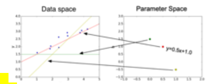
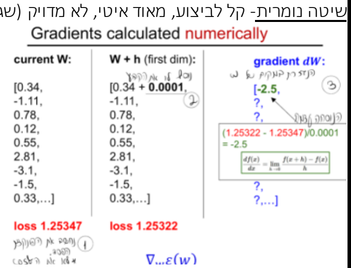
Linear regression- בעיית רגרסיה לערכים רציפים. נרצה למצוא
מודל = הקו הלינארי שייתן לנו מרחק מינימלי מכל הנקודות. y=wx+b



LSE(lose)= $Lse(y,t) = \frac{1}{2}(y-t)2$
MSE(cost)=$\varepsilon(w,b) = \frac{1}{2N}\sum(y^i - t^i)^2 = \frac{1}{2N}\sum(wx^i + b - t^i)^2$
בצורת וקטור: $y = Xw + 1b \quad \varepsilon = \frac{1}{2N}\| y - t \|^2$

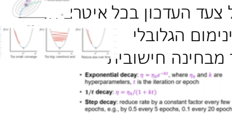
כדי למזער cost נמצא אופטימיזציה לפרמטרי המודל:
1.direct solution- $\frac{\partial f(w1,w2)}{\partial w} = \lim_{h\to 0}\frac{f(w1+h,w2)-f(w1,w2)}{h}$
שיטה נומרית- קל לביצוע, מאוד איטי, לא מדויק (שגיאת עיגול)



שיטה אנליטית- גזירת סימבולים. מהיר ומדויק, מיועד לשגיאות.
$\frac{\partial\varepsilon(w,b)}{\partial b} = \frac{1}{N}\sum(x^{iT}w + b - t^i) \qquad \frac{\partial\varepsilon(w,b)}{\partial w} = \frac{1}{N}\sum(x^{iT}w + b - t^i)x^i$
הבעיה- גרסה לינארית בלבד. אם T מאוד גדול (פרמטרים רבים) ובנוסף גם
אי אפשר לעשות תמיד נגזרות עדיף לרדת בשיפוע.
2. פתרון GD iterative-נקודת מינימום גלובלי בטופוגרפיה. קל ליישום ותמי
נותן פתרון גם אם נתקל בלוקלי. המטרה למזער את פונקציית ההפסד
המודדת את ההבדל בין התפוקה החזויה לבין הפועל. ברוב המקרים המודל
הוא מורכב ובעל ממדים, עם מינימום מקומי ואוכף. מציאת מינימום גלובלי
אינה אפשרית לכן משתמשים בGD להפחית הפסד.
$$w \leftarrow w - \eta\Delta\varepsilon(w) \quad \Delta\varepsilon(w) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial L(w,y^t,t^i)}{\partial w}$$

עדכון הdata שיטות: עוצרים באופטימום נקודות ה0 או לא מתעדכן.
* batch- $w^{t+1} = w^t - \frac{\eta}{N}\sum_{i=1}^{N}\frac{\partial L(w)}{\partial w}$ החזרה- $w \leftarrow w - \frac{\eta}{N}\sum_{i=1}^{N}(w^Tx^i - t^i)x^i$.
מעדכנת את המשקולות לאחר עיבוד כל נקודות הנתונים בכל מחזור אימון
(השיפוע מחושב על פני כל נקודות הנתונים בכל איטרציה). סיבוכיות o(n)
* SGD- $w^{t+1} = w^t - \eta\frac{\partial L(w)}{\partial w}$ החזרה- $w \leftarrow w - \eta(w^Tx^i - t^i)x^i$ מהיר ורועש.
עבור כל נקודה במערך הוא מעדכן את המשקולות ומחיל תיקון לכיוון השיפוע
יתרונות על פני *Batch* *זמן אימון מהיר יותר כי מעדכנים בצורה קטנה את
המשקולת. מאפשר לעבוד עם סט גדול של ערכים. *יעילות טובה יותר עבור
מערכי נתונים עם נתונים לא אחידים. בbatch ניתקל ברעש שישפיע על
הכל ואילו עם SGD נעדכן עבור כל נקודה באופן עצמאי ונמנע מבעיות.
חסרונות על פני *Batch* *התכנסות איטית ושונות מוגבר יוביל להתנהגות לא
יציבה. *דרוש כוונון היפר פרמטר זהיר כדי להשיג ביצועים טובים. סיבוכיות o(1)
* מיני batch- $w^{t+1} = w^t - \frac{\eta}{B}\sum_{i=1}^{B}\frac{\partial L(w)}{\partial w}$ ההחזרה $w \leftarrow w - \frac{\eta}{B}\sum_{i=1}^{B}(w^Tx^i - t^i)x^i$
סיבוכיות o(N)

קצב הלמידה: משמש להחלטת גודל צעד העדכון בכל איטר
גדול מדי- עלול להיכשל ולעלות על המינימום הגלובלי
קטן מדי- ייקח הרבה זמן להתכנס, יקר מבחינה חישובית



1. עדכון ידני- ניסוי וטעיה ע"י ניחוש.
2. עדכון דינמי- אסטרטגיות בתמונה:
3. מומנטום- תנע, תנופה. כאשר אנחנו נמצאים בנקיק צר וארוך תוך זיגזג ע
פני גיא או שנתקענו במינימום לוקלי או אזור שטוח עקב vanishing או
נקודת אוכף (הנגזרת 0) או בהתכנסות איטית כאשר צעד העדכון קטן מדי.

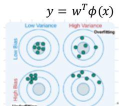
עדכון המומנטום $v \leftarrow av - \eta\frac{\partial L(w)}{\partial w} \quad w \leftarrow w + v$
α צריך להיות 0.9 או 0.99 פחות מ1 כי אם יהיה גדול מ1 המומנטום יגדל שו
ושוב עד לexploding ולא נשיג מינימום. קרוב ל0 זה יצטמצם לסטנדרט.
Nestrov- טכניקה נוספת: $w \leftarrow w + v \quad v \leftarrow av - \eta\frac{\partial L(w+v)}{\partial w}$ במקום להשתמ
במיקום הנוכחי כדי לחשב את השיפוע, מחשבת תחילה מיקום ביניים על ידי צעד בכיוון
המומנטום הקודם ורק לאחר מכן מחיל את המומנטום על השיפוע. תיקון וקטור המומנטום
לפני חישוב השיפוע. התכנסות מהירה יותר ולנתיב אופטימיזציה יציב.
4.עדכון אדפטיבי- גודל צעד שמתאים לכל וקטור בהתאמה. שיפועים גדולים
יקבלו מדרגות קטנות ושיפועים קטנים יקבלו גדולים. עוזר להשיג התכנסות
מהירה בנקיקים צרים כשמזגזג מעל גיא. עוזר באזורים עם שיפועים נעלמים א
מתפוצצים. RMSProp - $w_k \leftarrow w_k - \frac{\eta}{\sqrt{s_k+\epsilon}} * \frac{\partial L}{\partial w_k} \quad s_k \leftarrow \beta * s_k + (1-\beta)\left|\frac{\partial L}{\partial w_k}\right|^2$
אפסילון כדי שלא נחלק באפס. בכל שs גדול -צעד קטן יותר.

Adam – קומבינציה של RMSProp וmomentum- s_k נשאר זהה.
$$v_k \leftarrow \alpha v_k + (1-\alpha) * \frac{\partial L}{\partial w_k} \qquad w_k \leftarrow w_k - \frac{\eta}{\sqrt{s_k+\epsilon}} * v_k$$
האדפטיבי יותר טוב מומנטום כאשר ההתכנסות רגישה לגודל צעד. בנקודות אוכף
ובמישורים עם שיפועים קטנים מאוד. בנקיקים צרים בעת ריצוד על גיא.
סיבות להתכנסות איטית:
1.קצב למידה נמוך. חפש קצב למידה טוב יותר *rate decay* או אדפטיבי.
2.הרשת הגיעה לגיא צר ותלול בחלל הפרמטרים. מומנטום או אדפטיבי.
3. נקודת אוכף או אזור שטוח במרחב הפרמטרים. מומנטום.

הערכה של w ראשונית:
בעיה 1: אם נאתחל במספר קבוע 0 או 1 לכל היחידות הנסתרות יהיו אותו bias
ואותו משקל ועל כן אותו שיפוע (נגזרת זהה) ואותם עדכונים למשקלים, נהיה כלואים
במרחב פרמטרים. יחלצו אותם פיצ'רים ולא ילמדו להיות שונות.
בעיה 2: אם נאתחל שלילים כל האקטיבציות יהיו 0. vanishing gradient קטן
ממטריצת הזהות- מהר מתכנס ל0. אין עדכון של הרשת, אתחול לא טוב.
בעיה 3: 0.05 נקבל בין -1 ל1 ואין אימון. נאתחל עם מספר רנדומלי סביב ה0 כדי
שחלק יהיו חיובים ושלילים. מטריצת התוצאה מתפוצצת exploding gradient.
הנגזרות יתפוצצו כי משתמשים בtanh. הנגזרת 0 ואין אימון.

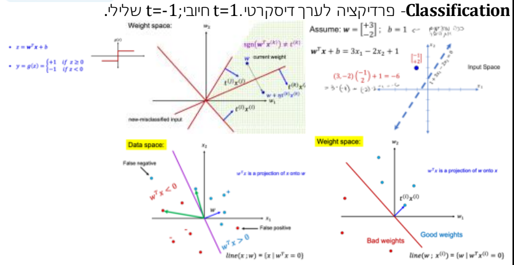
אתחולים נכונים:
Xavier $\sigma = \frac{1}{\sqrt{D_{in}}}$ ניקח שורש של המימד לפני (השכבה לפני) המטרה=
שונות של הקלט שווה לפלט $var(y_i) = var(x_i)$. בכל שD_{in} גדול אז שינוי קטן מאוד
ישפיע על ה variance של הoutput . המעבר הוא $var(y_i) = D_{in} * var(w_ix_i)$ ואז
בגלל ההנחה שwx לא תלוים וזירו מין (ביאס הוא אפס) $var(y_i) = D_{in} * var(w_i) * var(x_i)$
ולהציב את השונות הידועה של w $var(w_i) = \frac{1}{D_{in}}$ והכל מצטמצם.
Glorot&Bengio $\sigma = \frac{2}{\sqrt{D_{in}+D_{out}}}$ אולי השונות שונים אז עושים ממוצע ביניהם.
אתחול ביאס בשניהם: 0 כי לא פוגע.
Relu $\sigma = \sqrt{\frac{2}{D_{in}}}$ מקרים שהערכים הם 0 בחוויאר ולכן נשתמש בזה.

None Linear regression- פרדיקציה מכמה ממדים $y=x^Tw + b$
רגרסיה פולינומיאלית– מגדירים פיצ'רים במקום x המודל $\phi(x) = \begin{pmatrix}1\\ x1\\ x2\end{pmatrix}$
$y = w^T\phi(x)$ where $w^T = (w0, w1, w2)$;$\varepsilon(w) = \frac{1}{2N}\sum(y^i - t^i)^2 = \frac{1}{2N}\sum(w^T\phi(x) - t^i)^2$



נבחר מודל שייתן לנו התאמה טובה (לא יתרה ולא תת התאמה)
bias סטיה מהמרכז.**Variance** כמה הדגימות מפוזרות.
השגיאה: $f = bias^2 + var$ בכל שמספר פרמטרים
עולה= שגיאת Bias תרד. גודל המודל יעלה את var

Classification- פרדיקציה לערך דיסקרטי.t=1 חיובי;t=-1 שלילי.

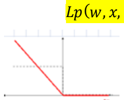


הגבלות מסווג לינארי-
patterns – ניקח ממוצע של פטרן A: 0.25 לכל כניסה(הכניסות השחורות).
הווקטור הממוצע לכל ההזות הוא...0.25,0.25 נניח שהוא מצליח לסווג ומה
שמקבל ערך גדול מ0 זה A. א. $(\sum_i w^Tx^{(i)} > 0,\ so\ then\ \frac{1}{16}w^T\sum_i x^{(i)} > 0)$
אפשר לעשות אותו דבר גם על קטן מ0 כי התבנית הממוצעת של B זהה ולכן
סתירה. הפתרון פיצ'רים לא לינארים בהתחלה ואחר-כך רשתות עמוקות.

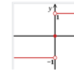
XOR- עם הטבלה ליצור משוואות ששוות לt
עם האיקסים-סתירה.

סוגי losses :
loss 01 - $L_{01}(z,t) = \mathbb{I}(tz < 0)$ אם התנאי יתקיים 1 אחרת 0.
כאשר z=wx+b וגם t=true label

$$L_{01}(y,t) = \begin{cases}0\ if\ y = t\\ 1\ if\ y \neq t\end{cases}$$
העלות $\varepsilon(w) = \frac{1}{n}\sum\mathbb{I}(t^iz^i < 0)$
לפי הגרף הנגזרת שווה ל0- לא יתעדכן w -נגזרת מתאפסת.
perceptron- $Lp(w,x,t) = \max(0, -tz)\ ;\ where\ z = w^T\ x$
t=c היא האמיתי, z הוא הפרדיקציה
הנגזרת $\Delta Lp(w) = -\mathbb{I}[tz < 0]tx$



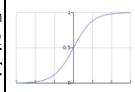
צעד העדכון $w \leftarrow w + \eta\mathbb{I}[t^iz^i < 0]t^ix^i$
SGD כי זה רק נקודה אחת.



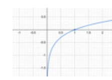
sign function- 0 ל0. -1 לשלילי. 1 לחיובי

Multi class- לכל מחלקה $t \in \{1,..k\}$ יש מסווג z= f1,..fk
כתיבה כללית: $f_j(x) = w_j^Tx + b_j \ ;\ y = \text{argmax}\, f_k$
$z_1 = (w_{11}\ w_{12}\ w_{13}\ w_{14})\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix} + (b_1)$
$z_1 = \sum_i w_{1i}x_i + b_1 = w_1^Tx + b_1$
t^i *המחלקה האמיתית לעומת j שזה הפרדיקציה החיזוי שלי*
פרספטרון- $L_p(w,x^i,t^i) = \sum_{j\neq t^i}\max[0, -(f_{t^i}(x^i) - f_j(x^i))$
עדכון SGD - $w_{t^i} \leftarrow w_{t^i} + \eta x^i \quad w_j \leftarrow w_j - \eta' x^i$
בפועל: להוריד ציונים שגבוהים מהמחלקה ולהעלות את המחלקה. יש נטרלים.

logistic regression $z = w^Tx \quad y = \sigma(z) = \frac{1}{1+e^{-z}}$
שיטה סטטיסטית לרגרסיה בינארית, תוחם את המשוואה לטווח σ .תכונות:



*מה הסיכוי שt=1 $P_w(t = 1|x) = \sigma(w^Tx)$
0<y<1 $P_w(t = -1|x) = 1 - \sigma(w^Tx)$
*התבוננות- $P_w(t = -1|x) = 1 - \sigma(z) = \sigma(-z)$
*סיכום $P_w(t|x) = \sigma(tz)\ \ where\ z = w^Tx$



Loss שימושי הוא **ML** $L_{ML(y,t)=-log\,P_w(t|x)=-log\sigma(tz)}$
הנגזרת $\nabla L(w) = -\sigma(-tz)tx$
צעד העדכון עם SGD $w \leftarrow w + \eta \times \sigma(-t^iz^i)t^ix^i$
מחיר $\varepsilon(w) = \frac{1}{n}\sum_{i=1}^{n} -log\sigma(t^iz^i) \quad \nabla\varepsilon(w) = \frac{1}{n}\sum_{i=1}^{n} -\sigma(-t^iz^i)t^ix^i$

cross entropy - מדד למדידת ההתאמה בין ההסתברויות שניתנות על ידי המודל
להסתברות האמיתית של הנתונים. ערך ההתפלגות גדול- שונים. ערך קטן- קרובים.
הפונקציה מחזירה ערך לא שלילי ולכן ערכים גבוהים מצביעים על דרגת טעות גבוהה
$$H(p,\hat{p}) = -\sum p(c)\log\hat{p}(c) = -\sum p(c|x)log\,\hat{p}(c|x) = -log\ \ \hat{p}(c = t|x)$$
עבור n מחלקות כאשר t_i זו התווית האמיתית וp_i זה הSoftMax $L_{CE=-\sum_i t_i\log(p_i)}$
בבינארי זה $L_{CE=-\sum t_i\log(p_i)} = -[t_1\log(p_1) + t_2\log(p_2)] = -[t\log(p) + (1-t)\log(1-p)]$
$y \in \{1,0\}; \hat{y} = p(c = 1|x) = \sigma(z)$-Binary cross entropy
אנטרופיה היא יחידת מידה והיא כמה חוסר סדר יש. כדי למצוא את פרמטרים טובים,
הערך המינימלי של אנטרופיה צריך להיות נמוך. ההפסד מייצג מרחק בין התפלגות
התווית הרצויה להתפלגות המתקבלת מהמודל. ערך המינימלי הוא 0 -מודל תואם בצורה
מושלמת את ההתפלגות יעד הרצויה.

הוכחה שML שווה לBCE
Let $t^{(i)}$ denotes the **true** class $t^{(i)} \in \{1,-1\}$ for inputs $x^{(i)}$.
Then $P_w(t^{(i)}|\ x^{(i)}) = \mathbb{I}(t^{(i)} = 1)\log\sigma(z^{(i)}) + \mathbb{I}(t = -1)\log\sigma(-z^{(i)}) = \ \sigma(t^{(i)}z^{(i)})$
The maximum-likelihood (ML) maximizes w s.t.:
$w^* = argmax_w \prod_i P_w(t^{(i)}|\ x^{(i)}) = argmin_w - \sum_i \log P_w(t^{(i)}|\ x^{(i)}) = -\sum_i \log\sigma(t^{(i)}z^{(i)})$
Cross entropy minimizes $ce = -\sum_i\sum_c P(c|\ x^{(i)})\ \log P_w(c\ |\ x^{(i)})$ where P is the true distribution and P_w is the approximated distribution.
In our case we have a binary class $t \in \{+1,-1\}$:
$w^* = argmin_w \sum_i -\mathbb{I}(t^{(i)} = 1)\log\sigma(z^{(i)}) - \mathbb{I}(t = -1)\log\sigma(-z^{(i)}) = -\sum_i \log\sigma(t^{(i)}z^{(i)})$
Q.E.D

KL-מדד להפרש בין שתי התפלגויות הסתברות. אין קשר למודל.
$$KL(P||P_w) = \sum -P(c|x)log\frac{P_w(c|x)}{P(c|x)} = \sum -P(c|x)logP_w(c|x) - \sum -P(c|x)logP(c|x) = H(P,P_w) - H(P)$$
אנטרופיה $H(P)$ היא קבועה ולא משפיעה עלw.

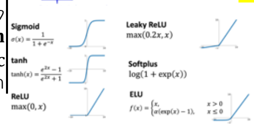
חוסר יציבות נומרית
אם t=1 וz<0 אז y→ 0 ואז log0→ ∞
numerically stable
$\mathcal{L}_{BCE}(z,t) = -\log\frac{1}{1+e^{-tz}} = -\log 1 + \log(1 + e^{-tz})$

SoftMax- ציונים לאחוזים.
$\text{softmax}(f_1,\ldots,f_K) = (\hat{y}_1,\cdots,\hat{y}_k) = \left(\frac{\exp(f_1)}{\sum_j\exp(f_j)},\cdots,\frac{\exp(f_K)}{\sum_j\exp(f_j)}\right)$
מובטח לי שכל ערך הוא בין 0 ל1 והמכנה 1.
$$Lsoftmax = -log\ P_w(t^i|x^i) = -\log(y_{t^i}) = -log\frac{e_{t^i}}{\sum e_j}$$
נגזרת: $x^i = (\hat{y_{t^i}} - 1)x^i$ $-x^i + \frac{\exp(w_{t^i}*x^i)}{\sum\exp(w_{t^i}*x^i)}$ או $\hat{y_j}x^i$
עדכון SGD $w_{t^i} \leftarrow w_{t^i} - \eta(\hat{y}_{t^i} - 1)x^i \ \ for\ \ j = t^i \quad w_j \leftarrow w_j - \eta\hat{y}_jx^i \ \ for\ \ j \neq t^i$
בפועל: הורדה הרבה למי שגבוה ממנו וקצת לניטרליים. העלאה של המחלקה.

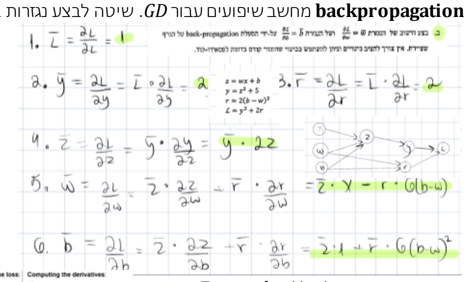
בעיות מסווגים לינארים: אי אפשר להעביר ישר. פתרון: אפשר לייצג אחרת את
הערכים כדי שנוכל להעביר קו ולשמור על הצורה הקודמת שנחזור. **Mlp** *10*
זה המחלקות. בעזרת אקטיבציה. $f(x) = W_2\max(0, W_1x + b_1) + b_2$

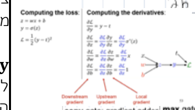
w2 הפך להיות המסווג. $L + \lambda|W|^2$ =MLP loss לכל w ניתן עונש.

פונקציות אקטיבציה לא לינאריות

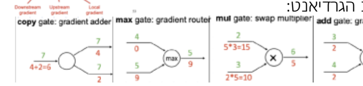


sigmoid *[0,1]*. נשתמש סיכויים.
חישוב יקר נוירונים רווים הורגים את
הגרדיאנט (הנגזרת שלילית). פלטי סיגמויד
לא יכולים להיות 0 (לא זירו סנטר)
נגזרת: $\nabla\sigma(x) = \sigma(x)(1 - \sigma(x))$
tanh טווח *[-1,1]*. הוא כן זירו סנטר. כבד חישובית. הנגזרת שלילית נוירונים
רווים הורגים את הגרדיאנט. נגזרת: $\nabla tanh(x) = 1 - tanh^{2x}$
Relu קל לחשב. אינו רווי באזורי *z>0* הנגזרת לא מתאפסת. סיבוכיות יעילה.
בעיה שלא זירו סנטר-> לפתור עם מיני בטץ. נגזרת: $\nabla relu\,(x) = \mathbb{I}(z > 0)$
Dead RELU 0 עבור כל הקלטים כשהx שלילי ואז הנגזרת 0 וצעד העדכון
יתבצע ללא עדכון. הנוירון לא יהיה פעיל.
Leaky Relu לא רווי כי לא מתאפס. סיבוכיות יעילה, לא ימות. אין חסרונות.
ELU לעולם לא מת. קרוב ל0 ב*output*. סיבוכיות *exp*.
backpropagation מחשב שיפועים עבור *GD*. שיטה לבצע נגזרות ב*MLP*

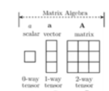




המעבר הראשון מקלט ללוס *Forward*
כל חישוב הנגזרות *Backward pass*
החישוב שעליו מדובר ברגע הנוכח *Local*
תבניות התנהגות הגרדיאנט:



טנזור- ייצוג כללי של ערכים ממימד כלשהו.

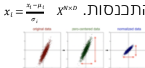


אקטיבציה-מפעילים על כל איבר בנפרד לכן אותם ממדים.
נגזרת של סקלר לפי סקלר=סקלר
נגזרת של סקלר לפי וקטור=הגרדיאנט =וקטור
נגזרת של וקטור לפי וקטור=מטריצה שגודלה-*M*N*
בהינתן סקלר אלפא, וקטור *a*, ומטריצה *A*

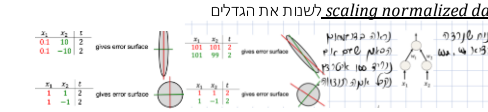
$x \in \mathbb{R}, y \in \mathbb{R}$ — Regular derivative: $\frac{\partial y}{\partial x} \in \mathbb{R}$
$x \in \mathbb{R}^N, y \in \mathbb{R}$ — Derivative is Gradient: $\frac{\partial y}{\partial x} \in \mathbb{R}^N$, $\left(\frac{\partial y}{\partial x}\right)_i = \frac{\partial y}{\partial x_i}$
$x \in \mathbb{R}^N, y \in \mathbb{R}^M$ — Derivative is Jacobian: $\frac{\partial y}{\partial x} \in \mathbb{R}^{N\times M}$, $\left(\frac{\partial y}{\partial x}\right)_{i,j} = \frac{\partial y_j}{\partial x_i}$

(C9)	$\frac{d\mathbf{Ax}}{dx} = \mathbf{A}$	(C6)	$\frac{d\mathbf{a}^\top\mathbf{x}}{dx} = \frac{d\mathbf{x}^\top\mathbf{a}}{dx} = \mathbf{a}^\top$
(C10)	$\frac{d\mathbf{x}^\top\mathbf{A}}{dx} = \mathbf{A}^\top$	(C7)	$\frac{d\mathbf{x}^\top\mathbf{x}}{dx} = 2\mathbf{x}^\top$
(C11)	$\frac{d\mathbf{x}^\top\mathbf{Ax}}{dx} = \mathbf{x}^\top(\mathbf{A} + \mathbf{A}^\top)$	(C8)	$\frac{d(\mathbf{x}^\top\mathbf{a})^2}{dx} = 2\mathbf{x}^\top\mathbf{a}\mathbf{a}^\top$

אופטימיזציות:**הגדרה חד פעמית*-אקטיבציה,עיבוד מקדים, אתחול,
רגולריזציה **דינמיקת האימון*- קצב לימוד, היפר פרמטרים
הערכה סופית- למשל *validation set* או *K-folds*
batch normalization נרמול הקלט- הופך לקל להתכנסות. $X^{N\times D}$ $x_i = \frac{x_i - \mu_i}{\sigma_i}$



כאשר *D* פיצ'רים *N* נקודות. ממוצע זה על כל קורדינטה.
zero center כלומר מרכזים את המידע.
בנקיק צר וארוך או כשהמידע רחוק מהראשית.
scaling normalized data לשנות את הגדלים



יתרונות: 1 מונע *overfitting* –מקרב אותנו לאזור לינארי של פונקציית אקטיבציה.
2 מונע *exploding*- מקטין את הערכים. 3 מונע *vanishing*- הגרדיאנט של ערכים
גדלים בפונקציות אקטיבציה. 4 מזרז התכנסות אימון והופך את האימון לחזק יותר על
ידי בחירת היפר פרמטרים ואתחול משקולות.
חסרונות: המיני בטץ שונה כל פעם, קשה לדבג כי לא יודעים ממוצע. חשוב לזכור את
הערכים גם לשלב הטסטים.

נכניס את*BN* לרשת אחרי *FC* לפני אקטיבציה כי זה מונע חלוקה ב0.
איך למנוע overfitting_1. שימוש רגולריזציה 4. *Early stopping* עצירה
מוקדמת.2. *Data arguments* לשלב דוגמאות חדשות בעזרת הגדלת קלט.
3. להפחית את מורכבות הרשת למשל הורדת מספר השכבות או הנוירונים.
רגולריזציה: למנוע מהמודל להסתגל ל*data* ולמנוע *overfitting*
1.*early stopping* כשמגיעים ל*overfitting* אז מפסיקים ללמוד את המודל.
2.*L2 regularization* נותן עונש על *w* גדול מדיי *Weight decay* עונש
מנורמה 2. $L2(w) = \frac{\lambda}{2}||w||_2^2 + \sum l(w, x^i, y^i)$
הנגזרת $\nabla L(w) = \lambda w + \sum\nabla l(w, x^i, y^i)$ צעד העדכון $w \leftarrow w - \eta[\lambda w + \nabla l(w, x^i, y^i)]$
3.*L1 normalization* סכום הערכים המוחלטים במקום *L2* סכום הערכים בריבוע.
יזהה את המאפיינים החשובים ביותר של ה*data* ומתעלמים מרעשים.

$L1(w) = \lambda||w||_1 + \sum l(w, x^i, y^i) \quad where \quad ||w|| = \sum w_j$

הנגזרת $\nabla L(w) = \lambda sign(w) + \sum \nabla l(w, x^i, y^i)$ צעד העדכון $w \leftarrow w - \eta_{\lambda sign(w)} - \eta[\lambda w + \nabla l(w, x^i, y^i)]$

4.*dropout*-ברנדומליות נחליט אם נבטל דגימה בסיכוי של-*p/1-p* . כל הקשרים מתבטלים גם. נקרא *ensemble learning* כי כל פעם מאמנים על מיני בטץ שונה. במקום לחשב כל אחד, לחשב את

$$E(a) = p^2 * (w_1x + w_2y) + p(1-p) * (0x + w_2y) + p(1-p) * (w_1x + 0y) + (1-p)^2 * (0x + 0y) = p * (w_1x + w_2y)$$

5.*dropout connect*-מחליטים על כל קשת.

6.*data argumentation* – לייצר *data* חדש נסובב, חיתוך ..ה*class* 6-9

mix-up תמונה שמורכבת מ2 תמונות בשקיפות. *cut mix*- עם *crop*.

transfer learning- לקחת רשתות שכבר אומנו ונעתיק את החלקים.

	Very similar database	Very different database
Very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
Quite a lot of data	Fine-tune a few layers	Finetune a larger number of layers

7. *self-supervised learning* – לייצר *data* מתויג אוטומטי.

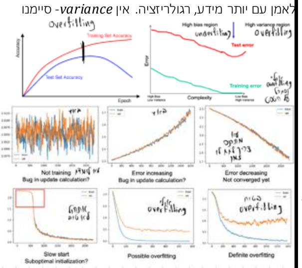
בלמידה מפוקחת יש כמה שלבים כדי לאמן את המודל:

1לאסוף *data* ותגיות. 2לדייק את המודל: לבחור סוג, *loss*, היפר פרמטרים 3אימון המודל: למצוא את הפרמטרים שיתנו לנו את ה*cost* הזול ביותר.**דוגמאות להיפר פרמטרים**: מספר שכבות, מספר נוירונים בכל שכבה, רגולריזציה, עוצמה, קצב למידה, מספר ה*epochs*.

אופטימיזציות של היפר פרמטרים: *Training, validation, test* איך בוחרים? *Grid search* לפרוס את הגריד ולבדוק נקודה ואת כל השילובים עד שנמצא אופטימלי. *random search* להגריל את הנקודות ככה קיבלנו פיזור במרחב

Debugging הרבה *bias*- רשת גדולה, יותר מדי זמן אימון. אין הרבה *bias*- הרבה *variance* – לאמן עם יותר מידע, רגולריזציה. אין *variance*- סיימנו

	Error			
Train	1%	15%	15%	0.5%
Test	11%	10%	30%	1%
	High Variance	High Bias	High B&V	Low B&V



convolution NN: מזהה מאפיינים מקומיים של התמונה.

כמה פילטרים יש בשכבה אחת *K- filter* גובה ואורך הפילטר *F-kernel*

הנוסחה- $input: W \times H \times D \quad output: W = \frac{W-F+2P}{S} + 1 \quad D = K$

$parameters: (F_H \times F_W \times C_{in} + S) \times C_{out} \quad C_{out} = numOfFiltersNow$

$bias = C_{out} \quad Memory(K) = parameters + bias$

$operations: optputeSize \times opsFilter = (C_{out} \times W \times H) \times (F_H \times F_W \times C_{in} + S)$

$memory(KB) = (C_{out} \times H_{out} \times W_{out}) \times 4/1024$

דוגמה : תמונה *32*32* עם *s=1 p=2 10 filters 5*5*3*

$output: 32 = \frac{32 - 5 + 2 * 2}{1} + 1 \quad 32 * 32 * 10$

$parameters: 5 * 5 * 3 + 1 = 76 \; for \; each \; filter \; \& \; 76 * 10 = 760 \; total$

$operations: 32 * 32 * 10 * (5 * 5 * 3 + 1) = 778{,}240 \; ops$

convolution היא פעולה לינארית נוסיף אקטיבציה- זה יאפשר לרשת ללמוד ייצוגים מורכבים יותר של נתוני הקלט. ללא אי לינאריות זו, הרשת תהיה מוגבלת ללמידת פו' אשר לא יספיקו ללכידת בעיות בעולם האמיתי

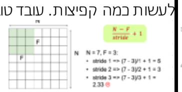
Receptive field – האזור בתמונה שלפיו נבחר את הפילטר.

$RF = 1 + L \times (K - 1) \quad for \; K \times K \; filter \; and \; L \; layers$

דוגמה לגודל של בשכבה ה 2^{nd} כלומר בדוגמה

$7 \times 7 = (5 + 3 - 1) \times (5 + 3 - 1)$ יש לשים לב שמדובר ב$K \times K$ פילטרים שזה 5+3-1

בעיה: אם נרצה שערך יושפע מכל התמונה ולא מחתיכה קטנה בתמונה, צריך לשרשר הרבה שכבות וזה יהיה בזבזני מאוד. פתרון–להקטין את התמונה! *stride* לעשות כמה קפיצות. עובד טוב עם 1,2. עם 3 צריך להוסיף *padding*.



-fully connected layer מספר פרמטרים עצום. לסיווג ורגרסיה. כל יחידה נסתרת מסתכלת על התמונה כולה. כלומר כולם מחוברים לכולם

$parameters: (W \times H \times D + S) * nouirons$

$input: W \times H \times D \quad output: N * 1 - hidden \; layer \; size$

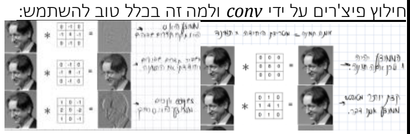
-pooling layer נשתמש כשנרצה לשנות רזולוציה כדי לחסוך זמן ריצה וזיכרון. השוני שאין פרמטרים והפעלת פילטר על כל צ'אנל באופן בלתי תלוי

$input: W \times H \times D \quad output: W = \frac{W - F}{S} + 1 \quad D = D. \quad parameters: 0$

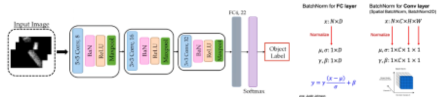
max pooling המקסימום תמיד אותו מקסימום -התמונה זזה.

average pooling לא שומרים מיקום למקס כי גוזרים הכל.

חילוץ פיצ'רים על ידי *conv* ולמה זה בכלל טוב להשתמש:



Batch normalization for fully conv כאשר הקלטים הם לא *zero center* או יש לנו לכל קלט *scaling* שונה או *vanishing exploding* פתרון *batch.*



הופך רשתות עמוקות לקלות לאימון. מאפשר שיעורי למידה גבוהים והתכנסות מהירה. חסרון- מתנהג אחרת במהלך אימון ובדיקה מקור נפוץ לבאגים.

ארכיטקטורות –

AlexNet–*Relu, GPU, dropout regularization*, מיליון וחצי *data*, אימנו מערכת בשבוע שלם. לרשת זו יש פחות שכבות מה שעלול להגביל יכולתה ללמוד תכונות מורכבות במקרים מסוימים.8 שכבות- 5 *conv* ו3 *FC* תוצאות טובות, קלה לתכנות. גודל משאבי מחשוב, רגישות ל*overfitting*.

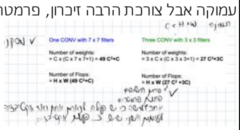
Layer	C (Input size)	H/W (Input size)	filters	kernel	stride	pad	C (Out size)	H/W (Out size)	memory	params	flop
conv1	3	227	96	11	4	0	96	55	1134	35	106
Pool1	96	55		3	2	0	96	27	273	0	0.6
conv2	96	27	256	5	1	2	256	27	547	307	224
pool2	256	27		3	2	0	256	13	127	0	0
conv3	256	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
Flatt.	256	6					9216		36	0	0
FC6	9216		4096				4096		16	37,749	38
FC7	4096		4096				4096		16	16,777	17
FC8	4096		1000				1000		4	4,096	4

עיקר הזיכרון הוא ב*conv* הראשונות.

מספר הפרמטרים העיקרי הוא ב *FC* לעומת *conv*

עיקר החישובים זה ב*conv* כי נדרש לחשב הכל.

VGG –רשת עמוקה ומדויקת, מה שמקל על ההבנה והיישום הוא גדלים אחידים *filter 3*3* .*S=P=1* .*Max pooling 2*2 S=2*. יש 16 שכבות- 13 *conv* ו3 *FC* פילטרים קטנים, מבנה פירמידה כי תמיד מחלקים בחצי כמות *pooling* ביצועים טובים, פחות פרמטרים. איטי לאימון. להימנע *overfitting* שיש ב*AlexNet* כרשת עמוקה אבל צורכת הרבה זיכרון, פרמטרים וחישובים.



GoogleNet רשת שמורכבת מרשתות. רשת עמוקה עם 22 שכבות.*1FC.* *vanilla* ומעשיר מעשיר את כל ה*conv* ובוחר את הפילטר הכי טוב וממשיך הלאה ללא קשר לגודל הפילטר. הרבה פחות פרמטרים מ*VGG* ו מה*AlexNet*. צוואר בקבוק, אי אפשר לעבוד במקביליות, קשה לאימון, יקר. הרעיון הוא לעשות במקביל את הפילטרים, לעבד אותם ולשלוח לרשת הבאה.

Vanilla ארכיטקטורה שמבצעת קבוצת פעולות קבועות כגון *convolution,* *pooling and activation* עם קבוצת פרמטרים קבועים.

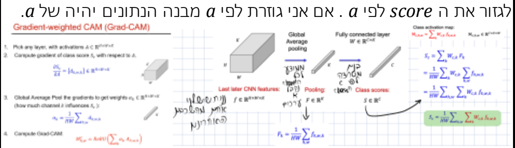
Inception שבה משתמשים במודלים שעובדים במקביל על תכונות שונות.

Stem תחילת הרשת שמחלצת ומעבדת את התכונות הראשוניות כדי לפשט.

ResNet- מודל עמוק ללא *overfitting* כי מעביר בקלט לשכבה הבאה מידע. 152 שכבות. 3*3 *conv*. *skip connection* 2 *conv* ואז קפיצה. ביצועים טובים בזמן סביר. *WideResNet*-הקטנת עומק השכבות והגדלת הפילטרים. *ResNext*-הרבה פילטרים במקביל בעומק 4 ואז סכמו אותם.

Receptive filed- האזור של תמונת הקלט המשפיע על ההפעלה של אותו נוירון, שהנוירון רואה דרך המסננים הקונבולציונים. ברשתות *conv* כל נוירון אחראי על למידה וזיהוי של דפוסים או תכונות בקלט. הבנה יכולה לעזור לנו לקבל תובנות לגבי האופן שבו הרשת מעבדת מידע. ראשונה היא ניתוח*RF* מתייחס לאזור של תמונות הקלט שמשפיע על ההפעלה של אותו נוירון. *deconv* שניה היא טכניקה המשמשת כדי לדמיין את ההפעלות של נוירונים. החלת אותם מסנני *conv* המשמשים במעבר קדימה הפוך- לשחזר את הקלט. לעזור לראות תכונות חשובות להפעלת הנוירון

activation map-קבוצת ערכי הפלט לאחר *conv* - מה משפיע על כל *h,w* מה יש בתמונה שנותן לי לזהות *class* ספציפי. יש גם לאזורי ביניים - לנסות לגזור את ה *score* לפי *a* . אם אני גוזרת לפי *a* מבנה הנתונים יהיה של *a.*

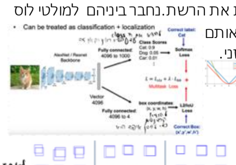


הגרדיאנט של *cam* – להגדיר לכל *channel* כמה הוא חשוב לאותו *class.*

Detection and localization

אתגרים:*מספר שונה של*output* לכל תמונה *סוגים שונים של *output* *גדלים שונים *הרבה חלונות *כמה אובייקטים באזור.

אובייקט יחיד: נכניס לרשת ידועה שלא לומדת את הרשת.נחבר ביניהם למולטי לוס שזה הסכום שלהם אבל ממושקל (אותו כוח, אותם סדרי גודל). שתי רשתות שונותיהיה יותר בזבזני.

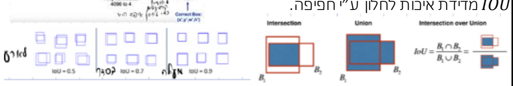


$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$

Multitask loss

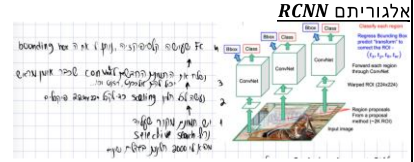
$L_{reg}(b, \hat{b}) = \sum_{i=\{x,y,w,h\}} smooth_{L_1}(b_i - \hat{b}_i)$

IOU מדידת איכות לחלון ע"י חפיפה.

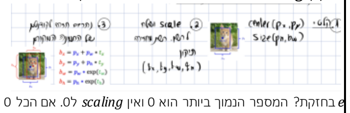


ריבוי אובייקטים-לחלק להרבה חלונות ולזהות ברשת עבור כל חלון. לקבל *classification* ונדרש גם לעשות רגרסיה כי יכול להיות שהחלון שהכנסתי לא מדויק והרגרסיה יכולה לתקן לי את החלון. בשביל לעשות את זה צריך לעשות *crop* לתמונה בגודל $h \times w$. אפשרויות למיקומי *x* : *W-w+1* אפשרויות למיקומי *y*: *H-h+1*. $\sum\sum(W - w + 1)(H - h + 1) = O(W^2H^2)$ גם אם ניקח תמונה קטנה ונחשב נראה שיוצא מספר גדול וזו פעולה יקרה.פתרון *RP*–ניחוש אזורים יש אובייקט.

אלגוריתם *RCNN*



נריץ *selective search*. נגדיל את החלונות שהם *RF* באופן לא אחיד כדי שיהיה לי 224 פיקסלים *scaling*. נשלח ל*convnet* שכבר אומן מראש. תיקון *scaling* הוא כדי לא לנרמל את הקורדינטות ולקבל את ההזה של החלון.



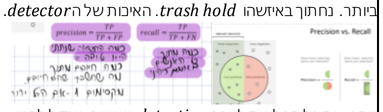
e בחזקת? המספר הנמוך ביותר הוא 0 ואין *scaling* ל0. אם הכל 0 אז נשאר עם הגודל המקורי ו*output=proposal* . ב*L2* אל תשנה את הגודל אם לא חייבים וזה עוזר להתכנסות.

אימון *RNN*-מפה נתאים בין ההצעה לחלון המקורי *RL to GT* באופן הבא לכל *RP*:

חיובי עבור *IOU>0.5 with GT*. שלילי עבור*IOU<0.3 with GT* אחרת נטרלי.

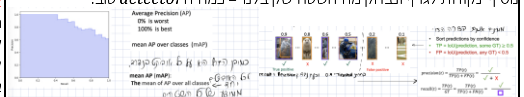
את כל החיובים נגזור. אם זה חיובי הרשת תיתן לי את ה*class* וגם את ה*transform* לגודל החדש (רגרסיה לחלון) בשלילי – נגזור רק את הסיווג ללא רגרסיה לחלון.

טסטים*RNN* - ניקח תמונה ונציע הצעות 2000. נעשה לכל חלון *scale*, נכנסי לרשת ונקבל טרנספורמציה ו*class*.נקבל את הסיכויים של כל *class* כמובן וניקח את הגבוה ביותר. נחתוך באיזשהו *trash hold*. האיכות של ה*detector*.

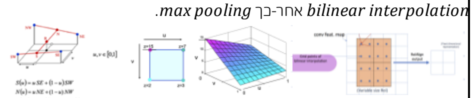


ניקח את כל החלונות לאחר *detection* ממוינים מגדול לקטן.

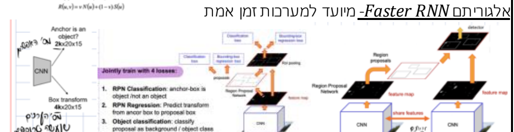
נוסיף נקודות לגרף ונבדוק מה השטח שקיבלנו = כמה ה*detector* טוב.



אלגוריתם *fast RNN*-איטי– עד 2000. *RP* חופפים. נריץ על הפיצ'רים עצמם. מפת הפיצ'רים. מהיר. מה שנשאר זה להבין איך עושים את ה *scale* בפיצ'ר *space*. נעשה *bilinear interpolation* אחר-כך *max pooling*.

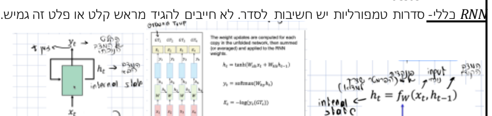


אלגוריתם *Faster RNN*- מיועד למערכות זמן אמת



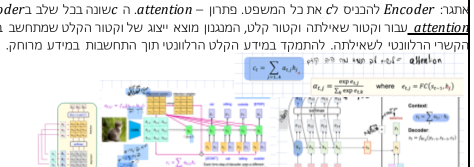
לממש *RP* על ידי רשת. כדי לחסוך עיבוד נשתמש באותם פיצ'רים. פעם אחת *RP* ופעם *crop*. לכל *RP* נקרא *anchor box* גודל קופסה אחד עם גודל קבוע מראש. עבור כל חלון –האם יש אובייקט אם כן מה הטרנספורמציה. הבעיה שהאנקור יכול להיות גדול מגודל האובייקט. או כמה אובייקטים שנופלים באותו מקום. פתרון-מאמנים מספר *anchor k* לכל כניסה בגדלים שונים. מאמנים את הרשת עם 4 *losses*: בתמונה למעלה.

RNN כללי- סדרות טמפורליות יש חשיבות לסדר. לא חייבים להגיד מראש קלט או פלט זה גמיש.



אתגר: *Encoder* להכניס ל*c* את כל המשפט. פתרון – *attention*. ה *c*שונה בכל שלב ב*decoder.*

attention עבור וקטור שאילתה וקטור קלט, המנגנון מוצא ייצוג של וקטור הקלט שמתחשב במידע הקשרי הרלוונטי לשאילתה. להתמקד במידע הקלט הרלוונטי תוך התחשבות במידע מרוחק.



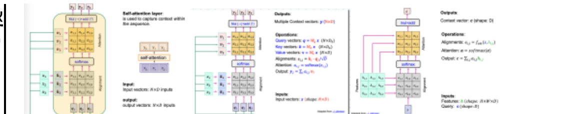
רכיב *attention* כללי:רכיבי w_k, w_v הם נלמדים

Inputs: $X = [x_{1 \ldots}]^T \quad ; shape \; N \times D \quad ; \quad Q = [q_{1 \ldots}]^T \quad ; shape \; M \times D_k$

Outputs: $y_j = \sum_i a_{i,j}v_i \; ; \; Y = A^TX \; ; shape \; (M \times D)$

רכיב *attention*:
Operations:
Key vectors: $k_i = W_k x_i$ (shape: D_k)
Value vectors: $v_i = W_v x_i$ (shape: D_v)
Alignments: $e_{i,j} = k_i \cdot q_j / \sqrt{D_k}$ $(N \times M)$
Attention: $a_{*,j} = \mathrm{softmax}(e_{*,j})$ $(N \times M)$

ברכיב *self-attention*:
Operations:
Alignments: $e_{i,j} = x_i \cdot q_j / \sqrt{D}$
$E = XQ^T/\sqrt{D}$ (shape: $N \times M$)
Attention: $a_{*,j} = \mathrm{softmax}(e_{*,j})$
$A = \mathrm{softmax}(E, dim = 1)$ $(N \times M)$



Self-Attention את ה *q*ניקח מתוך ה*x* וזו מטריצה נלמדת. מקבל *x* ומוציא *v*

input: $N \times D$ *inputs vector ;output:* $N \times D$ *outputs vector ;learned:* w_k, w_v, w_q

הטמעה של וקטור הקלט לתוך וקטורים שכל אחד מתייחס למידע הקשרי

query מחושב מהקלט בטרנספורמציה לינארית כך $q_i = W_q x_i$ המשימה שנעריך.

key מחושב מהקלט בטרנספורמציה לינארית כך $k_i = W_k x_i$ המפתח מציין עבור כל קלט את הייצוג שיש להשוות מול השאילתה. הדמיון בין המפתח לשאילתה מחושב כך $e_{ij} = q_i k_j$ או $e_{ij} = \frac{Q_i * k_j}{\sqrt{D}}$ נחלק בשורש די כדי שזה יהיה בלתי תלוי בממד. וסט נורמאלי אמור להסתכם ל1. $\widehat{e_{ij}} = \frac{e_{ij}}{\sum e_{ij}}$

value מחושב מהקלט בטרנספורמציה לינארית כך $v_i = W_v x_i$ מייצג את הקלטים. עבור כל שאילתה הקלט המשויך מוערך כך $y_j = \sum e_{ij} v_i$

לגבי הממדים- הקלט והפלט מורכבים מסדרה של *N* וקטורים ממימד *D* . ה*query* הוא ממימד D_k.

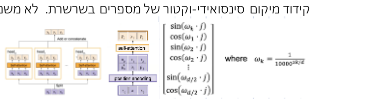
הממדים הם $k \epsilon R^{D_k} \; ; \; q \epsilon R^{D_k} \; ; \; v \epsilon R^D \; ; \; x \epsilon R^D$

סכום הערכים $\sum_{i,j} a_{i,j} = N$ *sum* שזה מספר ה*input vector*

פרמוטציה על *x* לא משפיע אם משנים את הסדר על *q,v,y*

אבל עם זאת אם משנים את הקלטים זה משפיע על הפלט.

פתרון-*positional encoding* וקטור שנוסיף לרשת כדי שנדע מה המיקום. היחס בין המיקומים. קידוד מיקום סינוסאידי-וקטור של מספרים בשרשרת. לא משנה איפה הוא נמצא. תדירויות שקטנות



Multi head attentions- להריץ במקביל מספר *n*ואז לחבר או לשרשר.

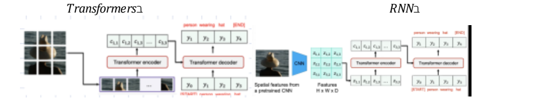
transformers- יכולים לעבד רצפים שלמים של *attention* בבת אחת. זה מושג באמצעות *self-attention* המאפשר למודל להתמקד בחלקים שונים של רצף הקלט בזמן שהוא מעבד אותו. עבודה מקבילית. *transformers*: נבנה כ *sec to sec* כמו תרגום למשל

Input: image I
Output: $y = y_1, y_2, y_3, \ldots, y_T$
Encoder: $c = T_E(z)$ where z is spatial CNN features and $T_E(\cdot)$ is the transformer encoder
Decoder: $y_t = T_D(y_{0:t-1}, c)$ where $T_D(\cdot)$ is the decoder transformer

encoder מקבל את כל המשפט.

decoder מקבל רישא ומוציא את המילה הבאה.

ב*Transformers* ב*RNN*



באורך משתנה, רצפים ארוכים ובעיית שיפוע נעלם *RNN*. לעבד במקביל *transformers*. יתכן שאינם יעילים עבור משימות הדורשות מידול מפורש של סדר הרצף. הרבה תמונות לאמן, *self-attention* יעבוד טוב. פחות תמונות לאמן *CNN*.

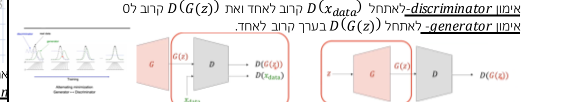
GAN

ב*discriminative* נקבל תמונה *x* ופרדיקציה ל*y*. נחפש את מפריד המחלקות. ב*GAN* לא נדע את הסיכויים של *x*. נתאר את פונקציית ההתפלגות במחלקות.

הכלים של המודלים הגנרטיביים-שניהם נמצאים בתוך *Maximum Likelihood*. *explicit* נותן לך תמונה *x* ונבדוק מה הסיכוי של *x*. *implicit* לא יודעים מה הסיכוי ל *x*אבל אנחנו יכולים לדגום מתוך $p(x)$ כי הרשת יודעת מה *p*. נגריל מספר בהתפלגות נורמאלית סביב ה0 ונקרא לזה *z* זה הווקטור הלטנטי. נעביר את *z* אוסף של טרנספורמציות כדי להביא אותו מההתפלגות הידועה להתפלגות של מה שאני רוצה- למשל התפלגות פרצופים. כאילו דגמתי מתוך ההתפלגות ה*g*. הרשת מעבירה אותי מהתפלגות להתפלגות. θ הוא משקל הרשת שצריך למצוא אותו על מנת לחשב את ה*loss*.

אימון *discriminator*-לאתחל $D(x_{data})$ קרוב לאחד ואת $D(G(z))$ קרוב ל0

אימון *generator*- לאתחל $D(G(z))$ בערך קרוב לאחד.



אימון ה*GAN* בתמונה השנייה ניקח שמאלה כי הפריור יותר גבוה, נעדכן בהתאמה *decimator* וחוזר חלילה. עד שהוא חצי וזה אופטימום שיווי משקל.

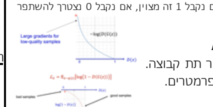
חישוב פונקציות *loss*

discriminator $L_D = \mathbb{E}_{x \sim p(x)}[-logD(x)] + \mathbb{E}_{z \sim q(z)}[-\log(1 - D(G(z)))]$

כאשר $D(x)$ זהו מידע אמיתי והמידע החדש $D(G(z))$

בעיה: אם נקבל 1 זה מצוין, אם נקבל 0 נצטרך להשתפר *generator* $L_G = \mathbb{E}_{z \sim q(z)}[\log(1 - D(G(z)))]$

פתרון: *generator* $L_G = \mathbb{E}_{z \sim q(z)}[-\log D(G(z))]$



הכללי הוא *minmax formulation* $L = \max_G \min_D L_D$

בעיות עם *GAN* : *Mode collapse*- יכול למדל רק חלק כלומר תת קבוצה.

יציבות- פרמטרים יכולים להתפצל. התנהגות רגישה להיפר פרמטרים.

הכנסות- לפעמים לא מתכנס לצורה שרצינו.

DCGAN- deep convulsion GAN לוקחים תמונה קטנה ומגדילים. עד שמקבלים את התמונה הסופית. אפשר לעשות אריתמטיקה במרחב הלטנטי.תמונות ברזולוציות נמוכות.

progressive growing Gan רזולוציה נמוכה ונגדיל בהדרגה. השלבים המוקדמים של האימון

style GAN מפרידה בין הסגנון ברמה הגבוהה של תמונה לבין המבנה ברמה נמוכה של התמונה