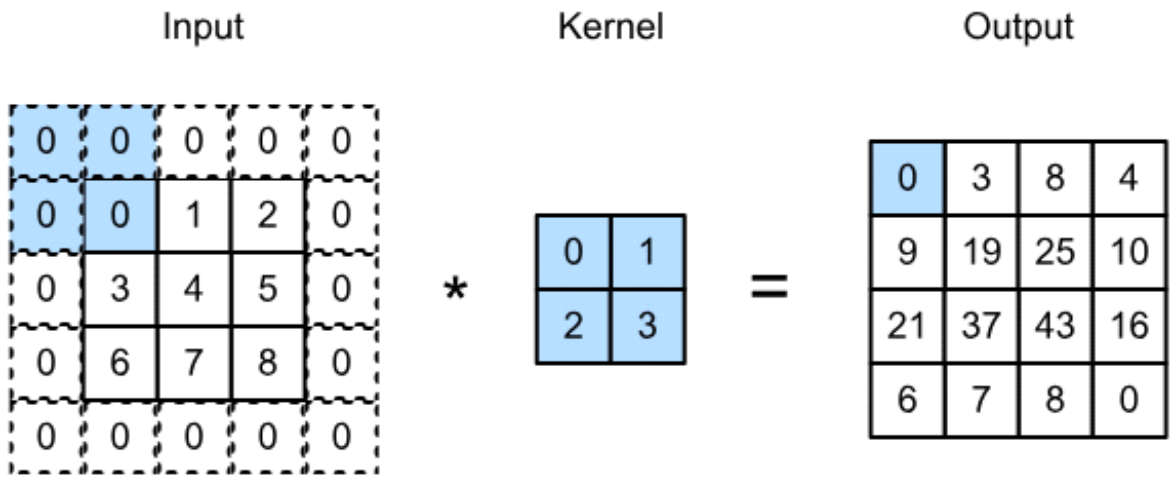
Kernel
...Convoluti

לפי הסרטון ניתן לראות כי כאשר אנו מבצעים Convolution בתמונה בגודל 6X6 עם Kernel בגודל 3X3 נקבל Feature Map בגודל 4X4. זה בגלל שיש רק 16 מקומות ייחודיים בהם ניתן להניח את Kernel שלנו בתוך התמונה.
משום שהתמונה שלנו מתכווצת בכל פעם שאנו מבצעים את ה-Convolution אנחנו יכולים לבצע את זה כמות מוגבלת של פעמים עד שהתמונה שלנו תיעלם לחלוטין. בנוסף לכך, כאשר ה-Kernel עובר על התמונה שלנו ניתן לראות כי ההשפעה של הפיקסלים הנמצאים בדופן התמונה היא קטנה הרבה יותר מהפיקסלים שבמרכז התמונה. ככה אנו מאבדים חלק מהמידע שכלול בתמונה ויכול להיות משמעותי עבורנו.

הדגמה כיצד מיקום הפיקסל משנה את השפעתו על ה-Feature Map:

Impact Of
...Pixel Posit

על מנת לפתור את הבעיות שצוינו קודם יש להוסיף מסגרת (Padding) לתמונה שלנו. לדוגמא בתמונה שמתחתינו כאשר אנחנו מוסיפים Padding בגודל פיקסל 1 לתמונה בגודל 3X3 תגדל התמונה לגודל 4X4 (ה-Padding הוא בעצם "המסגרת" של האפסים), וכאשר נבצע Kernel Convolution עם Kernel בגודל 2X2 נקבל Feature Map בגודל 4X4.



אם אנו בוחרים להוסיף Padding או לא, אנחנו מבצעים 2 סוגים שונים של Kernel Convolution אשר נקראים Valid

ו-Same.
Valid הוא כאשר אנו משתמשים בתמונה המקורית (ללא ה-Padding) ובכך ה-Feature Map קטן יותר מגודל התמונה המקורית אותה הכנסנו (ללא ה-Padding), ו-Same כאשר אנחנו משתמשים במסגרת מסביב לתמונה (עם ה-Padding) ובכך אנחנו שומרים על ה-Feature Map שהפלט שלו יהיה בגודל של התמונה המקורית אותה הכנסנו (ללא ה-Padding).
על מנת למצוא את ה-Padding הרצוי יש להשתמש בנוסחה הבאה, כאשר p מייצג את ה-Padding הרצוי ו-f זה גודל ה-Kernel/Filter.

$$p = \frac{f-1}{2}$$

מה זה Strided Convolution? -

בדוגמאות הקודמות, תמיד הזזנו את ה-Kernel שלנו הצידה בפיקסל אחד, אבל מספר הצעדים גם יכולים להשתנות לפי נוסחה.

בדוגמא הבאה ניתן לראות כיצד נראה Convolution בצעדים גדולים יותר:

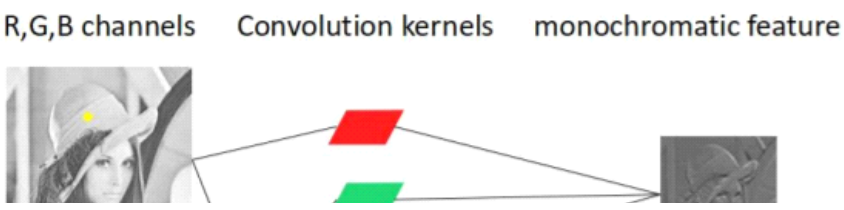
Strided Convoluti...

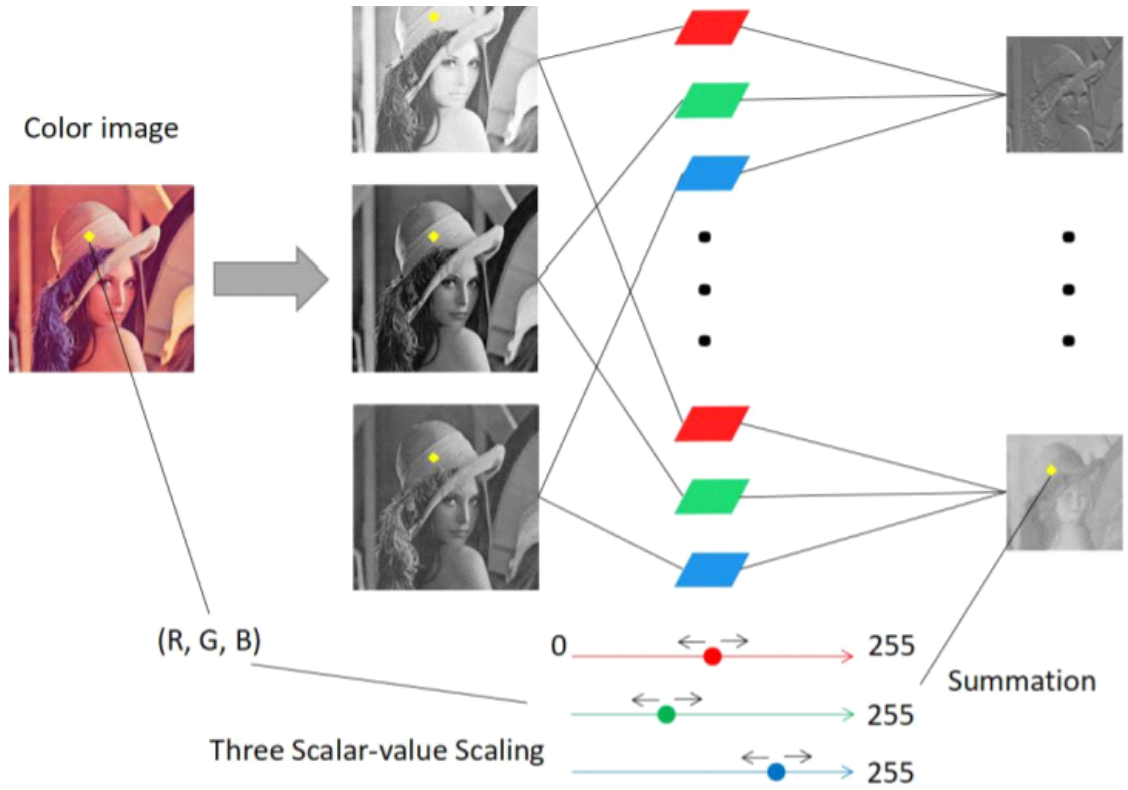
כאשר אנו מתכננים את הארכיטקטורה של ה-Convolutional Neural Networks שלנו, אנחנו יכולים להחליט האם להעלות את מספר הצעדים ובכך לקבל ערכים קטנים יותר של ה-Feature Map שלנו וגם ש-Receptive Fields שלנו יחפפו פחות.
ניתן לחשב את גודל ה-Feature Map שנקבל, תוך התחשבות ב-Padding ו-צעדים (Stride) באמצעות הנוסחה שלמטה.
כאשר n_{in} הוא גודל התמונה, p הוא ה-Padding, f הוא Filter/Kernel ו-s זהו גודל הצעדים (Stride).

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - f}{s} + 1 \right\rfloor$$

Convolution בתלת ממד -

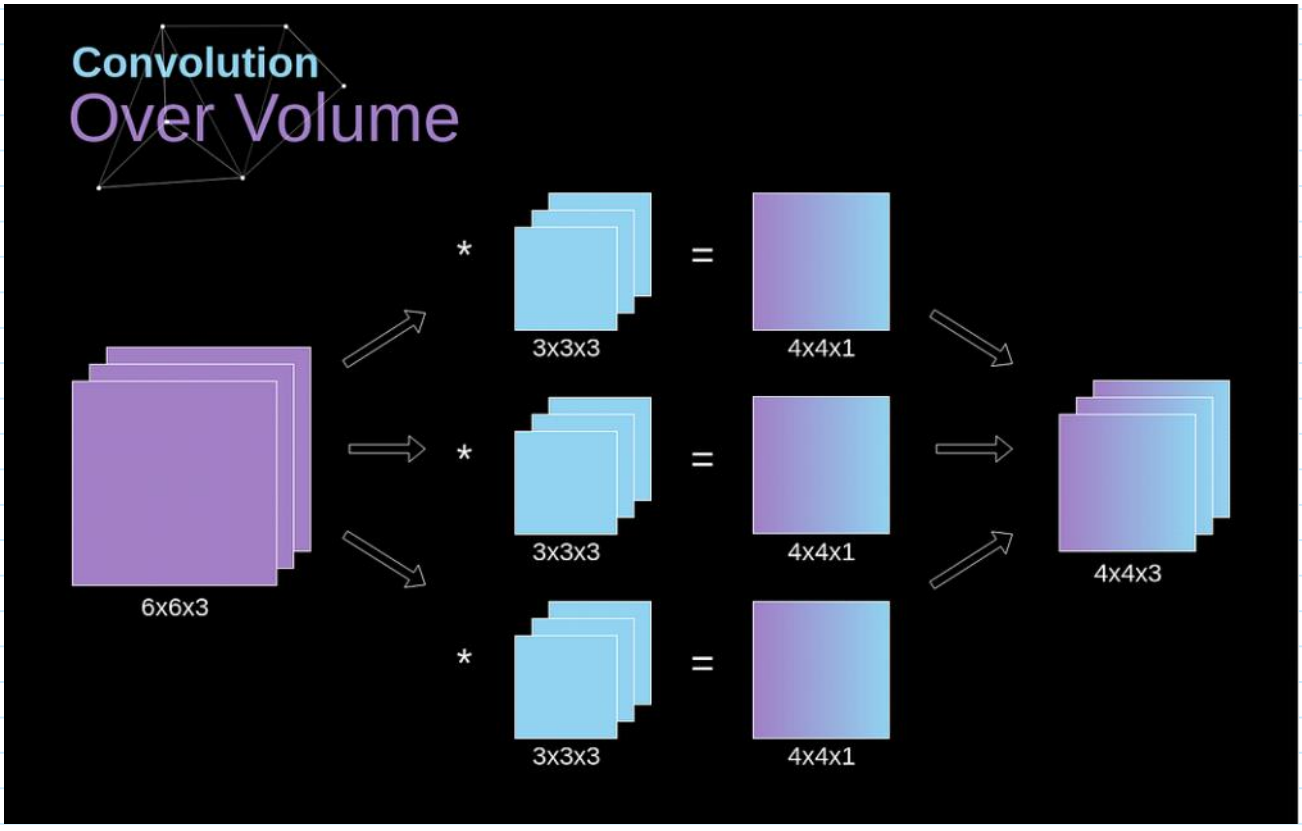
Convolution בתלת ממד הוא מושג מאוד חשוב, אשר לא רק עוזר לנו לעבד תמונות בצבע (בשונה מתמונה בשחור לבן אשר בנויה ממטריצת פיקסלים אחת שערכיה נעים בין 0-255, תמונה בצבע בנויה ממטריצה תלת ממדית המורכבת מ3 מטריצות שערכם בין 0-255),





(a) Real-valued CNN

היא גם עוזרת לנו להנחיל כמה פילטרים בשכבה אחת. החוק הראשון הוא של-Kernel ולתמונה יהיה אותה כמות של שכבות (Channels). מה שאנחנו עושים למעשה אנחנו מצמדים זוגות של שכבות בין התמונה ל-Kernel ועושים Convolution רגיל בניהם. אם נרצה להשתמש במספר פילטרים שונים על אותה התמונה, נבצע לכל פילטר Convolution בנפרד נערום את התוצאות אחד על השני ונאחד אותם לגוף אחד כפי שניתן לראות בתמונה הבאה.



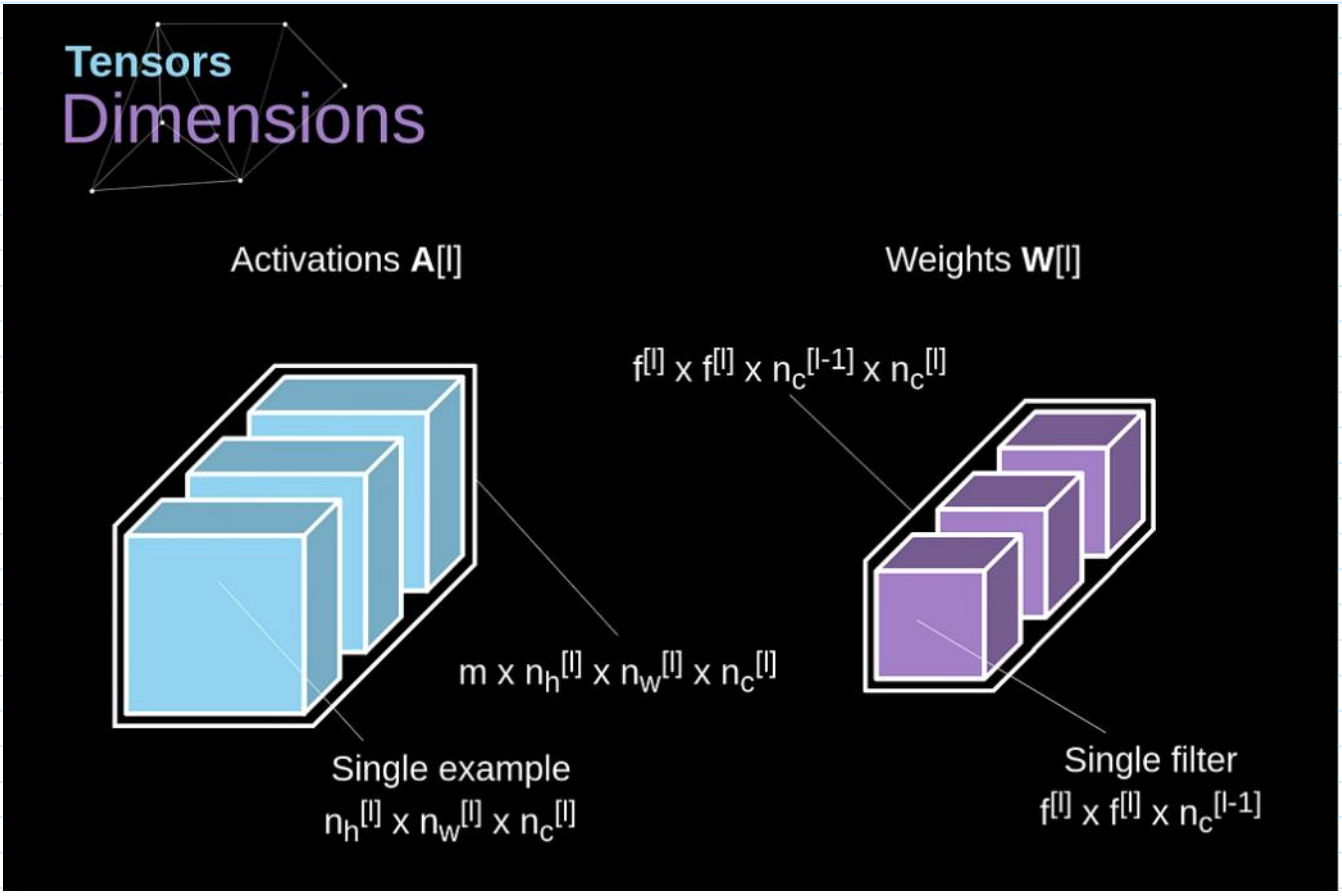
ניתן לחשב את הממדים של המטריצה התלת ממד (Tensor) שנקבל מהתהליך לפי המשוואה הבאה כאשר n הוא גודל התמונה, f גודל ה-Filter n_C מספר הערוצים בתמונה, p גודל ה-Padding, s גודל הצעדים (Stride) ו-n_f מספר הפילטרים.

$$[n,n,n_c]*[f,f,n_c] = \left[\left\lfloor\frac{n+2p-f}{s}+1\right\rfloor, \left\lfloor\frac{n+2p-f}{s}+1\right\rfloor, n_f\right]$$

שכבת Convolution -

הגיע הזמן להשתמש במה שלמדנו ולבנות שכבה אחת של Convolutional Neural Networks. המתודולוגיה שלנו כמעט זהה לזו שבסיכום המתמטיקה מאחורי רשתות נוירונים, ההבדל היחיד הוא שבמקום להשתמש בכפל מטריצה פשוט, נשתמש ב-Convolution.
כאשר אנו נבצע את ה-Forward Propagation שלנו, תחילה נחשב את ערך הביניים שלנו Z המתקבל כתוצאה מה-Convolution של נתוני הקלט מהשכבה הקודמת עם המטריצה התלת ממדית W (Tensor) אשר מכילה את הפילטרים, ולאחר מכן הוספת ה-Bias b. ולבסוף נשים את התוצאה ב-Loss Function אותה אנו רוצים.

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$



חיתוך חיבורים ושיתוף פרמטרים -

בתחילת הסיכום ציינתי ש-Densely Connected Neural Networks גרועות בעבודה עם תמונות, בגלל המספר העצום של פרמטרים אותם הם צריכים ללמוד. כעת, לאחר שהבנו מה היא Convolution, הבה נבחן כיצד היא מאפשרת לנו לייעל את החישובים.
בסרטון למטה, ה-Convolution הדו-ממדית הוצגה בצורה מעט שונה, נוירונים המסומנים במספרים 1-9 יוצרים את שכבת הקלט שמקבלת את הבהירות של הפיקסלים בתמונה, בעוד שנוירונים A-D מציינות את רכיבי ה-Feature Map המחושבים. ולבסוף הערכים I-IV הם ערכים מה-Kernel אותם נלמד במשך.

Connection
... s Cutting

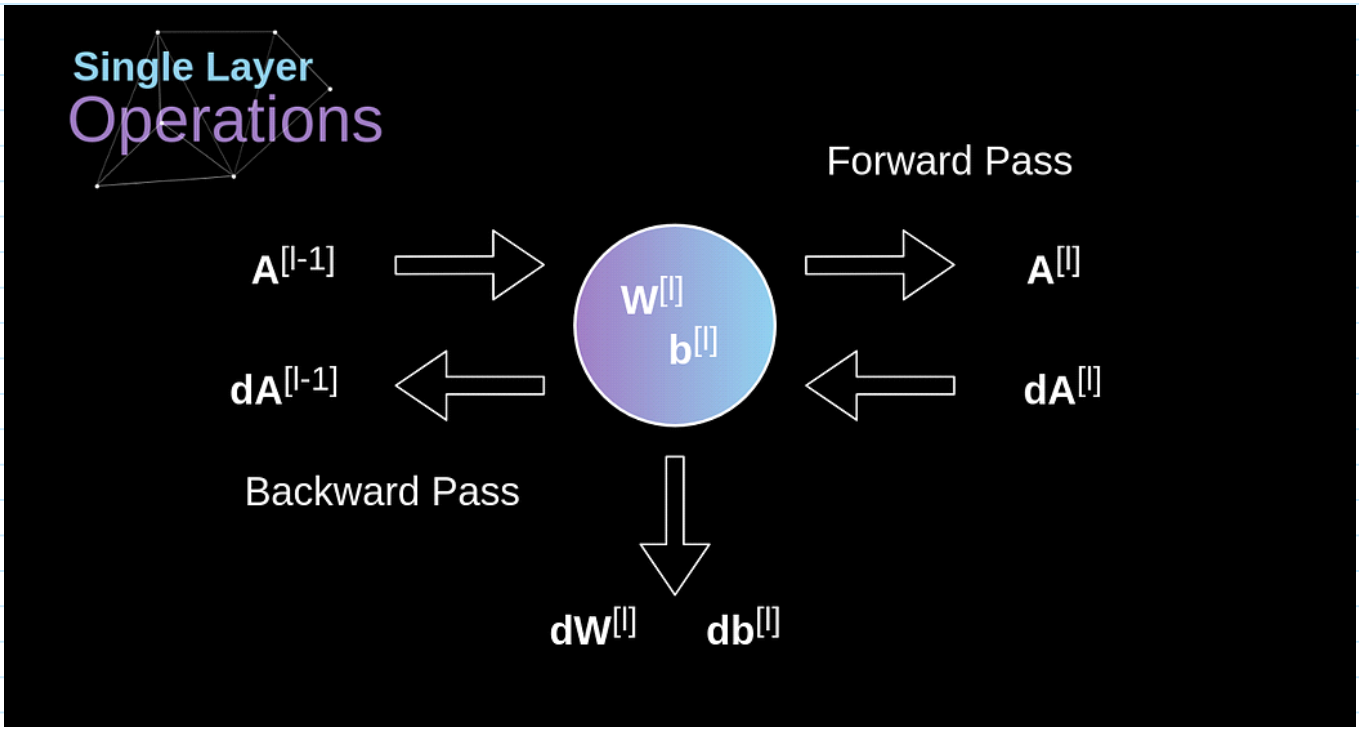
לפי הסרטון, בואו נתמקד בשתי התכונות החשובות מאוד של שכבות ה-Convolution, אשר מקנה להם יתרון בעבודה עם תמונות. קודם כל, ניתן לראות שלא כל הנוירונים בשתי השכבות הרצופות מחוברים זה לזה.
לדוגמה, נוירון 1 משפיעה רק על הערך של A.
שנית, אנו רואים שחלק מהנוירונים חולקים את אותם המשקלים (Weights).
שני המאפיינים מנמיכים את זמן הלמידה של המודל. אגב, ראוי לציין שערך בודד מה-Filter משפיע על כל רכיב ב-Feature Map, זה יהיה מכריע בהקשר של Backpropagation.

- Convolutional Layer Backpropagation

בדיוק כמו ב-Densely Connected Neural Networks, המטרה שלנו היא לחשב נגזרות ובהמשך להשתמש בהן כדי

לעדכן את ערכי הפרמטרים שלנו בתהליך שנקרא Gradient Descent הוא מוסבר לעומק בסיכום הקודם.
בחישובים שלנו נשתמש בכלל שרשרת, שגם אותו הזכרתי בסיכום הקודם. אנו רוצים להעריך את השפעת השינוי בפרמטרים על ה-Feature Map המתקבל, ובהמשך על התוצאה הסופית. לפני שנתחיל להיכנס לפרטים, הבה נסכים על הסימון המתמטי בו נשתמש, אוותר על הסימון המלא של הנגזרת החלקית לטובת המקוצר שהוא ∂.

$$\mathbf{dA}^{[l]} = \frac{\partial L}{\partial \mathbf{A}^{[l]}} \quad \mathbf{dZ}^{[l]} = \frac{\partial L}{\partial \mathbf{Z}^{[l]}} \quad \mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} \quad \mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}}$$



המשימה שלנו היא לחשב $dW^{[l]}$ ו-$db^{[l]}$, שהם נגזרות הקשורות לפרמטרים של השכבה נוכחית, כמו גם את הערך של $dA^{[l-1]}$, שיועבר לשכבה הקודמת. כפי שמוצג בתמונה, אנו מקבלים את $dA^{[l]}$ כקלט. כמובן, הממדים של המטריצות התלת ממדיות (Tensors) dW, db-W ו-b וכן dA ו-A זהים בהתאמה. הצעד הראשון הוא להשיג את ערך הביניים $dZ^{[l]}$ על ידי החלת נגזרת של הפונקציה שלנו על המטריצה התלת ממדית אותה קלטנו.
על פי כלל השרשרת, תוצאת הפעולה הזו תשמש אותנו בהמשך.

$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} * g'(\mathbf{Z}^{[l]})$$

כעת, עלינו להתמודד עם ה-Backpropagation של ה-Convolution, וכדי להשיג מטרה זו נשתמש בפעולת מטריצה הנקראת Full Convolution, אשר מוצגת בסרטון למטה. שימו לב שבמהלך תהליך זה אנו משתמשים ב-Kernel, אותו סובבנו ב-180 מעלות. ניתן לתאר את הפעולה הזו באמצעות הנוסחה הבאה, שבה ה-Filter מסומן ב-W, ו-$dZ[m,n]$ הוא סקלר השייך לנגזרת חלקית שהתקבלה מהשכבה הקודמת.

$$\mathbf{dA}+ = \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} \mathbf{W} \cdot \mathbf{dZ}[m,n]$$

Full
...Convoluti

- Pooling Layers

מלבד שכבות Convolution, רשתות Convolutional Neural Networks משתמשות לעתים קרובות מאוד במה שנקרא Pooling layers. הם משמשים בעיקר כדי להקטין את גודל המטריצה התלת ממדית ולהאיץ את החישובים. שכבות אלה הן פשוטות - עלינו לחלק את התמונה שלנו לאזורים שונים, ולאחר מכן לבצע פעולה כלשהי עבור כל אחד מאותם חלקים. לדוגמה, עבור Max Pool Layer, אנו בוחרים ערך מקסימלי מכל אזור ומניחים אותו במקום המתאים בפלט. כמו במקרה של שכבת ה-Convolution, יש לנו שני Hyperparameters זמינים, Filter Size וגודל הצעד.
לבסוף, אם אתם מבצעים Pooling לתמונה מרובת ערוצים (כגון תמונה בצבע), יש לבצע את האגירה של ה-Pooling עבור כל ערוץ בנפרד.

Max
...Pooling Ex

- Pooling Layers Backpropagation

בחלק זה נדון רק על Backpropagation של Max Pooling, אך הכללים שנלמד, עם התאמות קלות, חלים על כל סוגי ה-Pooling Layers. מכיוון שבשכבות מסוג זה, אין לנו פרמטרים שנצטרך לעדכן, המשימה שלנו היא רק להפיץ gradients בצורה מתאימה. כזכור, מ-Forwardpropagation ל-Max Pooling, אנו בוחרים את הערך המקסימלי מכל אזור ומעבירים אותם לשכבה הבאה. לכן ברור שבמהלך ה-Backpropagation, השיפוע לא אמור להשפיע על אלמנטים של המטריצה שלא נכללו במעבר קדימה. בפועל, זה מושג על ידי יצירת מסכה שזוכרת את מיקומם של הערכים ששימשו בשלב הראשון, שבה נוכל לנצל מאוחר יותר להעברת ה-Gradients.

Max
...Pooling B