

Дипломски рад

Детекција карата за игру коришћењем YOLO алгоритма за детекцију објекта

Студент: Алекса Величковић 576/2015

Професор: др Владимир Миловановић

18. септембар 2020.

Универзитет у Крагујевцу

Факултет инжењерских наука



Садржај

Увод

Припрема тренинг скупа

Добијање слика из видеа

Означавање знака и броја карте на слици

Прављење скупа за тренирање

Вештачке неуронске мреже - теорија

Увод у вештачке неуронске мреже

Конволуцијске неуронске мреже

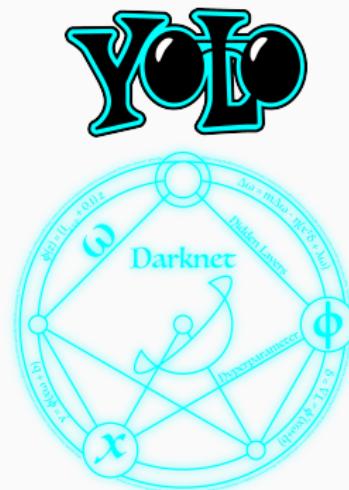
YOLO алгоритам

Тренирање - Darknet програмски оквир

Закључак

YOLO алгоритам и програмски оквир Darknet

- YOLO (You Only Look Once) - алгоритам за детекцију објекта у реалном времену
- Darknet је програмски оквир и представља имплементацију YOLO алгоритма



Садржај

Увод

Припрема тренинг скупа

Добијање слика из видеа

Означавање знака и броја карте на слици

Прављење скупа за тренирање

Вештачке неуронске мреже - теорија

Увод у вештачке неуронске мреже

Конволуцијске неуронске мреже

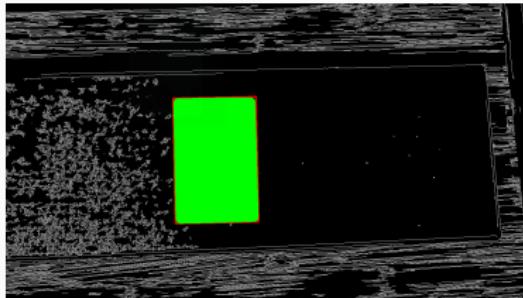
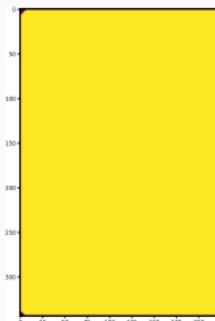
YOLO алгоритам

Тренирање - Darknet програмски оквир

Закључак

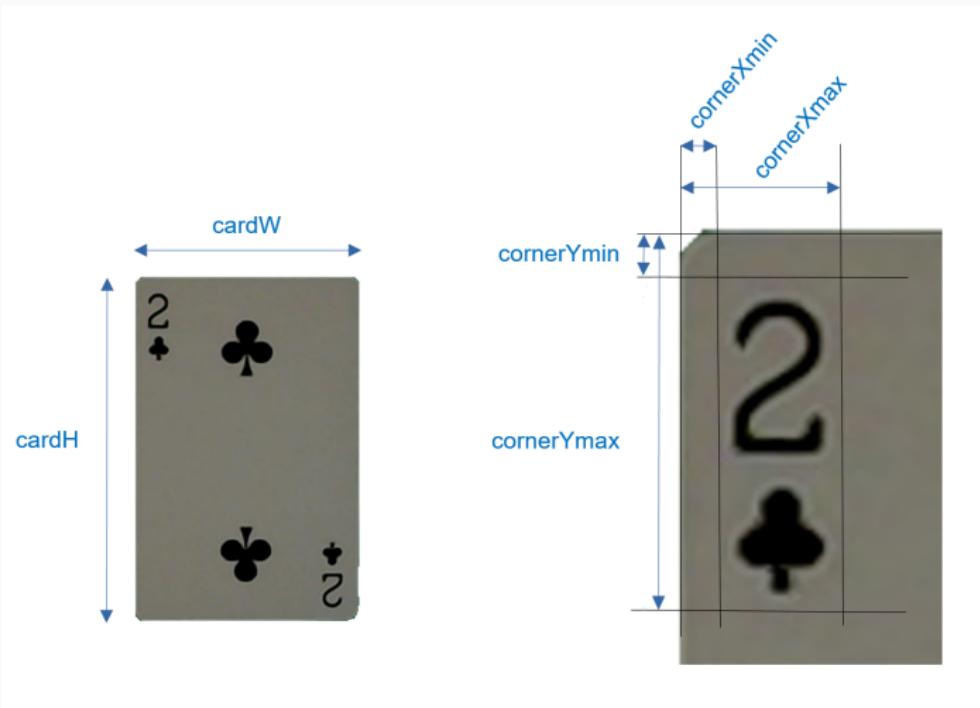
Добијање слика из видеа

- Шаблон за издвајање карте са снимка
- Функција `extract_card()` издваја карту са прослеђене слике
 - Провера замућења слике; смањење шума; издвајање ивица (прелаза) са слике и налажење контура (спаја све тачке дуж границе, исте боје и интензитета)
 - Требало би да је највећа контура карта; проверава се да ли је приближно правоугаоног облика
 - Простор унутар контуре се претвара у правоугаоник димензија карте (шаблона)



Означавање знака и броја карте на слици

- Дефинисане су позиције ћошкова правоугаоника који окружује знак и број карте



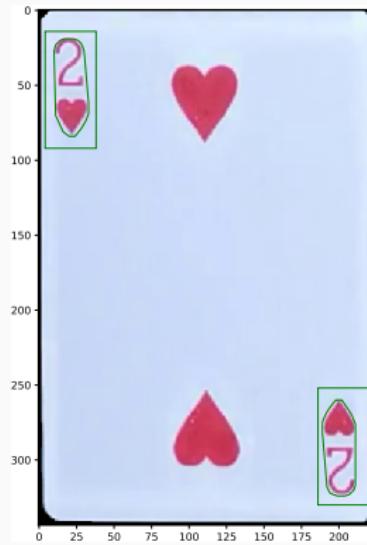
Означавање знака и броја карте на слици

- Функција `find_hull()` унутар правоугаоника налази полигон (контуру), унутар кога се налазе знак и број
 - Издаваја се део слике карте означен позицијама правоугаоника; ивице (прелази) се „појачавају”; налазе се контуре и конвексни полигони око њих
 - Врше се провере да ли је: контура већа од минимално дефинисане (`min_area`); однос површина контуре и конвексног полигона већи од минималног (`min_solidity`) и да ли је тежиште контуре близу центра правоугаоника



Означавање знака и броја карте на слици

- Све контуре су спојене у једну и одеђен је полигон око ње
- Уколико је површина полигона у границама минималне и максималне дефинисане вредности, позицију полигона треба дефинисати у односу ивицу карте



Прављење скупа за тренирање

- Скуп за тренирање се генерише подвлачењем једне од много позадина испод карата, а оне се распоређују на два начина:
 - Насумичним транслирањем, ротацијом и увећавањем **две карте**
 - Извешавањем трансформација за две карте, а потом трансформацијом све **три карте** карте као групе, после чега се налазе једна до друге као када се држе у руци



Садржај

Увод

Припрема тренинг скупа

Добијање слика из видеа

Означавање знака и броја карте на слици

Прављење скупа за тренирање

Вештачке неуронске мреже - теорија

Увод у вештачке неуронске мреже

Конволуцијске неуронске мреже

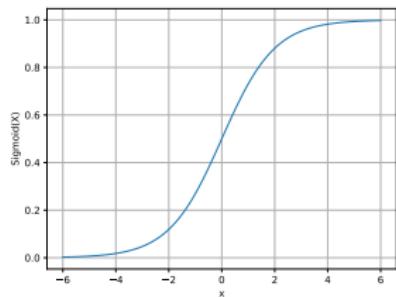
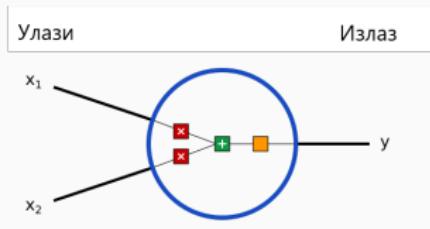
YOLO алгоритам

Тренирање - Darknet програмски оквир

Закључак

Увод у вештачке неуронске мреже

- **Неурон** је основна јединица вештачке неуронске мреже и има своје улазе, над којима извршава рачунске операције и добијену вредност уписује као излаз



- Формула за израчунавање вредности излаза је $y = f(x_1 * w_1 + x_2 * w_2 + b)$, где је f **активациона функција** (нпр. Сигмоидна функција, дефинисана као $S(x) = \frac{1}{1+e^{-x}}$)
- Процес прослеђивања улазних вредност напред назива се **пропагација унапред**

Увод у вештачке неуронске мреже

- Неуронска мрежа је скуп међусобно повезаних неурона



- Функција губитка L и функција трошка J
- Тренирање мреже је поступак промене вредности тежине веза и прагова активације са циљем да се одреди **што нижа функција трошка**
- На пример, функција трошка може бити *средња квадратна грешка* (*mean square error*) и дефинише се као:

$$J = MSE = \frac{1}{m} \sum_{i=1}^m (y_{true} - y_{pred})^2 \quad (1)$$

Увод у вештачке неуронске мреже

- Функција трошка може да се представи као функција са више променљивих: $J(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$
- Промена вредности w_1 , мења вредност J , па је потребно израчунати парцијални извод и своди се на чланове које је могуће израчунати и добија се:

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1} \quad (2)$$

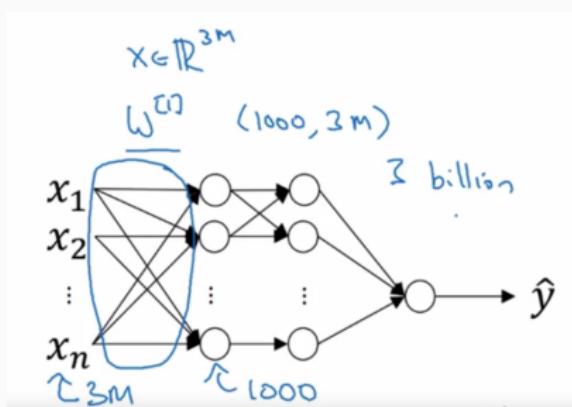
- Поступак рачунања парцијалних извода идући од излаза ка улазу неуронске мреже назива се **пропагација уназад**
- Алгоритам **Стохастички градијентни спуст** би могао да се употреби за минимизовање функције трошка J :

$$w_1 \leftarrow w_1 - \eta \frac{\partial J}{\partial w_1} \quad (3)$$

- η - дужина корака при учењу (learning rate)

Конволуцијске неуронске мреже

- Рачунарски вид (Computer vision)
 - Аутономна вожња, препознавање лица...
- Проблем - слике (подаци) на улазу могу да буду велике
 - Нека је слика на улазу димензија $1000 \times 1000 \times 3$, а први слој мреже има 1000 неурона, тада би матрица тежина веза између прва два слоја имала димензије (1000,3000000) и израчунавање постаје неизводљиво



Конволуцијске неуронске мреже

- Операција **конволуције** је једна од основа за прављење конволуцијске неуронске мреже.
- Вредност поља $(1,1)$ у излазној матрици једнака је збиру производа чланова исечка (димензија филтера), улазне матрице и чланова филтера. Исечак се помера у страну (следећи ред) и извршава се исти поступак.

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	0	1	2	7	4
-1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

$\downarrow *$ "convolution"

3×3
filter

=

-5			

4×4

6×6

Конволуцијске неуронске мреже

- Први корак при детекцији лица је **детектовање ивица**



- Детектовање **вертикалних** ивица

- Прелаз **светло-у-тамно**

$$\begin{array}{|c|c|c|c|c|c|} \hline & \downarrow & \downarrow & & & \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & \boxed{10} & \boxed{10} & \boxed{0} & 0 & 0 \\ \hline 10 & \boxed{10} & \boxed{10} & \boxed{0} & 0 & 0 \\ \hline 10 & \boxed{10} & \boxed{10} & \boxed{0} & 0 & 0 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array}_{3 \times 3} = \begin{array}{|c|c|c|c|} \hline & \downarrow & & \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & \boxed{30} & 30 & 0 \\ \hline \end{array}_{4 \times 4}$$

Diagram illustrating the convolutional step for vertical edge detection. A 6x6 input image (labeled 6x6) is convolved with a 3x3 kernel (labeled 3x3). The result is a 4x4 output image (labeled 4x4). The input image has values 10 and 0 in its 3x3 receptive field. The kernel has values 1, 0, -1. The output value 30 is highlighted in red. Below the input image is a small diagram showing a vertical edge being processed by a 3x1 kernel.

Конволуцијске неуронске мреже

- С обзиром да постоји **много различитих ивица**, а њихова детекција се врши применом различитих филтера тј. вредности **поља матрице филтера** су различите, могуће их је посматрати као **параметре за тренирање**
- Допуњавање матрице (padding)**
 - мање информација које се односе на средину матрице
 - чување информација из поља са ивице слике

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \end{bmatrix} \quad (4)$$

Конволуцијске неуронске мреже

- Конволуција са већим кораком (Strided convolution) се разликује у кораку (stride) који исечак прави кроз улазну матрицу.

The diagram shows a 7x7 input matrix being convolved with a 3x3 filter. The input matrix has stride=2, as indicated by the blue boxes and arrows. The result is a 2x2 output matrix.

2	3	7	3	4	4	6	4	2	9
6	6	9	1	8	0	7	2	4	3
3	4	8	-1	3	0	8	3	9	7
7	8	3	6	6	6	3	4		
4	2	1	8	3	4	6			
3	2	4	1	9	8	3			
0	1	3	9	2	1	4			

$\frac{7 \times 7}{2 \times 2}$

*

3	4	4
1	0	2
-1	0	3

3×3

=

q ₁	l ₀₀	

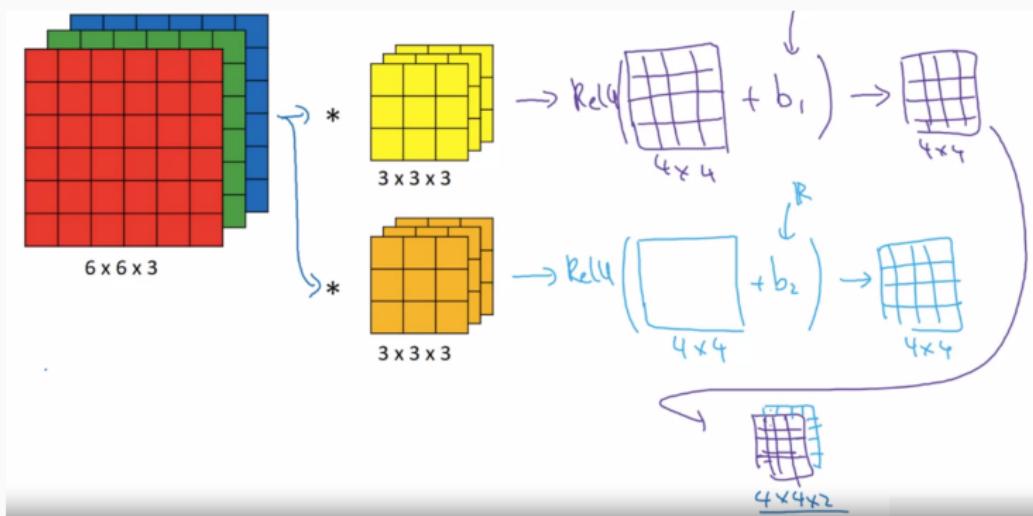
Stride = 2

- Нека је слика димензија $n \times n$, а филтер $f \times f$, корак је s , а допуна p , тада је димензија излазне матрице:

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

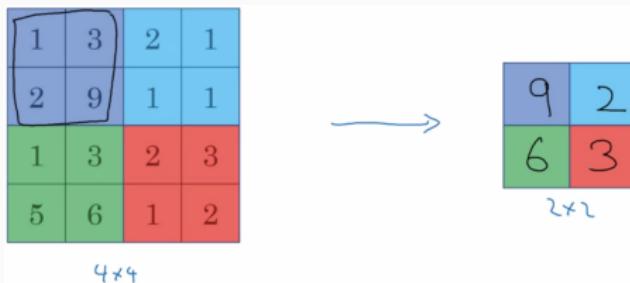
Конволуцијске неуронске мреже

- Пример једног слоја конволуцијске неуронске мреже
- Колико год да је слика велика, број параметара за тренирање је дефинисан димензијама филтера!



Конволуцијске неуронске мреже

- Врло често је матрица на излазу веома велика и можда ју је потребно смањити, а да притом главне одлике буду сачуване. Ово се постиже **Слојевима удружилања (Pooling layers)** и слика приказује један тип који се назива **Удружилање максимума (Max pooling)**.
- Максимална вредност у исечку се уписује као излазна
 - већа вредност од осталих у исечку представља одлику и остаје сачувана
 - уколико су вредности приближне, уписана вредност је само мало већа

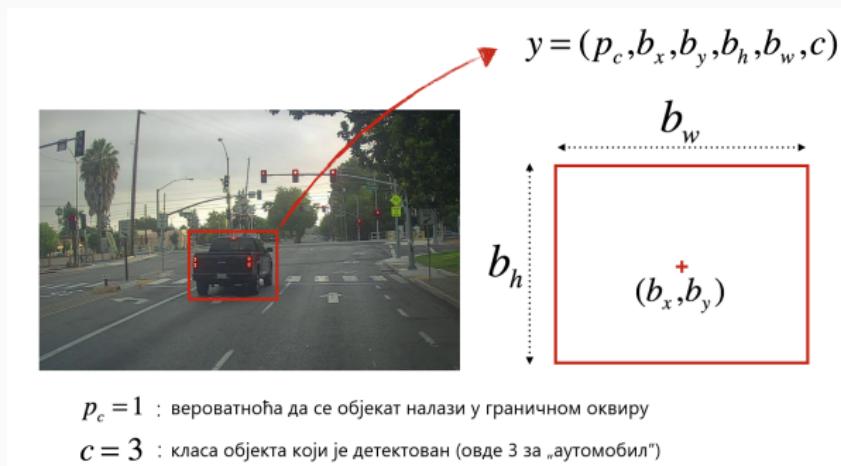


Конволуцијске неуронске мреже

- Две предности коришћења конволуцијских неуронских мрежа су:
 - Дељење параметара: филтер који детектује одређене ивице у једном делу слике, врло вероватно ће подједнако добро радити и у другом делу слике.
 - Ређе зависности (Sparsity of connections): У сваком слоју, ниједна излазна вредност не зависи од много улазних. Свако поље матрице на излазу је израчунато из малог исечка слике са улаза и не зависи од осталих.
- Конволуцијске неуронске мреже, овим одликама, омогућавају тренирање над мањим скуповима и теже долази до претренирања (overfitting).

YOLO алгоритам

- Класификација слике је одређивање класе објекта на слици, а локализација објекта је његово означавање
- Детекција објекта је проблем када се на слици налази више објекта које треба класификовати и локализовати
- Претпоставимо да детектујемо три класе: 1. пешак, 2. аутомобил, 3. мотоцикл



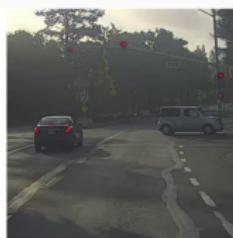
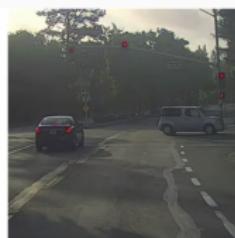
YOLO алгоритам

- Параметар c може да буде или целобројна вредност која представља класу или вектор димензија броја класа које детектујемо, где члан вектора који представља класу има вредност 1, а остали имају вредност нула.
- Ако нема објеката, вероватноћа p_c ће бити 0, па су остали чланови вектора небитни

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad (5)$$

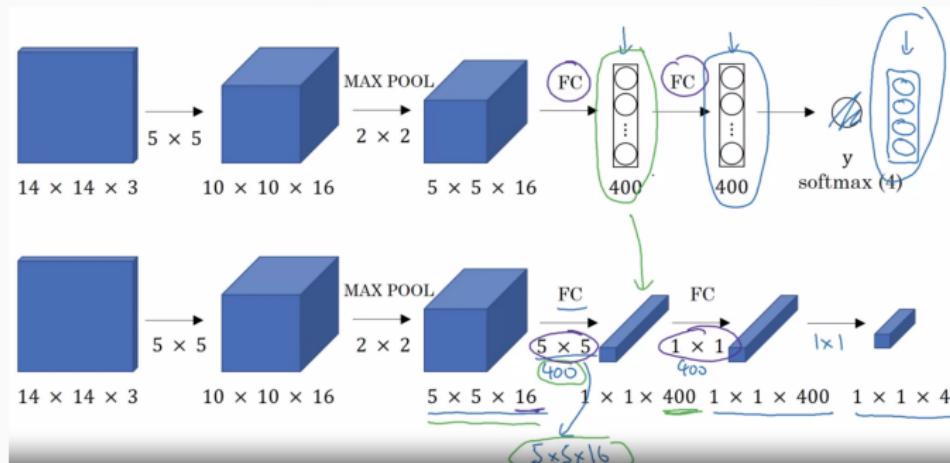
YOLO алгоритам

- Детекција објекта клизним прозорима
- Сваки прозор се проследи конволуцијској мрежи, која даје резултат да ли има објекта у прослеђеном исечку слике или не. Прозор треба да проклиза кроз целу слику и овај поступак се може поновити и са већим прозором
- Проблем оваквог приступа је превелико време извршавања



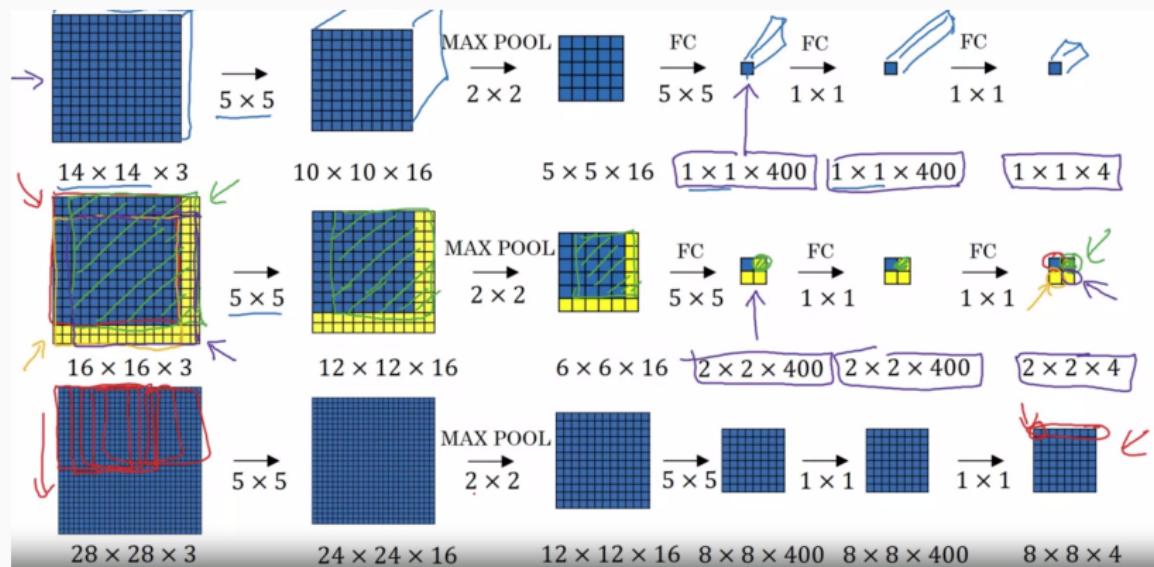
YOLO алгоритам

- Конволуцијска имплементација је ефикаснија
- Прва мрежа на крају садржи два потпуно повезана слоја (*fully connected layers*). Ова два слоја праве проблем тј. успоравају детекцију, јер је потребно посебно извршавање за сваки прозор тј. исечак слике.
- У другом случају се примењују филтери ...



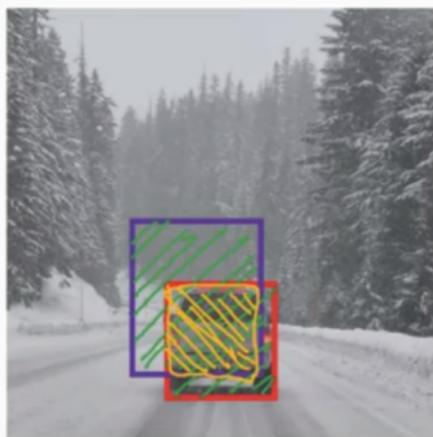
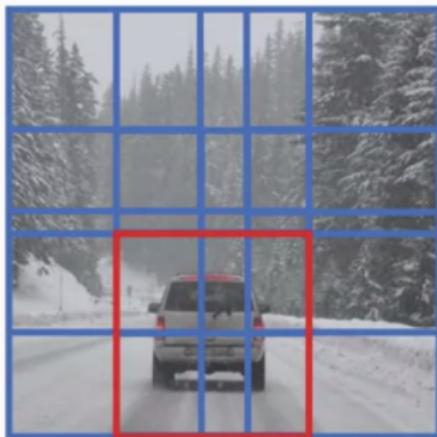
YOLO алгоритам

- Случај када се на улазу нађе **слика већих димензија**



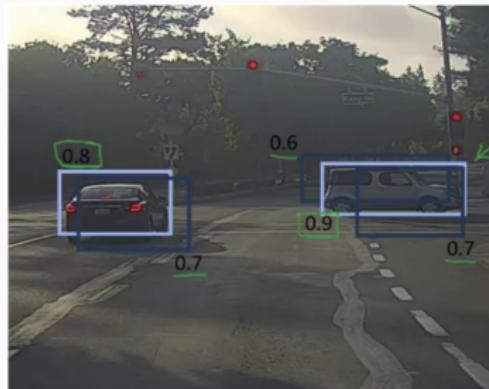
YOLO алгоритам

- Вероватноће да се унутар квадрата налазе тражени објекти се добијају **истовремено** за све исечке слике
- Потребно је изабрати најбољи гранични оквир, па се уводи **ПпУ - Пресек преко Уније** (**IoU - Intersection over Union**), дефинисан као $IoU = \frac{I}{U}$:



YOLO алгоритам

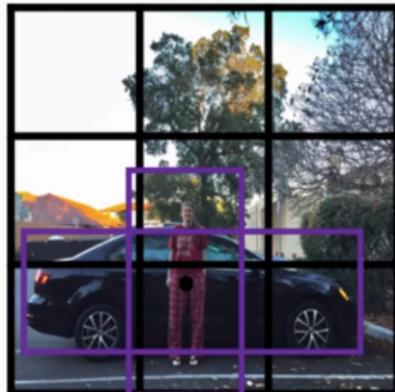
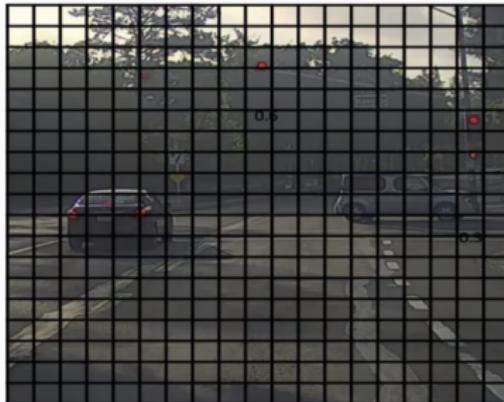
- Поступак у коме се **потискују** оквири који немају максималну вероватноћу детекције назива се **Не-Максимално потискивање (Non-Max suppression)**



- Одбацити све оквире где је вероватноћа $p_c \leq 0.6$
- Док год има непроверених оквира:
 - Изабрати оквир са највећом вероватноћом p_c
 - Одбацити све преостале оквире где је $IoU \geq 0.5$ у односу на оквир из претходног корака

YOLO алгоритам

- Слика се дели решетком димензија 19×19 , а објекат се додељује пољу у коме се налази центар објекта
- Могуће је да центри припадају истом пољу, па се уводе класни оквири (Anchor boxes)
- У случају детекције пешака и аутомобила, вектор детекције је дупло већи ($2 \times 8 = 16$)
- Објекат се додељује пару (пољеРешетке, класниОквир)



YOLO алгоритам

- YOLO (You Only Look Once) алгоритам на примеру детекције пешака и аутомобила:
 - За сваку ћелију решетке наћи 2 гранична оквира
 - Занемарити оквире са малом вероватноћом детекције
 - За сваку класу је потребно извршити Не-Максимално потискивање (Non-Max suppression)



Садржај

Увод

Припрема тренинг скупа

Добијање слика из видеа

Означавање знака и броја карте на слици

Прављење скупа за тренирање

Вештачке неуронске мреже - теорија

Увод у вештачке неуронске мреже

Конволуцијске неуронске мреже

YOLO алгоритам

Тренирање - Darknet програмски оквир

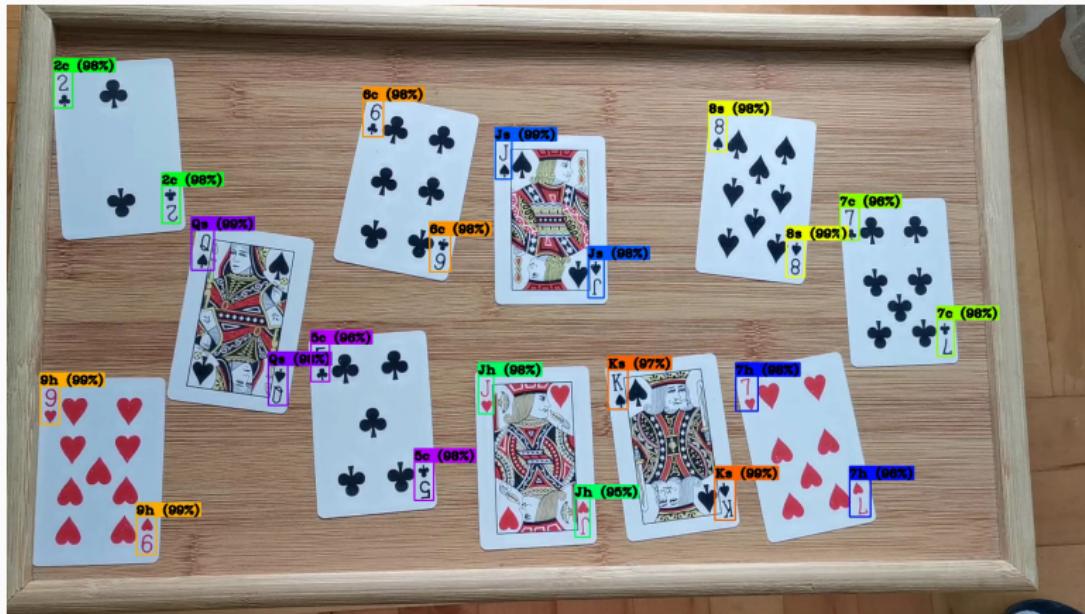
Закључак

Тренирање - Darknet програмски оквир

- Тренирање је извршено на **GoogleColaboratory** виртуелној машини
- Пребацивање слика из GoogleDrive-а у виртуелну машину
- У датотеци **yolov4-obj.cfg** се подешавају конфигурациони параметри:
 - **width=416; height=416; max_batches=104000; steps=83200,93600; filters=171**
- Тренирање почиње од иницијалних вредности тежина веза сачуваних у датотеци **yolov4.conv.137**

Тренирање - Darknet програмски оквир

- Параметар **-thresh 0.8** омогућава да се као резултат прикажу само оне детекције где је вероватноћа ≥ 0.8



Промене конфигурација током тренирања

- Тренинг скуп има 50000 слика, а валидациони 10000 (20%)
- До 3000. итерације је вршено тренирање са већом вредношћу параметра `lerning rate=0.001`
 - Мера тачности мреже над тренинг скупом (Mean Average Precision - mAP) је била око 97.5%
 - При тестирању над видеом детекције нису биле тачне
- После тренирања од још 400 итерација, за финије подешавање, са мањом вредношћу параметра `lerning rate=0.0001`
 - mAP је износио 100% над тренинг скупом
 - На 3400. итерацији резултати при тестирању су били врло добри
- `width` и `height` су постављени на 608 (ближе резолуцији камере)
 - Тренирање је завршено после извршених 5400 итерација

Садржај

Увод

Припрема тренинг скупа

Добијање слика из видеа

Означавање знака и броја карте на слици

Прављење скупа за тренирање

Вештачке неуронске мреже - теорија

Увод у вештачке неуронске мреже

Конволуцијске неуронске мреже

YOLO алгоритам

Тренирање - Darknet програмски оквир

Закључак

Закључак

- Програм потребно прилагодити физичким објектима
- Тренирање траје неколико сати, па резултат промене неких конфигурационих параметара није одмах видљив
- YOLO (You Only Look Once) извршава детекције у једној итерацији
- Главни аутор рада о YOLO алгоритму, Joseph Redmon се повукао са пројекта из етичких разлога
- Задивљујућа је чињеница да је сада рачунар способан да *види*
- Несумњиво је да ће, по речима ментора, развој Вештачке интелигенције обележити наше каријере

Литература I

-  *playing-card-detection*, приступљено (септембар 2020.) на
<https://github.com/geaxgx/playing-card-detection.git>
-  *Machine Learning for Beginners: An Introduction to Neural Networks*, приступљено (септембар 2020.) на
<https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9>
-  Andrew Ng, *Convolutional Neural Networks* [Интернет курс],
приступљено (септембар 2020.) на
<https://www.coursera.org/learn/convolutional-neural-networks/home/welcome>

Литература II

-  *Darknet: Open Source Neural Networks in C*, приступљено (септембар 2020.) на <https://pjreddie.com/darknet/>
-  *Running a YOLOv4 Object Detector with Darknet in the Cloud! (GPU ENABLED)*, приступљено (септембар 2020.) на https://colab.research.google.com/drive/1_GdoqCJWXsChr0iY8sZMr_zbr_fH-0Fg?usp=sharing
-  *Playing card detection with YOLO*, приступљено (септембар 2020.) на <https://youtu.be/pnntrewH0xg>
-  *Diplomski rad - YOLO detekcija*, приступљено (септембар 2020.) на <https://github.com/aleksavelickovic5762015/Diplomski-rad---YOLO-detekcija.git>