סיכום רשתות נוירונים לתמונות

11 באוגוסט 2023

1 רשתות קונבולוציה - CNN:

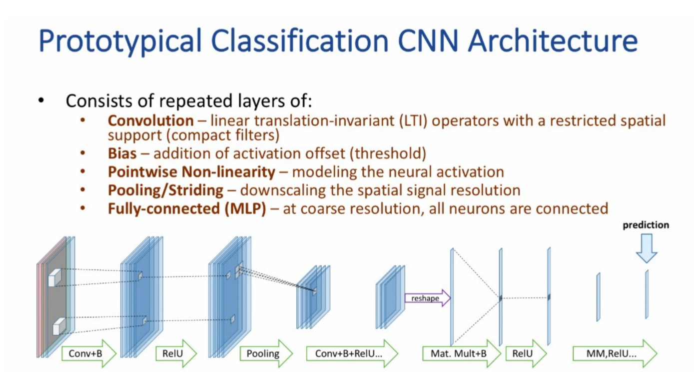
אנחנו משתמשים ברשתות קונבולוציה לתמונות משום שהן עובדות טוב עם שתי ההנחות הבאות:

1. כל נוירון לא מושפע מכל התמונה, אלא רק חלק קטן שנקרא - receptive field.
2. בתמונות אנחנו משתמשים בסטציונריה מרחבית, כל פריט יכול להופיע במקומות שונים ובצורות שונים בתמונה, ואנחנו רוצים שהמודל ינתח את התמונה על כולה, ולא לפי מיקום ספציפי - translation variance.

נשים לב כי בתמונות נרצה שכל הנוירונים באותה שכבה יגיבו באותו האופן לאותם גירויים (כי האובייקט יכול להופיע במקומות שונים התמונה), לכן הם יהיו עם אותם משקלים weight shering.

רשת קונבולוציה: רשת שתכיל שכבות כך שכך שכבה מקבלת את אותו המשקל, כל שכבה תפעיל פולטר שנלמד על הקלט. היא תעשה את זה על חלון בגודל מסויים שאנחנו נקבע מה גודלו.
למעשה: הרשת מזהה פרמטרים קטנים בתמונה ועושה להם לבסוף אינטגרציה, ובודקת האם היאמרכיבה אובייקט מסויים.

1.1 ארכיטקטורה של $Alex\ Net$:



1.1.1 שכבת קונבולוציה:

אוסף של טרנספורמציות לינאריות - לינארי ואינווריאנטי לזמן או למרחב.

$$(Lx_{i+k})_j = (Lx_i)_{j+k}$$

הם לינארים ולכן ניתן לכתוב אותה כמטריצה, כך שהמטריצות יקיימו תכונות סויימות. מטריצה סימטרית עם אלכסונים שווים, ויש בה מלא אפסים.

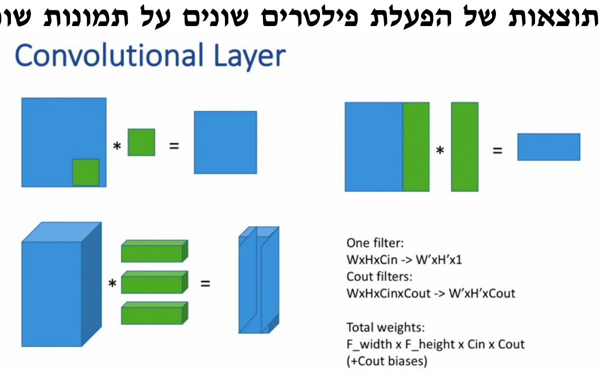
הקשר בין קונבולציה להתאמת דפוס - pattern matching:

$$\sum_x (I(y+x) - f(x))^2 = \|I(y+x)\|^2 + \|f(x)\|^2 - 2\sum_x (I(y+x)f(x))$$
$$\sum_x (I(y+x)f(x)) = \sum_x (I(y-x)f(-x)) = (I * \bar{f})(y)$$

למעשה, הרשת מחפשת דפוסים בדאטה, אך לא בהכרח דפוס שאנחנו חשבנו שהוא חשוב.

- אנחנו מורידים את הרזולוציות המרחביות, ולכן כדי לפצות על זה אנחנו נעלה את מספר הערוצים (פיצ'רים). כך אנחנו יותר אבסטרקטים.
- ה receptive fieldשל הנוירונים בשכבות האחרונות גדול יותר, כי הוא מכיל את כל התמונה כמעט.
- בתמונות טבעיות, יש קורולציות מרחביות במרחקים כטנים, לאן אין טעם להשתמש בפילטר גדול, כי ככל הנראה לא יהיה קשר בין חלקים רחוקים בתמונה.
- הקונבולוציה היא על הצירים x, y. אך על ציר ה z אנחנו משתמשים במ"פ, ולכן אנחנו מקבלים סקלר. כלומר - כל פילטר מופעל על x, y, וציר ה z מצטמצם ל 1. כל פילטר מחזיר לנו ערוץ בודד לשכבה הבאה. מספר הפילטרים בשכבה ה i, יהיו מימד ה z של הקלט בשכבה ה $i+1$.
- נשים לב כי התמונה שנקבל לאחר פילטר תהיה קטנה יותר בגלל קצוות התמונה.

תוצאות של הפעלת פילטרים שונים על תמונות שונות:



1.1.2 שכבה לא לינארית והביאס:

- נרצה להכניס למודל פונקציה לא לינארית כדוגמת: סייגמואיד, או $ReLU$.

$$Sigmoid := \frac{1}{1+e^{-x}}$$
$$ReLU := \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

- בחירה של פונקציה ספציפית לא כלכך תשנה, העיקר שתהיה פונקציה שמממשת אי לינאריות.

- לבסוף נוסיף $bias$ שמזיז לנו את הסף של הגרף. ביאס חיובי יזיז את הגרף שמאלה, וביאס שלילי יזיז את הגרף ימינה.

1.1.3 שכבת $Pooling$:

- שכבה בה אנחנו נוריד את מימד התמונה.

- יש לנו מודל שנקרא $stribe$ שבו אנחנו נדגום כי פיקסל שני אם $stribe = 2$ וכן הלאה.

- ניתן להשתמש ב $avrage\ pooling$ או $Max\ pooling$.
אין טעם ללמוד $Average$ כי הרשת הייתה יכולה ללמוד את זה בעצמה אם זה היה חשוב.
Max יהיה טוב יותר כי הוא לא לינארי.

1.2 השכבה האחרונה ומדידת ביצועי הרשת:

ברשת קלאסיפיקציה יש לנו מספר קלאסים שאנחנו רוצים שהרשת תמפה אליהם את הקלט. כלומר בהינתן קלט, נרצה שהרשת תגיד לנו מה הקלאד שהקלט שייך אליו.

1.2.1 $Softmax$:

כדי לבצע זאת, נוסף לאחר שכבת ה FC, שכבת $softmax$. הרשת פולטת לנו וקטור לאחר שכבת ה FC (משום שכפלנו מטריצה בווקטור), מימד הווקטור יהיה מספר הקלאסים.
אנו נרצה להמיר את הווקטור שהרשת פולטת לווקטור הסתברות, שמנבא את ההסתברות שהפלט שייך לכל אחר מהקלאסים.
נעשה זאת בעזרת פונקציית $softmax$:

$$p_i = \frac{e^{q_i}}{\sum_i e^{q_i}}$$

לאחר מכן נמדוד את ביצועי הרשת בעזרת $Loss$. כמה רחוק הווקטור שיצא משכבת ה $softmax$ לעומת הווקטור המקורי - $lable$.

1.2.2 פונקציות $Loss$:

1. $Cross\ Entropy\ Loss$ **- לרשת קלאסיפיקציה:**

נמדוד את ביצועי הרשת באופן הבא:

כאשר p הוא וקטור הפלט ו y הוא ה $lable$.

$$H(p,y)=H(p)+D_{KL}(p\|y),\quad H(p,q)=-\sum y_i\log\left(p_i\right)\quad D_{KL}(p\mid y)\equiv p_i\log\left(\frac{y_i}{p_i}\right)$$

למעשה, אנחנו רוצים פונקציה גזירה כדי שנוכל להביא אותה למקסימום, או להביא את ההפסד למינימום.

אם אנחנו מתמודדים על שני קלאסים ב"ת, כלומר שקיום של קלאס אחד מעיד בהכרח שהשני לא מתקיים. נוכל למדוד באופן הבא:

$$2\ classes:L(p,y)=-y_i\log\left(p_i\right)-\left(1-y_i\right)\log\left(1-p_i\right)$$
$$more\ classes:L(p,y)=-\sum_i y_i\log\left(p_i\right)$$

כאשר את מיפוי הערך לטווח $[0,1]$ נוכל לעשות בעזרת פונקציה לוגיסטית:

$$\text{logistic function } p=\frac{1}{1+e^{-q}}$$

2. **Regression Loss- לרשת רגרסיה:**

בבעיות רגרסיה אנחנו רוצים שהרשת תחזה לנו מספר בטווח (למשל גיל או מחיר), כדי למדוד את ביצועי הרשת נוכל להשתמש בפונקציית ה $Loss$ הבאה:

$$\sum_n\left(N_\theta\left(x^n\right)-y^n\right)^2$$

1.3 אימון הרשת - למידה:

אנו רוצים ללמוד את משקולות הרשת, כדי שהיא תחזה לנו את הפלט הנכון.

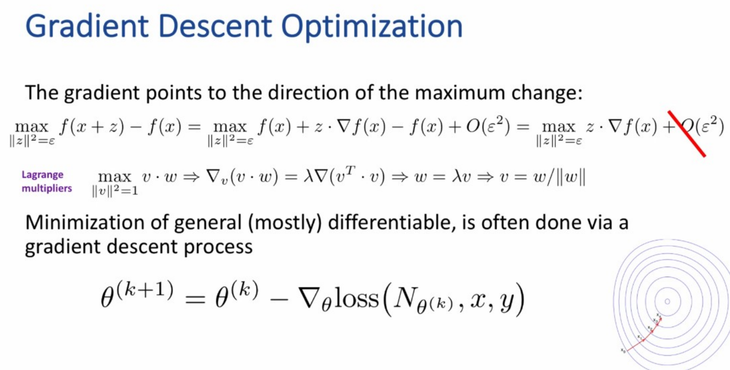
למעשה אנחנו רוצים למזער את ה $Loss$, נעשה זאת ע"י פונקציות גזירות ומציאת המינימום ע"י הנגזרת.

$$\min_\theta loss=\frac{\min_\theta-\sum_n\sum_i y_i^n\log p_i^n}{\sum_n\left(N_\theta\left(x^n\right)-y^n\right)^2}$$

1.4 האלגוריתם Gradient Descent Optimization:

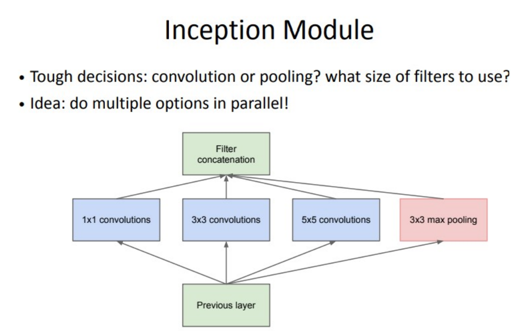
יש לנו פונקציה, ואנחנו מנסים למצוא את המינימום \ מקסימום שלה. נעשה זאת ע"י הליכה בכיוון הגרדיאנט (השינוי הגדול ביותר בנגזרת בכל אחד מהכיוונים - x,y) לחילופין עבור בעיית מינימום נלך בכיוון המנוגד לגרדיאנט. נבחר את גודל הצעד, וכך בכל שלב נלך בכיוון הגרדיאנט בגודל הצעד שבחרנו, עד שניתכנס למינ' \ מקס' מקומי (כשהגרדיאנט יתאפס).

פורמלית:



2 ארכיטקטורת רשתות:

- חוקרים מצאו כי עדיף להחליף שכבת קונבולוציה עם קרנל גדול, במספר שכבות קונבולוציה עם קרנלים קטנים יותר. כך אנו מקטינים את מספר הפרמטרים, וגם יש לנו יותר שכבות אי לינאריות שנותנות אקספרסיביות לרשת.
- Inception Module ברשת $GoogLeNet$: חוקרים של גוגל החליטו לפצל את הרשת לרוחב, כך שבשלב מסויים יופעלו עליה כמה קרנלים בגדלים שונים, בנפרד. לאחר מכן נחבר את כולם, כך שהרשת תבחר את הקרנל הטוב ביותר.



- קונבולוציה עם קרנל 1×1: למעשה היא מעבירה ווקטור לסקלר. אם נפעיל קונבולוציה עם קרנל כזה על מטריצה $n\times n\times m$ אזי כל קאורדינטה (i,j) תעבור להיות סקלאר. כך למעשה ניתן להוריד או להעלות מימד, כתלות במספר הפילטרים בקונבולוציה 1×1.
- נרמול האינפוט - Batch Normalization: בשלב מסויים גילו כי יעיל לנרמל את האינפוט של כל שכבה באופן הבא:

$$\text{BN}(x)=\gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta$$

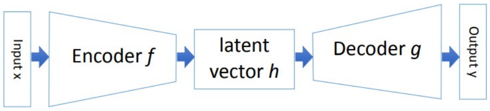
כך שהתוחלת תהיה 0, ותהיה שונות מסויימת.
באוסן כזה אין הבדלים בין התמונות, אלא רק השוני שאחנו רוצים שהרשת תלמד.
γ, β הם פרמטרים נלמדים, ו σ, μ הם ווקטורים.

- הרשת $ResNet$: רשת זאת של מייקרוסופט באה לפתור את בעיה אימון הרשתות העמוקות. עד אז, רשתות עמוקות לא חזו טוב, והרשת הזאת עושה זאת ע"י זה שהשכבות לומדות משהו מסויים, לאחר מכן מוסיפים להן את הקלט המקורי. כלומר, בשלבים מבויימים מכניסים לרשת את האינפוט לפלט של השכבה האחרונה. כך ערכי הגרדיאנט לא יהיו קטנים מידיי ונוכל לחשב אותם ביותר יעילות.

3 $Autoencoders$:

רשת שמבוססת על $encoder - decoder$.
ניתן להשתמש בה להורדת מימד לא לינארית, או להסקת אינפורמציה על תמונות משום שהיא שומרת מטא דאטה שיכול לשמש ללמידה. בנוסף הרשת משמשת למודלים גנרטיביים. רשת זאת היא רשת ללמידה **בלתי מפוקחת**.

מבנה הרשת: הרשת מקבלת אינפוט ומקודדת אותו לוקטור במימד מסוים. בנוסף יש רשת $decoder$ שמקבלת ווקטור מקודד ומוציאה את הפלט המקורי (מטרתה לשחזר את האינפוט).



כך, אנחנו יכולים להוריד מימד, כי רשת לומדת את הפרטים החשובים ביותר.

חישוב ה $Loss$: נמדוד עם נורמה 2 כך:

$$L(x,y) = \sum \|y - g(f(x))\|^2 = \sum \|x - g(f(x))\|^2$$

טענה: אם ה $encoder$ - מטריצה לינארית, וה $decoder$ הוא אותה מטריצה בטרנזספוז - אז מתקיים כי הרשת מחשבת את המטריצה של PCA (אלגוריתם לינארי להורדת מימד).

כדי למצוא את ההתפלגות נשתמש בלוג לייקליהוד: נוכל להחליף את זה בנורמה 2

$$L(x,y) = -\log P(y \mid h(x)) = -\log P(x \mid h(x))$$

שימוש ב $cross - entropy$:

$$L(x,y) = -\sum_i x_i \log(y_i) + (1 - x_i)\log(1 - y_i)$$

Sparse Autoencoders: אנחנו נרצה לגרום לווקטור הפלט של ה $encoder$ להיות דליל. נעשה זאת כך ע"י הוספת גורם נוסף ללוס -

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \boldsymbol{\Omega}(\boldsymbol{h})$$

ע"י הוספת עונש $\boldsymbol{\Omega}(\boldsymbol{h})$ ששווה ל:

$$\Omega(\boldsymbol{h}) = \lambda \sum_i |h_i|$$

או העונש הבא, שיציב יותר לרעש ולשינויים קטנים באינפוט:

$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i \|\nabla_x h_i\|^2$$

Denoising Autoencoders: נקח את האינפוט המקורי x ונוסיף לו רעש באופן מלאכותי ונגדיר את האינפוט המורעש להיות $\tilde{\boldsymbol{x}}$. לאחר מכבן נדרוש מהרשת שלנו להוציא פלט שדומה ל x. כלומר הרשת תלמד לנקות רעשים מתמונה.

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$$

4 מודלים גנרטיבים - Adversarial Generative Models - $GANS$:

המשימה: המודל צריך ללמוד את כל ההתפלגויות שיש לנו, בניגוד לרשתות קלסיפיקציה. מודלים אלו יכולות לייצר דגימות מתוך ההתפלגות - לייצר תמונות חדשות.

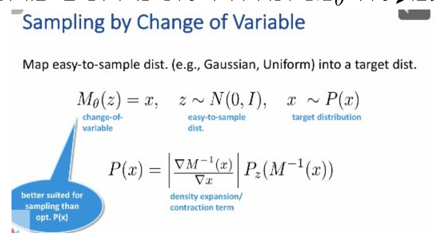
משימה נוספת: מודל זה יכול לדגום תמונה בהינתן משט, או תמונה מטושטשת.

כיצד ניצור תמונה: נדגום ווקטור z מההתפלגות הרב מימדית שלנו, לאחר מכן נכניס את הווקטור z לפונקציה M_θ שתמפה אותו, כך ניצור מ"מ חדש.

$$M_\theta(z) = x$$

כאשר $x \sim P(x)$, אך אנחנו לא יודעים מהי ההתפלגות $P(x)$.

למעשה M_θ תהיה רשת נוירונים מרובת פרמטקים, ואנו נלמד את הפרמטרים שלה.



4.1 כיצד נאמן את M:

יש לנו שתי רשתות $G - generator$ שהיא למעשה תהיה M, ורשת $D - discriminator$, שתי הרשתות ילמדו אחת את השניה.

- G: תקבל z־ים מההתפלגות הפשוטה שאנחנו מכירים, לאחר מכן Gתייצר לנו תמונה.
- D: תקבל שני פרמטרים ־ את הפלט של G, ותמונה מהדאסט סט. היא תצטרך להגיד איזה מהתמונות אמיתית (1) ואיזו מג'ונרטת ע"י G (0).
  המטרה של D היא למצוא את התמונה המזוייפת, והמטרה של G היא להטעות את D.

האימון של שתי הרשתות נעשה באמצעות פונקציית הלוס הבאה:

$$\min_G \max_D V(D, G) = \min_G \max_D E_{p_{\text{data}}}[\log D(x)] + E_{p_z}[\log(1 - D(G(z)))]$$

קודם כל, נרצה ש D תתכנס לאנשהו ־ ניתן לו מלא איטרציות, ולאחר מכן נתחיל להריץ את G עם מעט איטרציות.
מתי נסיים: כשהרשת G תצליח להערים על D, אח"כ נזרוק את D ונשתמש ב Gשתייצר לנו תמונות.

4.2 בעיית Mode Collaps:

- **בעיית סטורציה:** אם D מזהה טוב מידיי אילו תמונות הן המזוייפות, אזי G לא יוכל לגזור את הפונקציה ש D מחזיר לו ־ הוא יגיע לסטורציה (הגרדיאנטים יתאפסו). ולכן לא משנה מה הוא יעשה הוא לא יצליח לרמות את D.
  הפתרון: נעזור ל G ־ במקום לתת ל D להגיע להתכנסות וכך הוא יצבור יתרון על G, נריץ כל אחר מהם לתהליך של גרדיאנט דסנט אחד.

- הפתרון הקודם יכול לגרום לבעיה אחרת: אם ננסה למזער את G קודם, במקום למקסם את D תחילה. כך ־

$$\min_G \max_D V(D, G) \Leftrightarrow \max_D \min_G V(D, G)$$

  אזי G יבחר תמונה אחת ש D חושב שהיא אמיתית, וישלח לו אותה בכל פעם כדי "לנצח".

לכן: נצטרך לאמן בזהירות ולבחור את ההיפר פרמטרים כך שלא ניכנס לבעיית Mode Collaps

בעיות נוספות של GAN**:** הם לא יודעים לייצר תמונות מהדאטה סט, כלומר הם לא יודעים לייצר את כל התמונות מההתפלגות.

4.3 Conditional GAN - דגימת התפלגויות מותנות:

שיטה זו נכונה באופן כללי, אך נראה אותה בהקשר של GAN.
אנחנו דוגמים וקטור מתוך התפלגות $p(x)$, אך למעשה אנחנו יכולים להתנות אותה במאורע y כך $p(x|y)$.
לדוגמה: שהגנרטור שלנו ייצר ספרה מסויימת.

הביצוע:
נכניס ל G, D את ה $class$ כך שהם יידעו לאן לכוון.

- נצטרך להכניס לדסקרימינטור את התיוג של התמונה בכל פעם, כך הוא יראה את הקורלציה בין התמונה לתיוג.
- לעומת זאת הגנרטור יצטרך לייצר את התמונה באופן אקראי, וגם לדעת מה אנחנו רוצים שהוא ייצר.

הארכיטקטורה:
אנחנו צריכים לשנות את ארכיטקטורת הרשת, כי אנחנו רוצים להזין את ה $class$ גם כן לתוך הרשת.
ניתן לעשות זאת ע"י הכנסת כופל של ווקטור $bias$ שנלמד.

המודל:
בהינתן מונה I_i של $class$ שמתוייג C_i:

$$\mathrm{D}\left(\mathrm{I_i}, \mathrm{c_i}\right) -> 1$$
$$\mathrm{D}\left(\mathrm{G}\left(\mathrm{z}, \mathrm{c_i}\right), \mathrm{c_i}\right) -> 0$$

עבור העלאת רזולוציה של תמונות - $SRGAN$:
ניתן לאמן באופן הבא:

$$\begin{array}{ll} \mathrm{D}\left(\mathrm{H_i}\right) -> 1 & \mathrm{D}\left(\mathrm{H_i}\right) -> 1 \\ \mathrm{D}\left(\mathrm{G}\left(\mathrm{z}, \mathrm{L_i}\right)\right) -> 0 & \mathrm{D}\left(\mathrm{G}\left(\mathrm{L_i}\right)\right) -> 0 \end{array}$$

כאשר $L_i = down\left(H_i\right)$ ו - z הוא ווקטור. ניתן לאמן את הרשתות גם ללא הווקטור z.

4.3.1 :Pix2Pix

דרך נוספת לייצר התפלגויות מותנות בעזרת GAN, עם ארכיטקטורה טובה יותר.

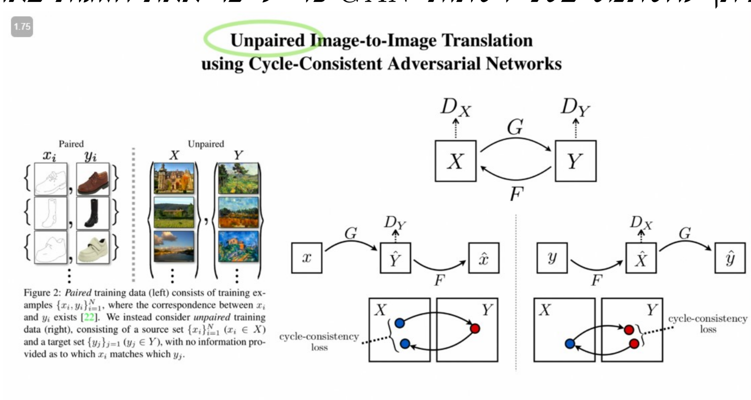
דאטה סט: יש לנו צמדים של תמונות ואנחנו רוצים ללמוד את ההתאמה בין הצמדים.

לדוגמה: איור של תמונה, ותמונה. ואנחנו רוצים שהרשת תקבל איור ותייצר ממנו תמונה.
נלמד כך: כאשר I, J תמונות (איור ותמונה).

$$\mathrm{D}\left(\mathrm{I_i}, \mathrm{J_i}\right) -> 1$$
$$\mathrm{D}\left(\mathrm{I_i}, \mathrm{G}\left(\mathrm{I_i}\right)\right) -> 0$$

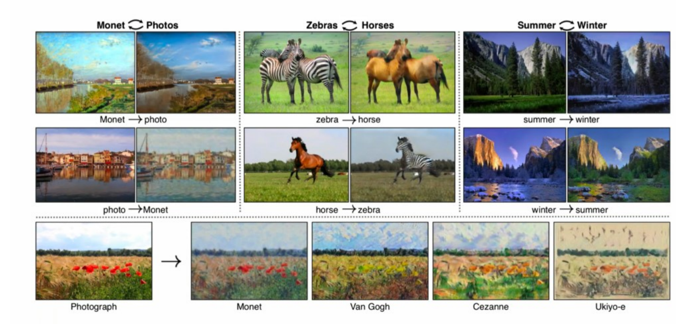
G מקבל את האיור ומייצר ממנו תמונה.

כיצד נייצר זוגות של תמונות:
ניתן להשתמש בשני רשתות GAN כדי לייצר אאת הזוגות באופן הבא:



כך, נייצר מידע משני קלאסים - X, Y. נקח תמונות מ X נכניס ל G והוא יעביר אותנו ל Y. באופן האופן נמפה תמונות מ Y ל X.
נעשה זאת כך: G ימפה תמונה מ X ל Y, לאחר מכן נבקש ממנו למפות את התמונה הזאת חזרה ל X. באופן כזה נקבל את אותה התמונה (תיאורתית). נעזה באותו האופן מ Y דרך X חזרה ל Y, ונרצה ששני המעגלים ימוזערו. כך יהיו לנו זוגות של תמונות.

התוצאות נראות כך:



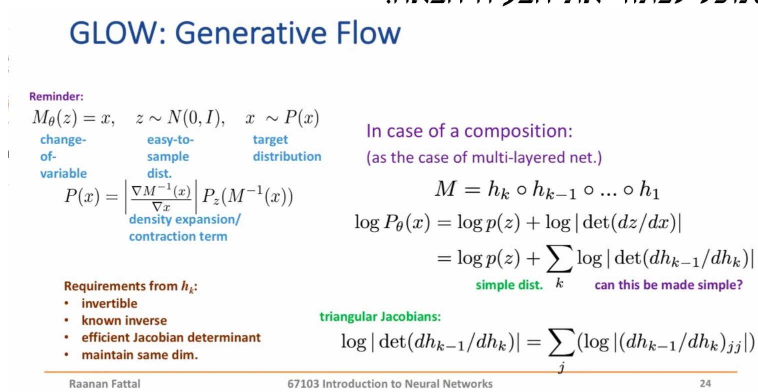
5 :Non-Adversarial Generative Models

שיטות שהתפתחו כאסטרטגיה חלופית לרשתות GAN.

באותו האופן של GAN נשתמש בחילוף משתנה - אנחנו רוצים ללמוד התפלגות, והרשת שלנו צריכה ללמוד את ההתפלגות שנשלטת ע"י משתנה נלמד.

5.1 :GLOW – *Generative Flow*

היוצרים בחרו רשת שהארכיטקטורה שלה תייצר לנו מטריצה משולשית הפיכה שקל לחשב את הדטרמיננטה שלה. כדי שנוכל לפתור את הבעיה הבאה:



5.2 : GLO - Generative Latent Optimization

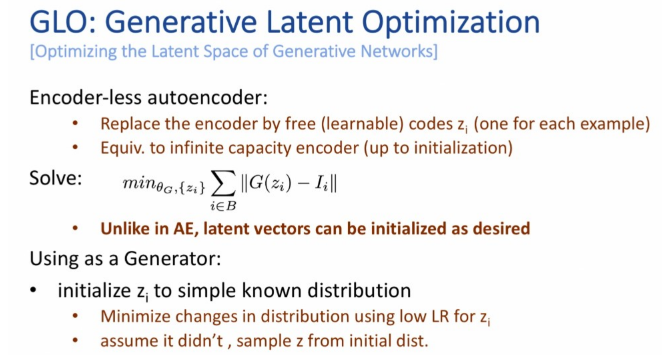
שיטה נוספת, שאינה אדברסריאלית.

הרעיון: נאפטם את G כך:

$$\min_{\theta_G, \{z_i\}} \sum_{i \in B} \|G(z_i) - I_i\|$$

כדי שיוכל לפרוש את ההתפלגות שלנו, אך היא מאמנת את פרמטרי הרשת ואת הפרמטרים של וקטורי המקורות.

נאתחל את ה z־ים להיות דגימות של גאוסיאן סטנדרטי. כך ההתפלגות של z לא תזוז הרבה, ולכן נוכל לדגום מהגאוסיאן הרגיל.

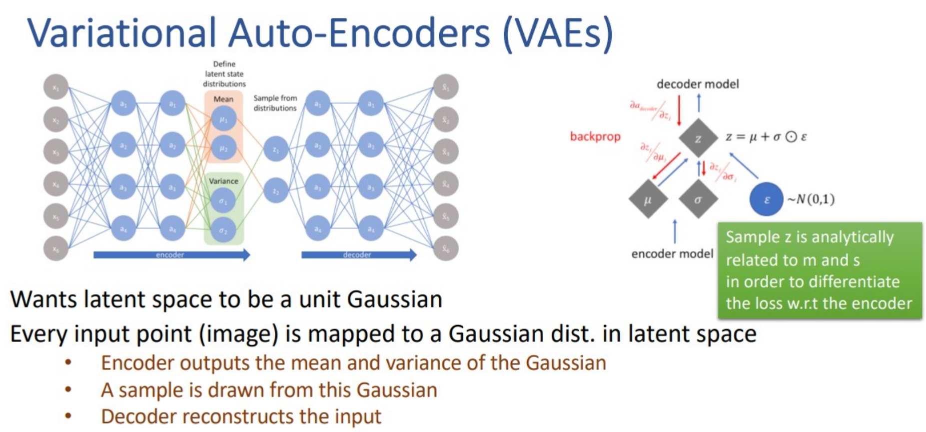


5.3 Variational Auto-Encoders (VAEs):

השיטה הזאת גם לא כך טובה כמו הקודמות.

מבנה הרשת: יש לה מבנה של אוטו־אינקודר. אנחנו מכניסים תמונות ומקבלים קידוד. אל הווקטור שאנחנו מקבלים ב $latent\ space$ אנחנו נתייחס כאל ווקטור התפלגות שנדגם מבתפלגות מסויימת, האנקודת מייצר את התוחלת והשונות. כלומר התמונה הופכת להתפלגות.

לאחר מכן נכניס את התמונה ל $decoder$ שייצר לנו תמונה מהווקטור.



שלב האימון: כל תמונה מתמפה לגאוסיאן ספציפי לתמונה. דוגמים מהגאוסיאן לפי הנוסחה שמופיעה בשקף $z = \mu + \sigma \cdot \varepsilon$, מכניסים ל $decoder$ ומגדירים $Loss$ שישחזר את התמונה.

הדגימה: יש בשיטה שני לוסים, אחד של האוטו־אינקודר, ואחד נוסף. אנו רוצים שכל אחד מהגאוסיאנים יהיו סטנדרטי (תוחלת 0 ושונות 1) משום שכל תמונה מתמפה לגאוסיאן אחר, אך בסופו של דבר כדי לייצר תמונות אנחנו רוצים לדגום מגאוסיאן אחד, ולשלוח ל $decoer$. לכן נרצה לוס נוסף שימדוד לנו את הגאוסיאן הכללי.

פונקציית הלוס: נשתמש בלוס הבא

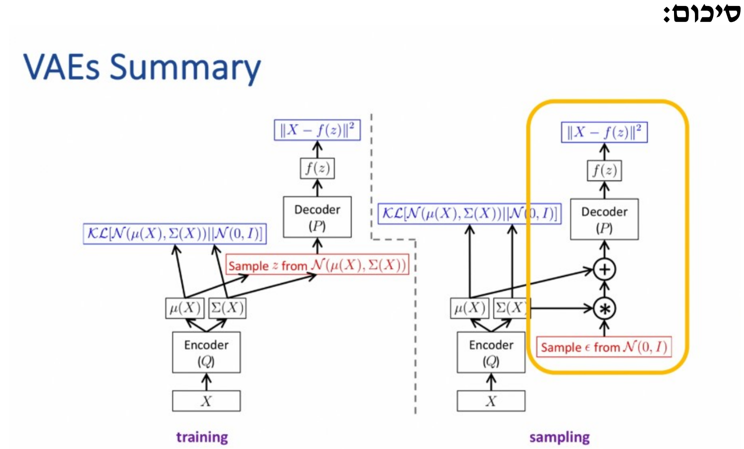
$$D_{KL}\left(G_1, G_2\right) = \int \left(\log\left(G_1(x)\right) - \log\left(G_2(x)\right) G_1(x)dx = \right.$$

$$\int \left(\sum_k \left(x^k - \mu_1^k\right)^2 / 2\sigma_1^k - \sum_k \left(x^k - \mu_2^k\right)^2 / 2\sigma_2^k \right) \left(\sqrt{2\pi^d}\right) e^{-\sum_k \left(x^k - \mu_1^k\right)^2 / 2\sigma_1^k} dx$$

הלוס הסופי:

$$\sum_i \| {}_{\text{reconstruction}} \| \left(I_i\right)\Big) - I_i\| + E_{D_\neg}\left(I_i\right)^2 - E_\sigma\left(I_i\right)^2 - 1 - 2\log\left(E_\sigma\left(I_i\right)\right)$$

סיכום:



5.4 Wasserstein Auto-Encoders:

שיטה זאת בעיה לפתור את הבעיתיות בשיטה הקודמת
בסיטה הקודמת דרשנו כי כל גאוסיאן יהיה סטנדרטי, כדי שהגאוסיאן הכללי יהיה סטדנרטי. אך זה גורם לכך שתמונות שונות ימופו לאותה תמונה משום שה *latent vector* לקח אותם לגאוסיאנים שונים.
הפתרון הוא להוסיף לוס נוסף.

פוקנציית הלוס:

$$\text{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbf{E}_{x\sim p}[f(x)] - \mathbf{E}_{y\sim q}[f(y)] \right), .$$

$$\text{MMD}[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(\frac{1}{m} \sum_{i=1}^{m} f\left(x_i\right) - \frac{1}{n} \sum_{i=1}^{n} f\left(y_i\right) \right)$$

נקח הרבה תמונות, ונחשב את התוחלת של פונקציה ביחס לדגימות הללו.
לאחר שהאנקודם ייצר לנו מהתמונות *latent vector*, נקח אותם. נחשב את ממוצע שלהם על איזושהי פונקציה f, ונשווה לדגימות שנדגמו מהתפלגות נורמלית סטנדרטית. כך נוכל לבחור את הפונקציה f המתאימה ביותר כך שהאנקודר ימצא לנו את ההתפלגות הסטנדרטית של הדגימות.

6 Score Matching / Denoising Diffusion Probabilistic Models:

בסיס השיטה - SDE: בשונה משיטות קודמות, כאן לא נשתמש בשינוי משתנה.

$$\frac{\partial x}{\partial t} = \nabla E(x)/2T + \eta$$

$$\frac{\partial}{\partial t}x \approx \frac{x^{n+1} = x^n}{\Delta t}, \quad x^{n+1} = x^n + \Delta t \frac{\nabla E(x)}{2T} + \sqrt{\Delta t}\eta, \quad \eta \sim N(0,1)$$

הרעיון הוא להסתכל על משוואה דפרנציאלית שיש לה חלק דטרמיניסטי שסוכם את סך כל ההשפעות והשינויים עד הזמן t הנוכחי, ועוד הוספת רעש. באופן כזה אנו יכולים לצאת מנקודה מסויימת, ולהגיע להתפלגות.

את המשתנה t אנחנו מגדירים, והוא מסמן את גודל הצעד.

המשתנה T הוא מעין פרמטר רגולריזציה שמגדיר מ משקל הרעש לעומת משקל פונקציית האנרגיה E.

בסוף נתכנס ל:

$$P(x) \propto e^{-E(x)/T}$$

אנו רוצים ללמוד את פונקציית האנרגיה E.

השיטה: נתחיל מ x^0, וננסה להגיע לנקודת שוויון. כשנגיע לשוויון זאת אומרת שהתייצבנו על התפלגות שחיפשנו.

כיצד נמצא את הגרדיאנט של E: אנו נחפש את הגרדיאנט של $log(P)$ כך

$$\mathcal{L}_{mse} = E_{\mathbf{x}\sim p(\mathbf{x})}\left[\|\mathcal{F}_\theta(\mathbf{x}) - \nabla_\mathbf{x} \log p(\mathbf{x})\|_2^2\right]$$

נשים לב שזה יתרון, כי גרדיאנט של לוג הוא מנורמל, ולרשתות נוירונים יותר קל לעבוד כך.

הבעיה העיקרית בנוסחה הנ"ל: אין לנו את p.

לכן נשתמש בנוסחה הבאה:

$$\mathcal{L}_{\mathbf{matching}} = E_{\mathbf{x}\sim p(\mathbf{x})}\left[tr\left(\nabla_\mathbf{x}\mathcal{F}_\theta(\mathbf{x})\right) + \frac{1}{2}\|\mathcal{F}_\theta(\mathbf{x})\|_2^2\right]$$

כך אנו לא צריכים ללמוד את p.

מימוש: נדגום x־ים מתוך ההתפלגות p, ונרצה למזער את הפונקציה הזאת על הרשת שלנו. איפה שתהיה התפלגות גבוהה של p שם אנו נרצה למזער את ה $divergence$ על מקומות שיותר ריאלי שהם שייכים להתפלגות.

מה זה $divergence$: מקומות שהרבה חיצים (הצבעות של הגרדיאנט) נכנסים אליהם יוגדרו שליליים. מקומות שהרבה יוצאים מהם יוגדרו חיוביים, ומקומות שיש איזון בין מספר החיצים היוצאים והנכנסים יוגדרו כ 0.

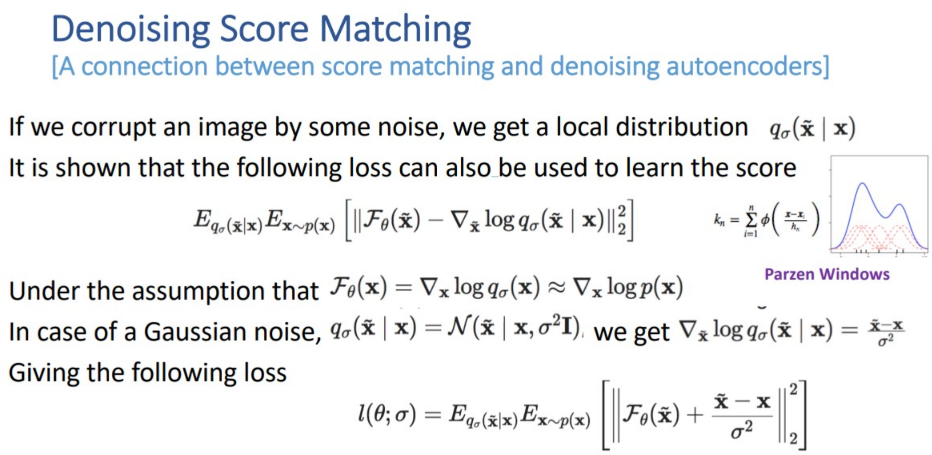
אנו נשאף למזער את ה $divergence$, כלומר שמספר החיצים שיכנסו יהיה גבוה. כך אנו נעודד התכנסות להתפלגות, כי אם יש וקטורים שמצביעים למקום הזה כנראה הוא שייך להתפלגות, לכן נעודד עוד וקטורים להצביע לשם.

ייעול התהליך עם הורדת מימד: נחשב באופן הבא:

$$E_{\mathbf{v}\sim\mathcal{N}(0,1)}E_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbf{v}^T\nabla_{\mathbf{x}}\mathcal{F}_\theta(\mathbf{x})\mathbf{v}+\frac{1}{2}\left\|\mathbf{v}^T\mathcal{F}_\theta(\mathbf{x})\right\|_2^2\right]$$

6.1 Denoising Score Matching:

ננסה ללמוד את ההתפלגות בשיטה אחרת.



6.2 Diffusion Models:

הרעיון: מודל של תהליך סטוכסטי, שבכל שלב יש לנו התפלגות נורמלית שמבטאת את ההתפלגות המותנית משלב i לשלב $i+1$.

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_t;\sqrt{1-\beta_t}\mathbf{x}_{t-1},\beta_t\mathbf{I}\right)$$

$$q\left(\mathbf{x}_{0:T}\right)=q\left(\mathbf{x}_0\right)\prod_{t=1}^{T}q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$$

כאשר השוויון הראשון הוא ההסתברות לעבור מהמצב הנוכחי למצב הבא.

נוכל ללמוד מהסוף להתחלה באופן הבא:

$$p_\theta\left(\mathbf{x}_{0:T}\right)=p\left(\mathbf{x}_T\right)\prod_{t=1}^{T}p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\quad p\left(\mathbf{x}_T\right)=\pi\left(\mathbf{x}_T\right)$$

וההסתברות למעבר בין מצבים היא:

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \mu_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$$

נוטציות: q היא ההתפלגות האמיתית, שיש בעולם. ואנו רוים לקרב אותה באמצעות רשת נוירונים. נסמן ב $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$ את הקירוב של חזרה אחורה.

אימון הרשת: כאשר אנו הולכים קדימה התהליך הוא יחסית פשוט, כי אנחנו מתקדמים לכיוון גאוסיאן ומוסיפים רעש בכל שלב, נקודה הופכת לגאוסיאן, וזה דבר שאפשר לצפות. אך בכיוון ההפוך, אנחנו לא יודעים מה המקור ממנו באנו, כי לכל נקודת טשטוש (נקודה שהפכה לגאוסיאן) יכולות להיות כמה מקורות.
אנו רוצים שבכל שלב הרשת תפלוט לנו את התוחלת וסטיית התקן.

נקרב את הפתרון על ידי התניה בנוסף על X_0:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \tilde{\boldsymbol{\mu}}\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$$

כיצד נחשב את הצעד אחורה: אאנו יודעים את $q\left(x_{t-1}|x_0\right)$ ואת $q\left(x_t|x_0\right)$, וגם את $q\left(x_t|x_{t-1}\ x_0\right)$. בעזרת חוק בייס אנו יכולים להסיק את $q\left(x_{t-1}|x_t\ x_0\right)$.

פונקציית הלוס שנרצה למזער:

$$L_t=D_{\mathrm{KL}}\left(q\left(\mathbf{x}_t \mid \mathbf{x}_{t\psi 1}, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}\right)\right) \text{ for } 1 \leq t \leq T-1$$

7 Neural Style Transfer:

7.1 טקסטורה:

מה הרשת עושה: נשתמש ברשתות קונבולוציה כדי לייצר תמונות גדולות מדוגמה קטנה - פיתוח טקסטורה.

הרעיון: נקח תמונה מורעשת, וננקה אותה עד שהיא תזכיר את התמונה המקורית. נשווה בין התמונות בעזרת רשת VGG שתתן לנו את הפיצרים שנמצאים בתמונה.

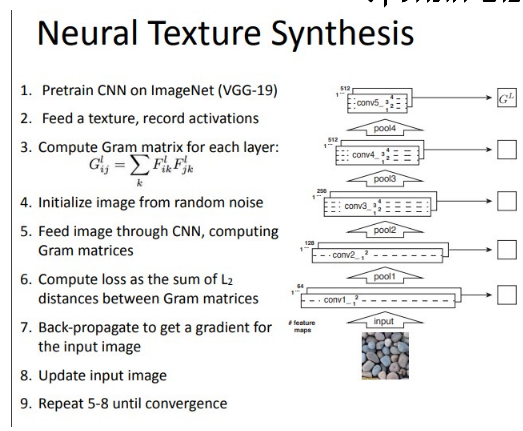
מטריצת גראהם: עבור כל שני ערוצים בשכבה l מסויימת, נבדוק את הקורלציה ביניהם ונשמור במטריצה G

$$\underset{1<i,j<C^l}{G^l_{i,j}}=\frac{1}{C_l H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} F_{h,w,i} \cdot F_{h,w,j}$$

את הלוס נחשב כך: בעזרת חישוב ההפרשים בין מטריצות גראהם

$$L_{style}(x, G(z)) = \sum_{i=1}^{5} w_i E_i \quad E_l = \left\| G^l - A^l \right\|_F^2 = \sum_{i,j} \left(G^l_{i,j} - A^l_{i,j} \right)^2$$

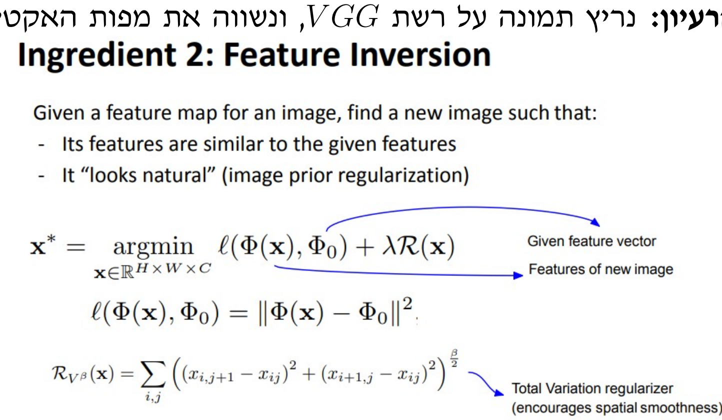
סיכום התהליך:



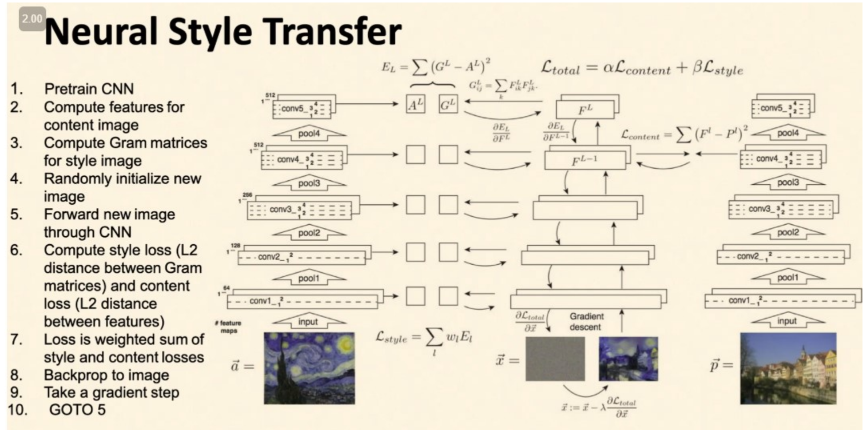
7.2 היפר ראליסטיקה:

אנו רוצים לקחת תמונה מסויימת ולצייר אותה בסגנון אחר. לדוגמה - להעביר תמונה שצילמנו לציור סטייל ואן-גוך.

הרעיון: נריץ תמונה על רשת VGG, ונשווה את מפות האקטיבציה



מימוש: נכניס תמונה שלנו ותמונה שנרה את הסטייל שלה, לאחר מכן יהיו לנו אוסף של מטריצות גראהם. נעשה תהליך אופטימיזציה כמו בפעם קודמת.


2.00
Neural Style Transfer
1. Pretrain CNN
2. Compute features for content image
3. Compute Gram matrices for style image
4. Randomly initialize new image
5. Forward new image through CNN
6. Compute style loss (L2 distance between Gram matrices) and content loss (L2 distance between features)
7. Loss is weighted sum of style and content losses
8. Backprop to image
9. Take a gradient step
10. GOTO 5
$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
conv5_1
pool4
conv4_1
pool3
conv3_1
pool2
conv2_1
pool1
conv1_1
input
feature maps
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$