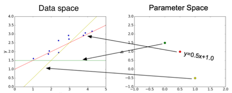
**Linear regression**- בעיית רגרסיה לערכים רציפים. נרצה למצוא מודל = הקו הלינארי שייתן לנו מרחק מינימלי מכל הנקודות. y=wx+b



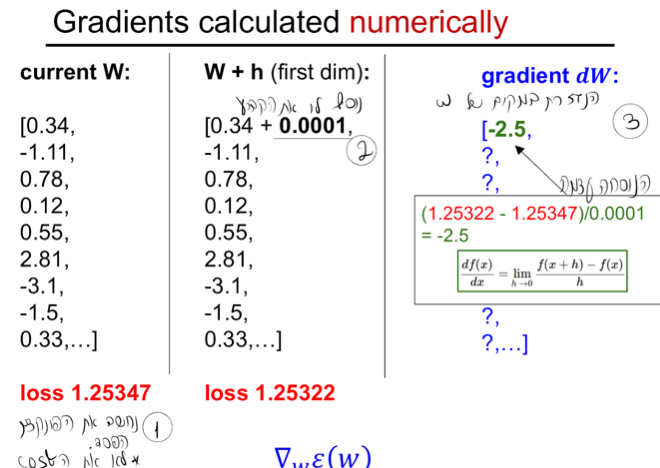
LSE(lose)=

MSE(cost)=

בצורת וקטור:

כדי למזער cost נמצא אופטימיזציה לפרמטרי המודל:

1.direct solution-

שיטה נומרית- קל לביצוע, מאוד איטי, לא מדויק (שגיאת עיגול)  


שיטה אנליטית- גזירת סימבולים. מהיר ומדויק, מיועד לשגיאות.

הבעיה- גרסה לינארית בלבד. אם T מאוד גדול (פרמטרים רבים) ובנוסף גם

אי אפשר לעשות תמיד נגזרות עדיף לרדת בשיפוע.

2. פתרון iterative GD-נקודת מינימום גלובלי בטופוגרפיה. קל ליישום ותמיד 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.  
**עדכון הdata שיטות:** עוצרים באופטימום נקודות ה0 או לא מתעדכן.  
\* batch- החזרה- .   
מעדכנת את המשקולות לאחר עיבוד כל נקודות הנתונים בכל מחזור אימון (השיפוע מחושב על פני כל נקודות הנתונים בכל איטרציה). סיבוכיות o(n)

\* SGD- החזרה- *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. ב* batch *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- ההחזרה סיבוכיות o(N)

**קצב הלמידה:** משמש להחלטת גודל צעד העדכון בכל איטרציה.

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 על פני גיא או שנתקענו במינימום לוקלי או אזור שטוח עקב vanishing או

נקודת אוכף (הנגזרת 0) או בהתכנסות איטית כאשר צעד העדכון קטן מדיי.

עדכון המומנטום   
 צריך להיות 0.9 או 0.99 פחות מ1 כי אם יהיה גדול מ1 המומנטום יגדל שוב ושוב עד לexploding ולא נשיג מינימום. קרוב ל0 זה יצטמצם לסטנדרט.

 Nestrov -טכניקה נוספת: במקום להשתמש 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 או מתפוצצים. RMSProp -   
אפסילון כדי שלא נחלק באפס. ככל שs גדול -צעד קטן יותר.

Adam – קומבינציה של RMSProp וmomentum- נשאר זהה.

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:*

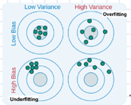
*1.קצב למידה נמוך. חפש קצב למידה טוב יותר rate decayאו אדפטיבי.*

*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 ניקח שורש של המימד לפני (השכבה לפני) המטרה= שונות של הקלט שווה לפלט. ככל ש גדול אז שינוי קטן מאוד ישפיע על הvariance של הoutput . המעבר הוא ואז בגלל ההנחה שwx לא תלוים וזירו מין )ביאס הוא אפס) ולהציב את השונות הידועה של w והכל מצטמצם.

Glorot&Bengio אולי השונות שונים אז עושים ממוצע ביניהם. אתחול ביאס בשניהם: 0 כי לא פוגע.  
Relu מקרים שהערכים הם 0 בחוויאר ולכן נשתמש בזה.

**Linear regression** **None**- פרדיקציה מכמה ממדים y=

רגרסיה פולינומיאלית– מגדירים פיצ'רים במקום x המודל

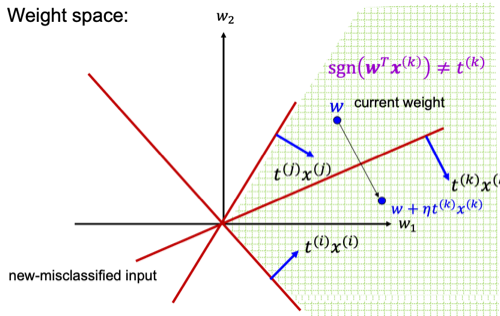
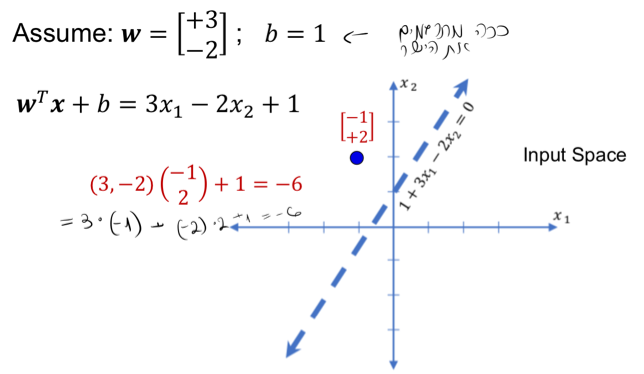
נבחר מודל שייתן לנו התאמה טובה (לא יתרה ולא תת התאמה)

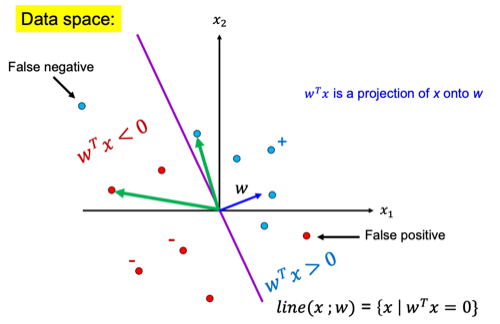
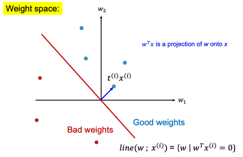
**bias** סטיה מהמרכז. **Variance** כמה הדגימות מפוזרות.

השגיאה: ככל שמספר פרמטרים

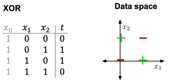
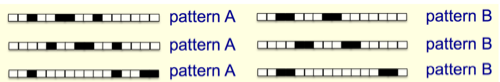
עולה= שגיאתBias תרד. גודל המודל יעלה את var

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



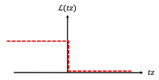
הגבלות מסווג לינארי-   
patterns – ניקח ממוצע של פטרןA : 0.25 לכל כניסה(הכניסות השחורות). הווקטור הממוצע לכל ההזזות הוא 0.25,0.25… נניח שהוא מצליח לסווג ומה שמקבל ערך גדול מ0 זה A. אז

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 :**

**01 loss** - אם התנאי יתקיים 1 אחרת 0.

כאשר z=wx+b וגם t=true label

העלות

לפי הגרף הנגזרת שווה ל0- לא יתעדכןw -נגזרת מתאפסת.

**perceptron**-

****t=c היא האמיתי, z הוא הפרדיקציה

הנגזרת

צעד העדכון

 SGD כי זה רק נקודה אחת.

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

**Multi class**- לכל מחלקה יש מסווג f1,..fk =z  
כתיבה כללית:   
 *המחלקה האמיתית לעומת j שזה הפרדיקציה החיזוי שלי.*

פרספטרון-

עדכון SGD -

בפועל: להוריד ציונים שגבוהים מהמחלקה ולהעלות את המחלקה. יש נטרלים.

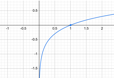
**logistic regression**   
שיטה סטטיסטית לרגרסיה בינארית, תוחם את המשוואה לטווח  .תכונות:

\*מה הסיכוי שt=1

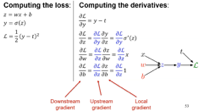
0<y<1

\*התבוננות-

\*סיכום

   
Loss שימושי הוא**ML**

הנגזרת

**צעד העדכון עם SGD

מחיר

**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

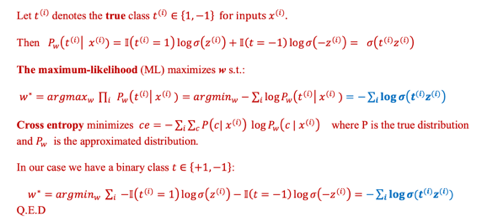
עבור n מחלקות כאשר זו התווית האמיתית ו זה הSoftMax

בבינארי זה

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**



KL-מדד להפרש בין שתי התפלגויות הסתברות. אין קשר למודל.

אנטרופיה  *היא קבועה ולא משפיעה על w.*

  
**חוסר יציבות נומרית**

אם t=1 ו z<0 אז 0 yואז log0



**SoftMax -**ציונים לאחוזים.

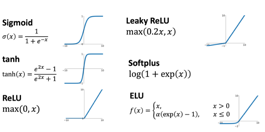
מובטח לי שכל ערך הוא בין 0 ל1 והמכנה 1.

**נגזרת: או

עדכון SGD

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.* 

*w2 הפך להיות המסווג.* *=*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 :*

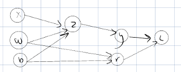
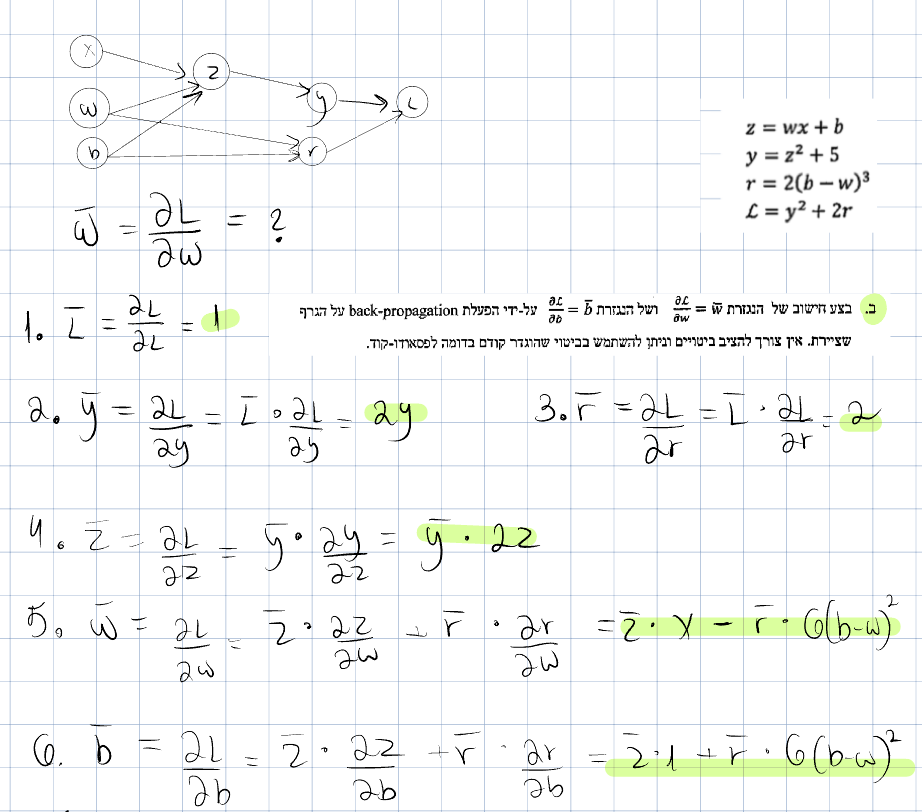
tanh *טווח [-1,1]. הוא כן זירו סנטר. כבד חישובית. הנגזרת שלילית נוירונים רווים הורגים את הגרדיאנט. נגזרת :*

Relu *קל לחשב. אינו רווי באזורי z>0 הנגזרת לא מתאפסת. סיבוכיות יעילה. בעיה שלא זירו סנטר-> לפתור עם מיני בטץ. נגזרת :*

*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 כoutput. סיבוכיות exp.*

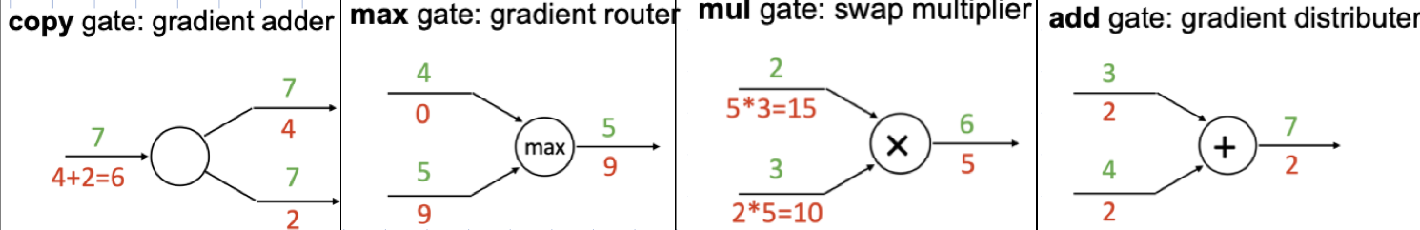
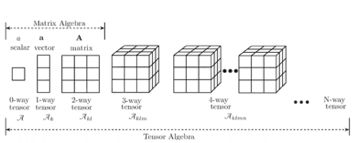
****backpropagation** *מחשב שיפועים עבור GD. שיטה לבצע נגזרות בMLP  
*

*Forward המעבר הראשון מקלט ללוס*

*Backward passכל חישוב הנגזרות*

*Local החישוב שעליו מדובר ברגע הנוכח*

*תבניות התנהגות הגרדיאנט:*

**

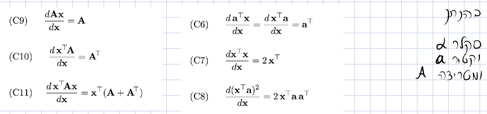
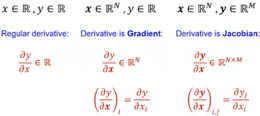
***טנזור****- ייצוג כללי של ערכים ממימד כלשהו.*

***אקטיבציה****-מפעילים על כל איבר בנפרד לכן אותם ממדים.*

***נגזרת של סקלר לפי סקלר****= סקלר*

***נגזרת של סקלר לפי וקטור****= הגרדיאנט = וקטור*

*M\*N****נגזרת של וקטור לפי וקטור****= מטריצה שגודלה-*

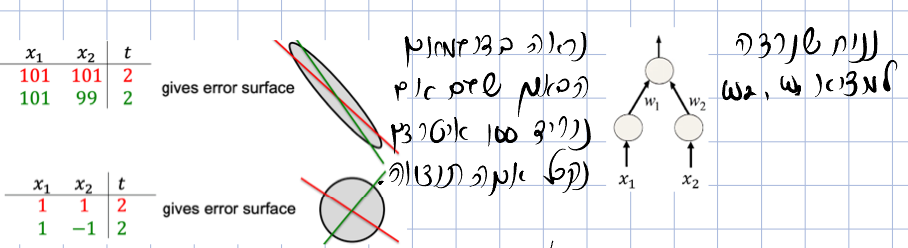
*בהינתן סקלר אלפא, וקטור a, ומטריצה A*

*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. כאשר D פיצ'רים N נקודות. ממוצע זה על כל קורדינטה.  
zero center כלומר מרכזים את המידע.*

*בנקיק צר וארוך או כשהמידע רחוק מהראשית.*

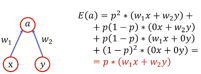
*scaling normalized data לשנות את הגדלים  
*

*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 BNלרשת 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 לdata ולמנוע overfitting*

*1.early stopping כשמגיעים לoverfitting אז מפסיקים ללמוד את המודל.*

*2.L2 regularization נותן עונש על w גדול מדיי Weight decay עונש מנורמה 2.   
הנגזרת צעד העדכון   
3.L1 normalization סכום הערכים המוחלטים במקום L2 סכום הערכים בריבוע. יזהה את המאפיינים החשובים ביותר של הdata ומתעלמים מרעשים.  
   
הנגזרת צעד העדכון*

*4.dropout-ברנדומליות נחליט אם נבטל דגימה*

*בסיכוי של p/1-p . כל הקשרים מתבטלים גם.*

*נקרא ensemble learning כי כל פעם מאמנים*

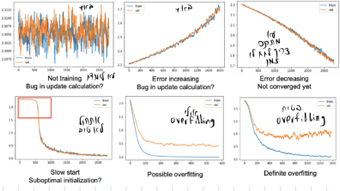
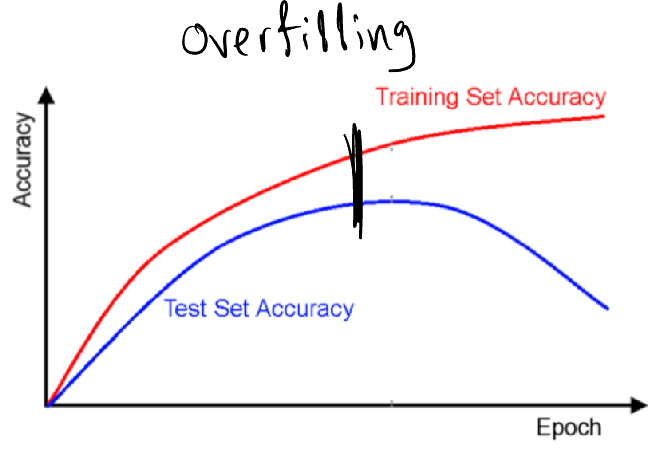
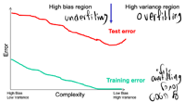
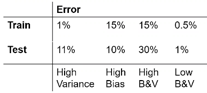
*על מיני בטץ שונה. במקום לחשב כל אחד, לחשב את הרשת ולהכפיל בp.  
5.dropout connect -מחליטים על כל קשת.*

*6.data argumentation – לייצר data חדש נסובב, חיתוך ..הclass 6-9  
mix-up תמונה שמורכבת מ2 תמונות בשקיפות. cut mix- עם crop.  
transfer learning- לקחת רשתות שכבר אומנו ונעתיק את החלקים.*

*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 הcost הזול ביותר.****דוגמאות להיפר פרמטרים****: מספר שכבות, מספר נוירונים בכל שכבה, רגולריזציה, עוצמה, קצב למידה, מספר הepochs.****אופטימיזציות של היפר פרמטרים****: Training, validation, test איך בוחרים? Grid search לפרק 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*

**

***NN convolution****: מזהה מאפיינים מקומיים של התמונה.*

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

*הנוסחה-*

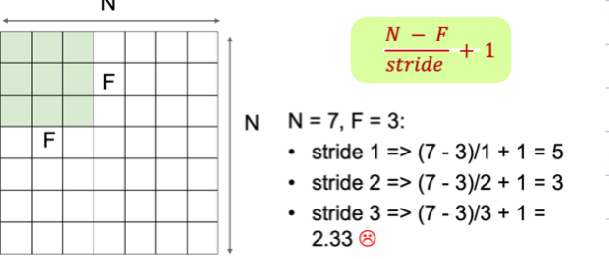
*דוגמה : תמונה 32\*32 עם filters 5\*5\*3 10 s=1 p=2*

*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.*

*דוגמה לגודל של בשכבה ה כלומר בדוגמה*

*יש לשים לב שמדובר ב פילטרים שזה 5+3-1*

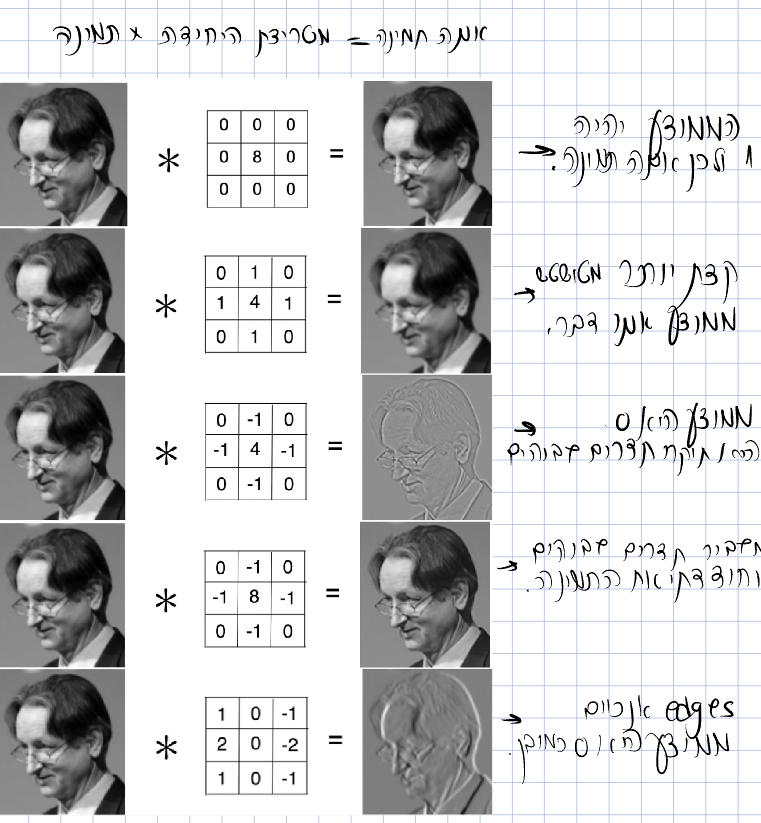
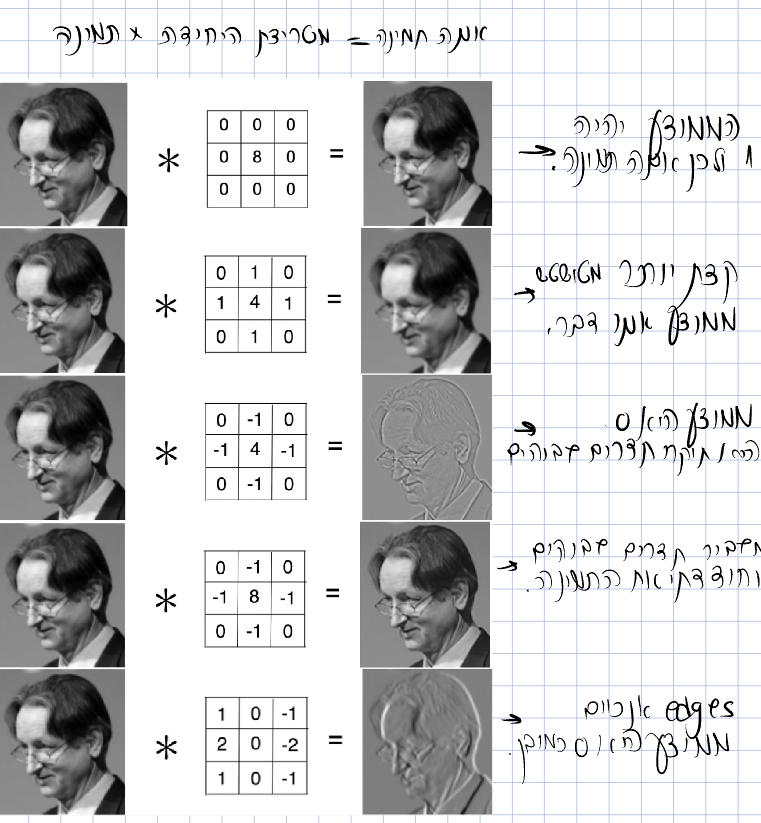
*בעיה: אם נרצה שערך יושפע מכל התמונה ולא מחתיכה קטנה בתמונה, צריך לשרשר הרבה שכבות וזה יהיה בזבזני מאוד. פתרון– להקטין את התמונה! stride לעשות כמה קפיצות. עובד טוב עם 1,2. עם 3 צריך להוסיף padding.   
*

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

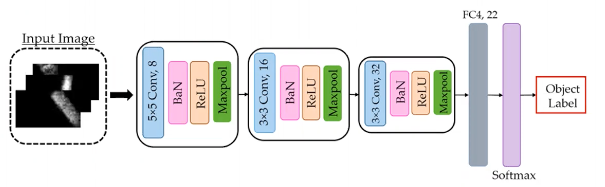
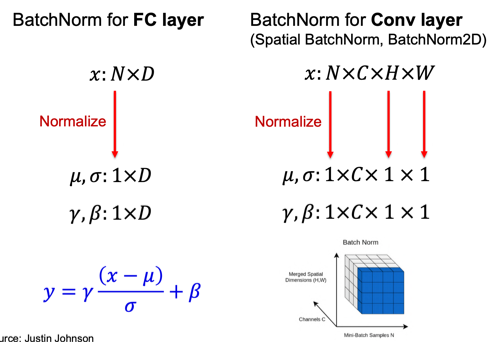
***pooling layer-*** *נשתמש כשנרצה לשנות רזולוציה כדי לחסוך זמן ריצה וזיכרון. השוני שאין פרמטרים והפעלת פילטר על כל צ׳אנל באופן בלתי תלוי.*

*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 –***

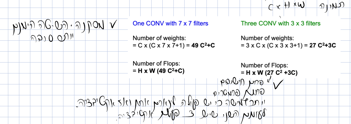
*AlexNet –GPU, Relu, dropout regularization, מיליון וחצי data, אימנו מערכת בשבוע שלם. לרשת זו יש פחות שכבות מה שעלול להגביל יכולתה ללמוד תכונות מורכבות במקרים מסוימים.8 שכבות-5 conv ו3 FC תוצאות טובות, קלה לתכנות. גודל משאבי מחשוב, רגישות לoverfitting.*

*עיקר הזיכרון הוא בconv הראשונות.*

*מספר הפרמטרים העיקרי הוא בFC לעומת conv*

*עיקר החישובים זה בconv 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 FCפילטרים קטנים, מבנה פירמידה כי תמיד מחלקים בחצי כמות pooling ביצועים טובים, פחות פרמטרים. איטי לאימון. להימנע overfitting שיש בAlexNet כרשת עמוקה אבל צורכת הרבה זיכרון, פרמטרים וחישובים.*

**

*GoogleNet רשת שמורכבת מרשתות. רשת עמוקה עם 22 שכבות.1FC.*

*vanilla ומעשיר מעשיר את כל הconv ובוחר את הפילטר הכי טוב וממשיך הלאה ללא קשר לגודל הפילטר. הרבה פחות פרמטרים מVGG ו מהAlexNet. 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.*

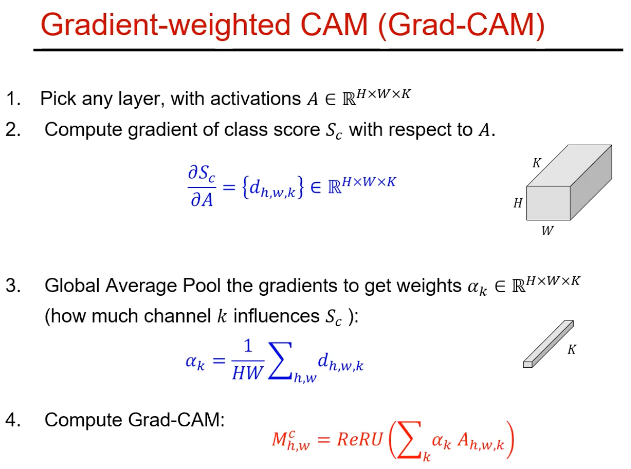
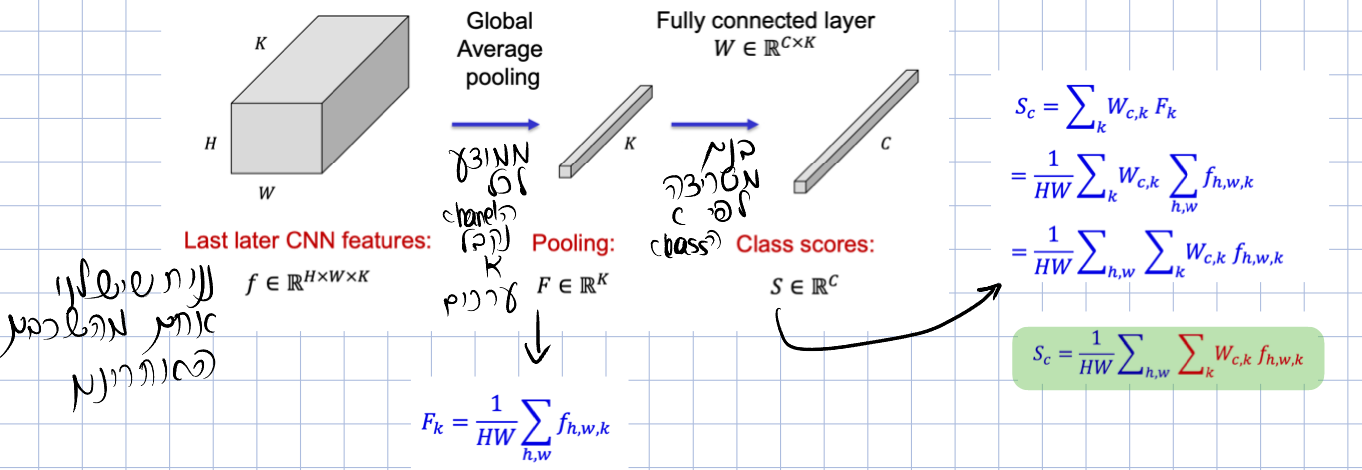
*Stem תחילת הרשת שמחלצת ומעבדת את התכונות הראשוניות כדי לפשט.*

*ResNet- מודל עמוק ללא overfitting כי מעביר כקלט 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 RFמתייחס 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*

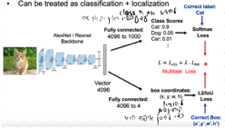
*מה יש בתמונה שנותן לי לזהות class ספציפי. יש גם לאזורי ביניים - לנסות לגזור את הscore לפי a . אם אני גוזרת לפי a מבנה הנתונים יהיה של a.*

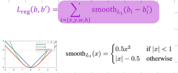
**

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

***Detection and localization***

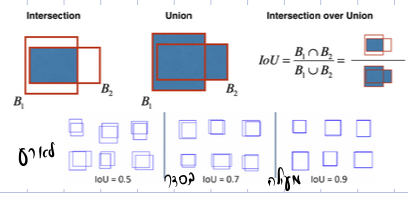
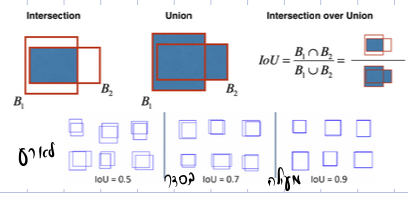
*אתגרים:\*מספר שונה של output לכל תמונה \*סוגים שונים של output \*גדלים שונים \*הרבה חלונות \*כמה אובייקטים באזור.*

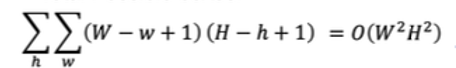
*אובייקט יחיד: נכניס לרשת ידועה שלא לומדת את הרשת.נחבר ביניהם למולטי לוס שזה הסכום שלהם אבל ממושקל (אותו כוח, אותם*

* סדרי גודל). שתי רשתות שונותיהיה יותר בזבזני.*

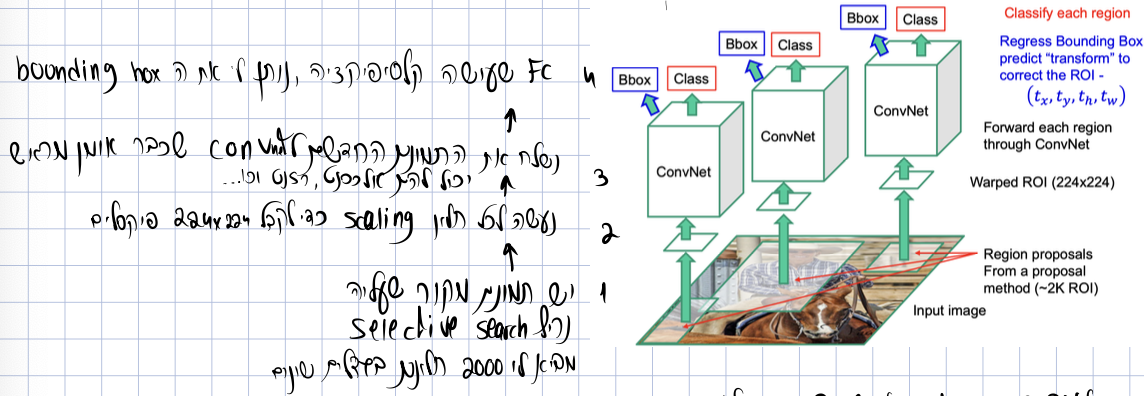
***Multitask loss***

*IOU מדידת איכות לחלון ע״י חפיפה.*

**

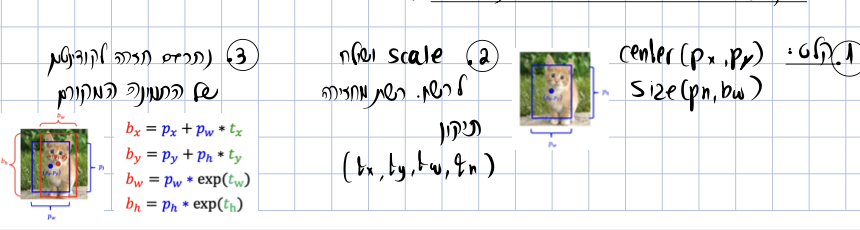
*ריבוי אובייקטים-לחלק להרבה חלונות ולזהות ברשת עבור כל חלון. לקבל classification ונדרש גם לעשות רגרסיה כי יכול להיות שהחלון שהכנסתי לא מדויק והרגרסיה יכולה לתקן לי את החלון. בשביל לעשות את זה צריך לעשות crop לתמונה בגודל .אפשרויות למיקומי x : W-w+1 אפשרויות למיקומי y: H-h+1. גם אם ניקח תמונה קטנה ונחשב 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 לconvnet שכבר אומן מראש.*

*תיקון scaling הוא כדי לא לנרמל את הקורדינטות ולקבל את ההזזה של החלון.*

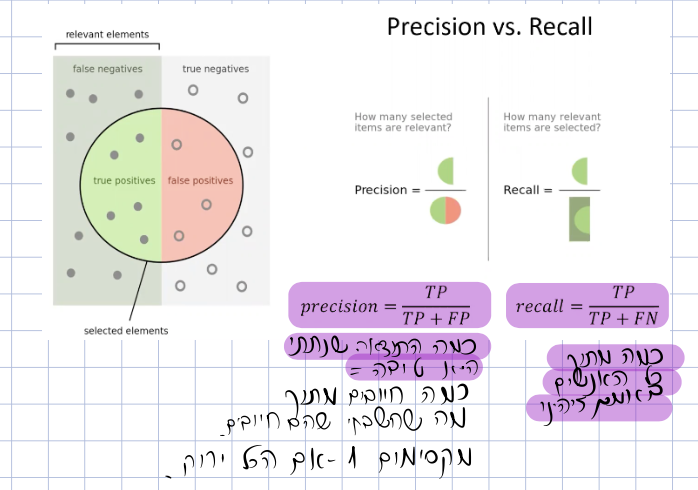
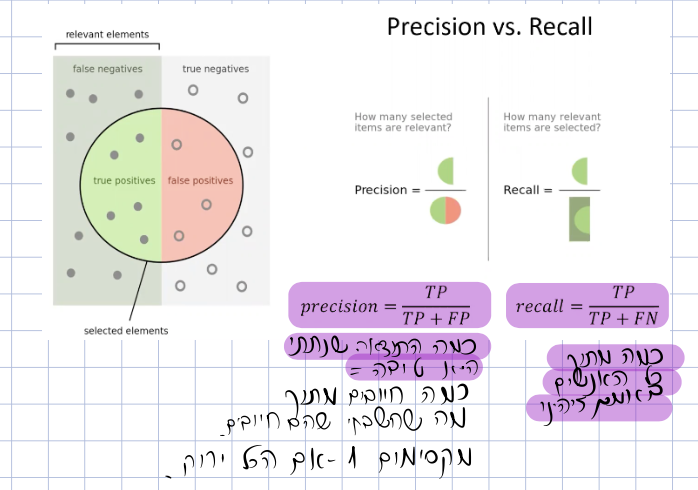
**

*e בחזקת? המספר הנמוך ביותר הוא 0 ואין scaling ל0. אם הכל 0 אז נשאר עם הגודל המקורי וoutput=proposal . בL2 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 עבורIOU<0.3 with GT אחרת נטרלי.*

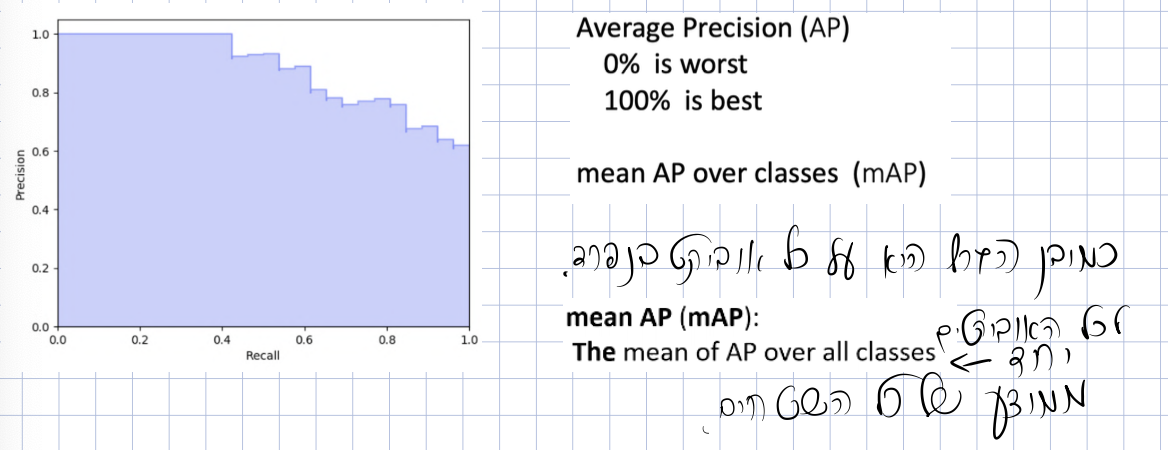
*את כל החיובים נגזור. אם זה חיובי הרשת תיתן לי את הclass וגם את הtransform לגודל החדש (רגרסיה לחלון) בשלילי – נגזור רק את הסיווג ללא רגרסיה לחלון.*

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

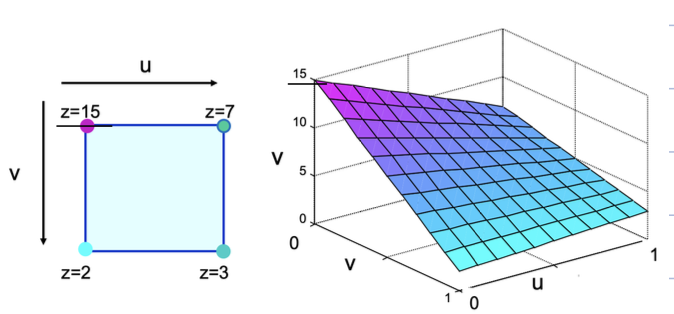
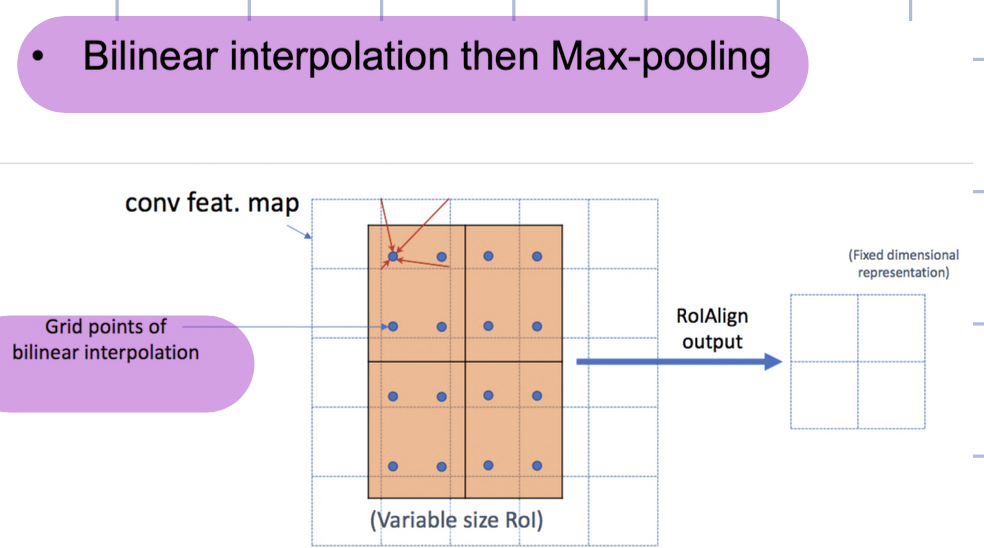
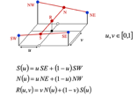
**

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

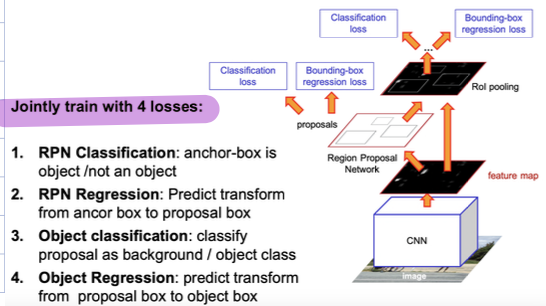
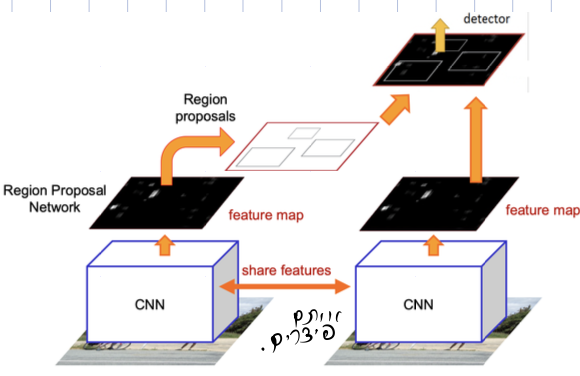
*נוסיף נקודות לגרף ונבדוק מה השטח שקיבלנו = כמה הdetector טוב.*

**

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

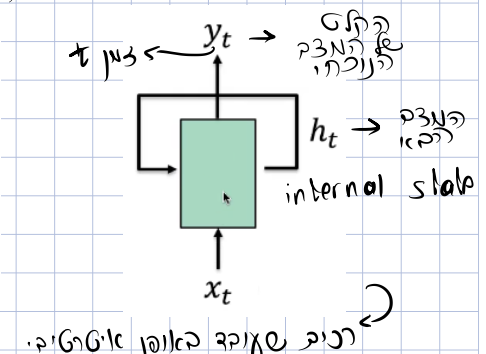
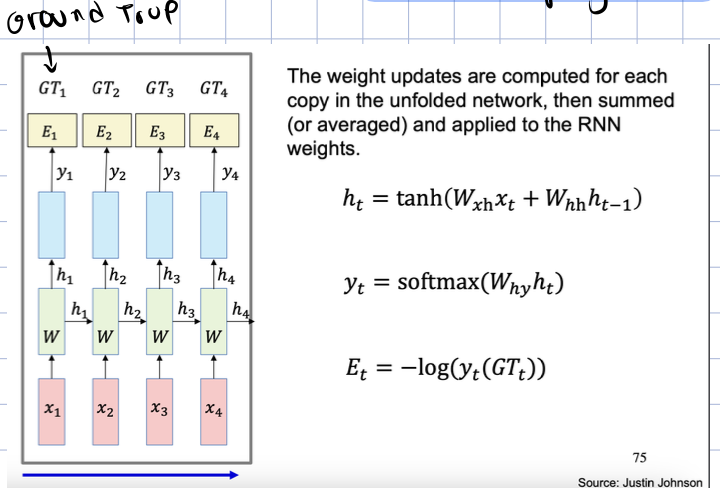
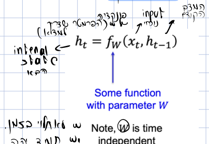
**

*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 ואם כן מה הטרנספורמציה. הבעיה שהאנקור יכול להיות גדול מגודל האובייקט. או כמה אובייקטים שנופלים באותו מקום. פתרון-מאמנים מספר k anchor לכל כניסה בגדלים שונים. מאמנים את הרשת עם 4 losses: בתמונה למעלה.*

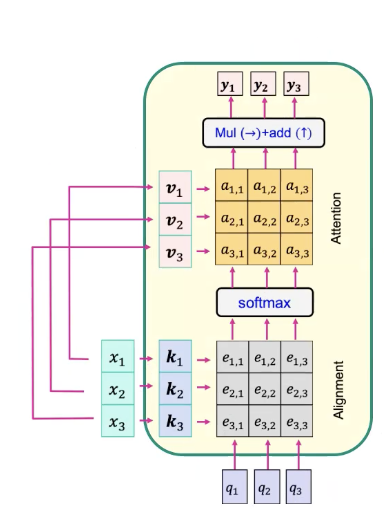
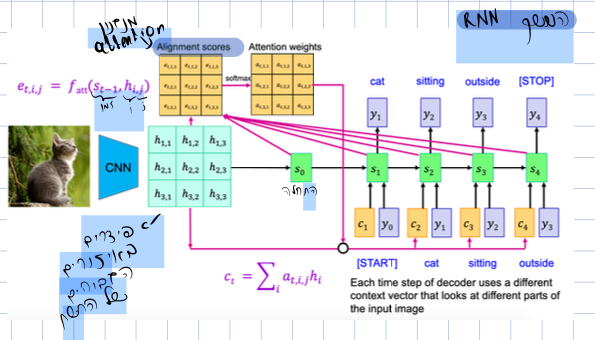
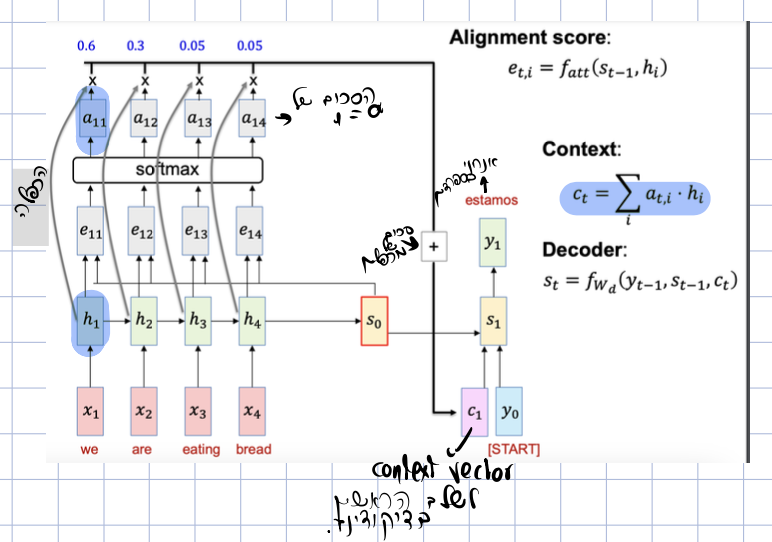
*RNN כללי- סדרות טמפורליות יש חשיבות לסדר. לא חייבים להגיד מראש קלט או פלט זה גמיש.*

**

*אתגר: Encoder להכניס לc את כל המשפט. פתרון – attention. הc שונה בכל שלב בdecoder.*

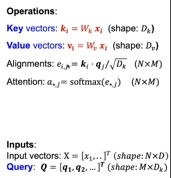
*attention עבור וקטור שאילתה וקטור קלט, המנגנון מוצא ייצוג של וקטור הקלט שמתחשב במידע הקשרי הרלוונטי לשאילתה. להתמקד במידע הקלט הרלוונטי תוך התחשבות במידע מרוחק.*

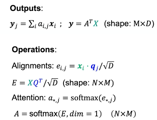
**

**

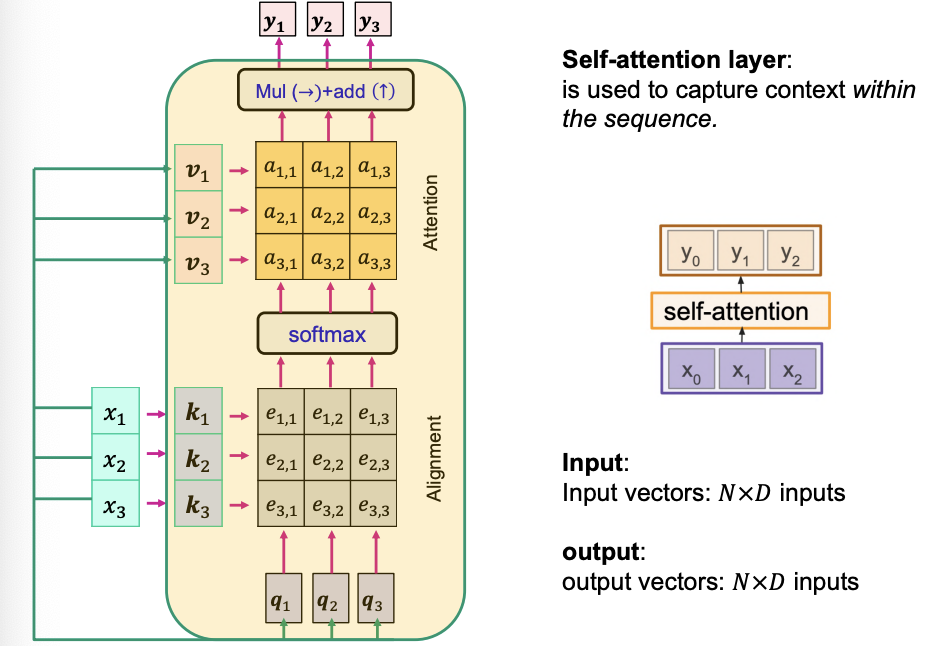
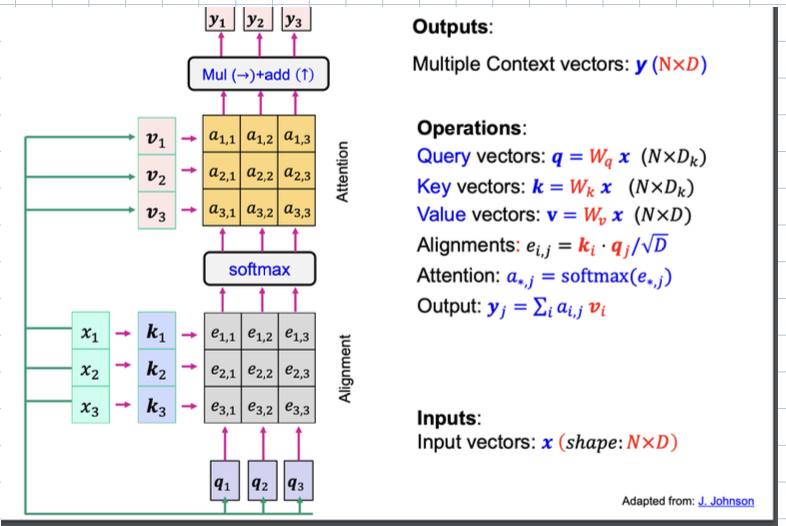
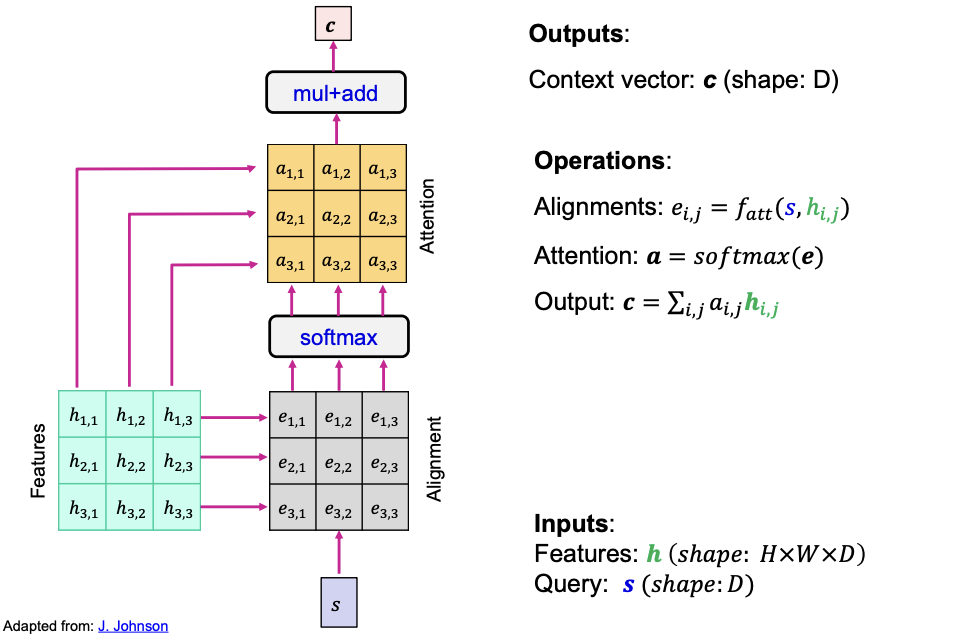
*רכיב attention כללי:רכיבי הם נלמדים*

*Inputs:*

*Outputs:*

**

*רכיב attention בתוך רשת RNN: רכיב attention כללי : ברכיב self-attention:*

**

*Self-Attention את הq ניקח מתוך הx וזו מטריצה נלמדת. מקבל x ומוציא v*

*input: ;output: ; learned:*

*הטמעה של וקטור הקלט לתוך וקטורים שכל אחד מתייחס למידע הקשרי*

*query מחושב מהקלט בטרנספורמציה לינארית כך המשימה שנעריך.*

*key מחושב מהקלט בטרנספורמציה לינארית כך המפתח מציין עבור כל קלט את הייצוג שיש להשוות מול השאילתה. הדמיון בין המפתח לשאילתה מחושב כך או נחלק בשורש די כדי שזה יהיה בלתי תלוי בממד. וסט נורמאלי אמור להסתכם ל1 .*

*value מחושב מהקלט בטרנספורמציה לינארית כך מייצג את הקלטים. עבור כל שאילתה הקלט המשויך מוערך כך*

*לגבי הממדים- הקלט והפלט מורכבים מסדרה של וקטורים ממימד . הquery הוא ממימד .*

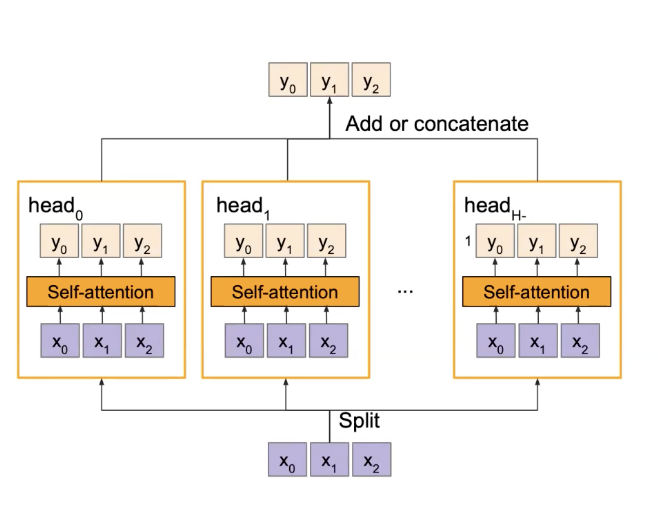
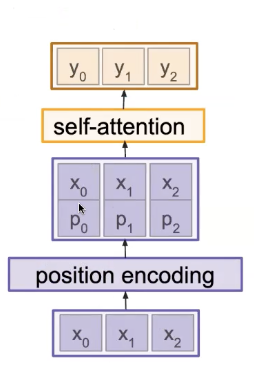
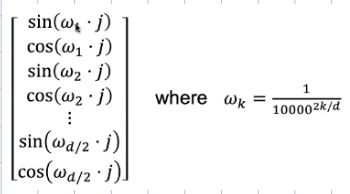
*הממדים הם*

*סכום הערכים שזה מספר הinput 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 סינסואידי-וקטור של מספרים בשרשרת. לא משנה איפה הוא נמצא. תדירויות שקטנות*

**

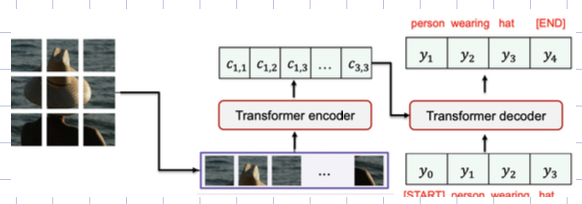
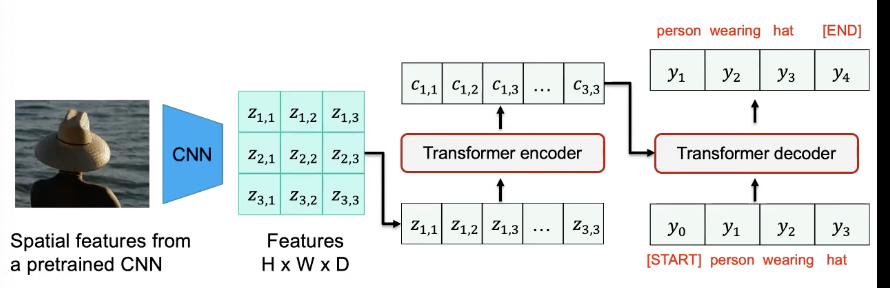
*Multi head attentions- להריץ במקביל מספרn ואז לחבר או לשרשר.*

*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 כsec to sec כמו תרגום למשל*

*encoder מקבל את כל המשפט.*

*decoder מקבל רישא ומוציא את המילה הבאה.*

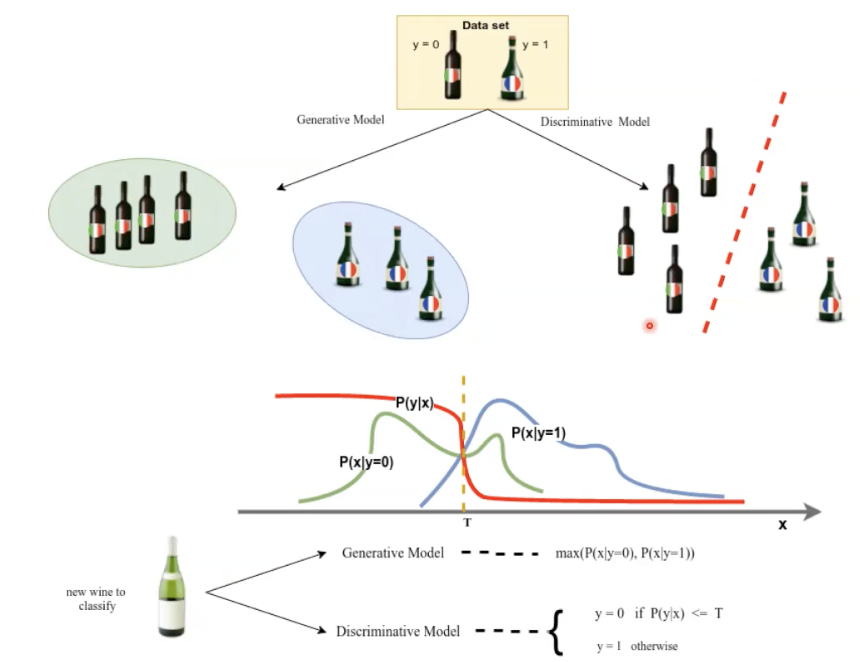
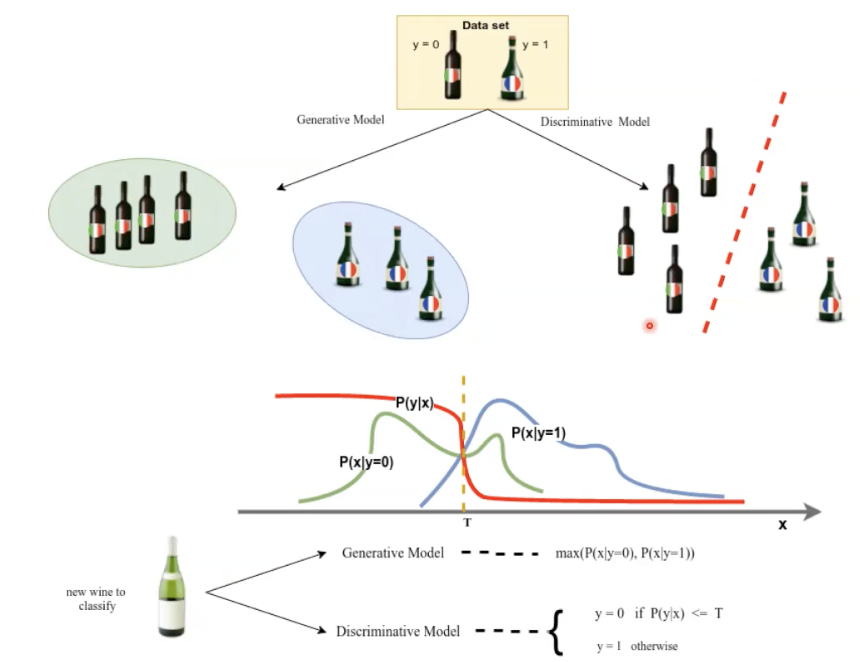
*בRNN בTransformers*

**

*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.*

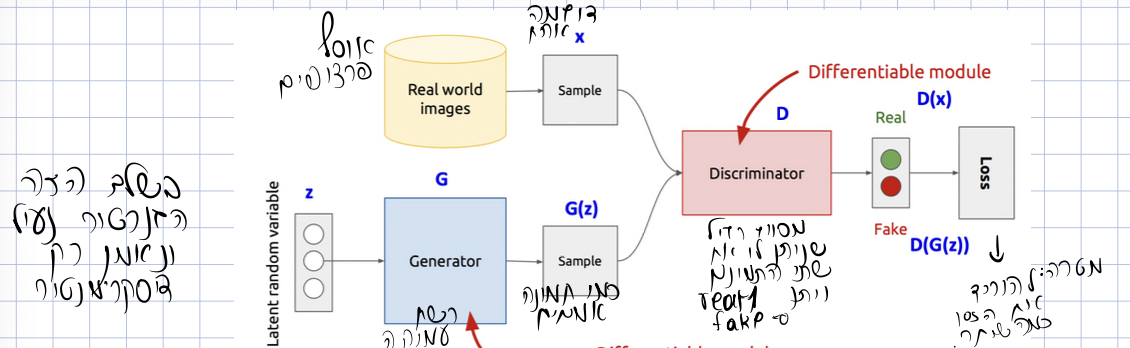
***GAN***

*בdiscriminative נקבל תמונה x ופרדיקציה לy. נחפש את מפריד המחלקות. בGAN לא נדע את הסיכויים של x. נתאר את פונקציית ההתפלגות במחלקות.*

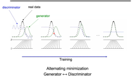
**

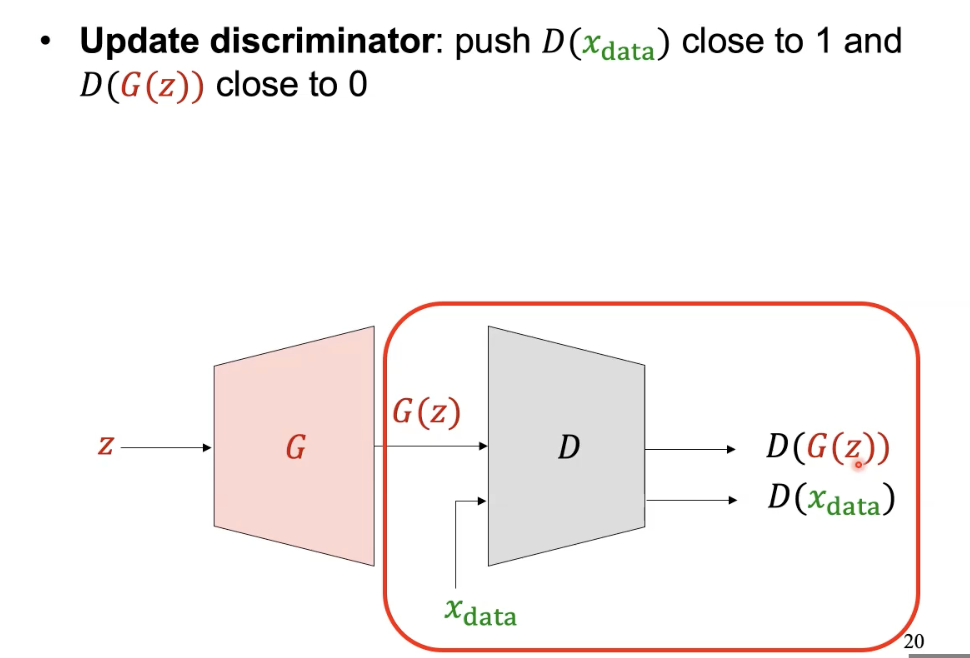
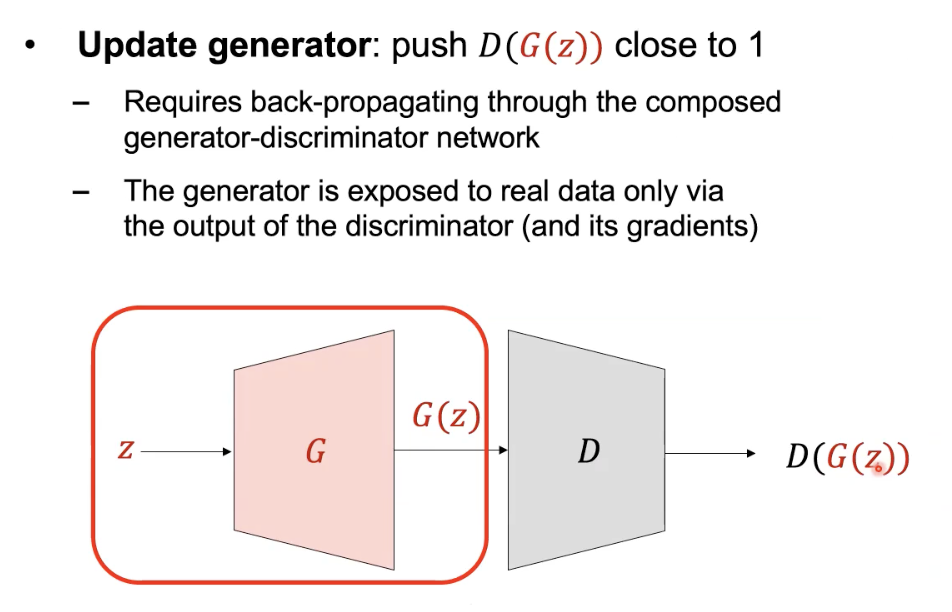
*הכלים של המודלים הגנרטיביים-שניהם נמצאים בתוך Maximum Likelihood. explicit נותן לך תמונה x ונבדוק מה הסיכוי של x. implicit לא יודעים מה הסיכוי לx אבל אנחנו יכולים לדגום מתוך כי הרשת יודעת מה p. נגריל מספר בהתפלגות נורמאלית סביב ה0 ונקרא לזה z זה הווקטור הלטנטי. נעביר את z אוסף של טרנספורמציות כדי להביא אותו מההתפלגות הידועה להתפלגות של מה שאני רוצה- למשל התפלגות פרצופים. כאילו דגמתי מתוך ההתפלגות הg.*

*הרשת מעבירה אותי מהתפלגות להתפלגות.*θ *הוא משקל הרשת שצריך למצוא אותו על מנת לחשב את הloss.*

**

*אימון discriminator-לאתחל קרוב לאחד ואת קרוב ל0*

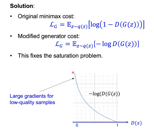
*אימון generator- לאתחל בערך קרוב לאחד.*

**

*אימון הGAN בתמונה השנייה ניקח שמאלה כי הפריור יותר גבוה, נעדכן בהתאמה*

*decimator וחוזר חלילה. עד שהוא חצי וזה אופטימום שיווי משקל.*

*חישוב פונקציות loss  
discriminator*

*כאשר זהו מידע אמיתי והמידע החדש*

*generatorבעיה: אם נקבל 1 זה מצוין, אם נקבל 0 נצטרך להשתפר*

*פתרון: generator*

*הכללי הוא minmax formulation*

*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*