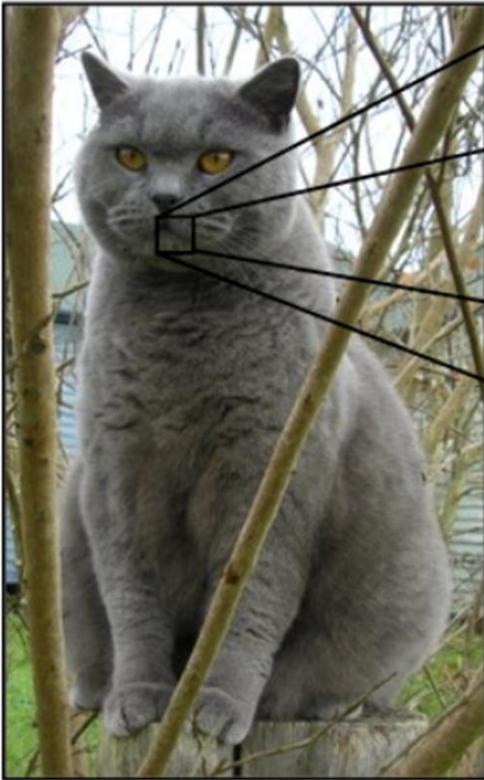


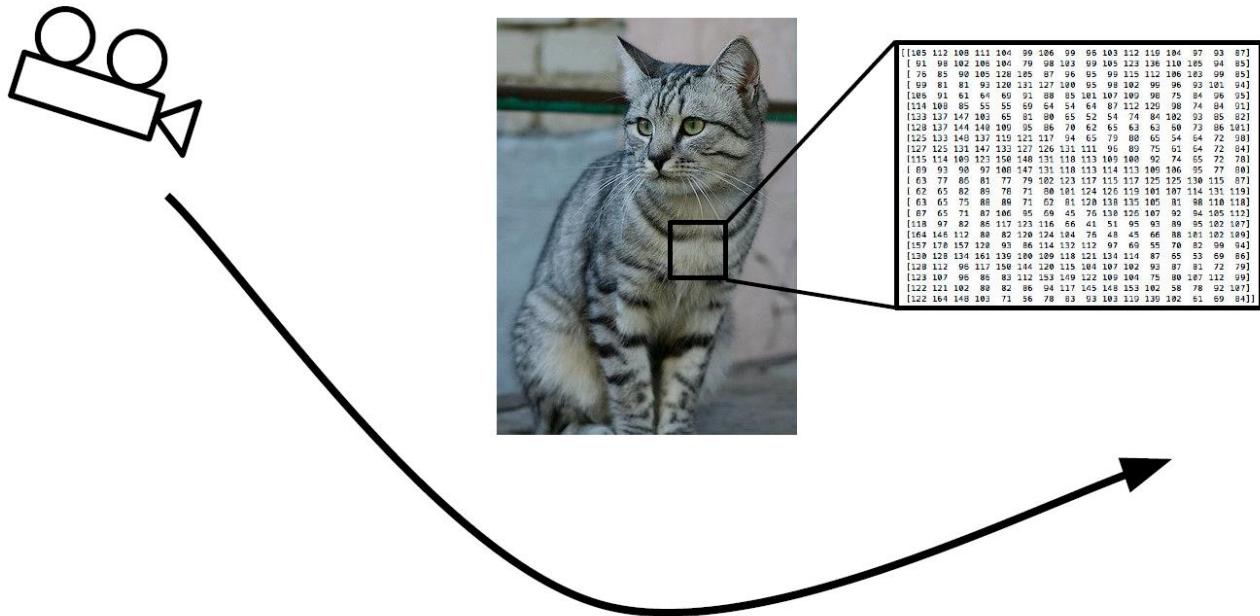
Klasifikacija slika

Klasifikacija slika

- Ulaz: slika
 - Izlaz: šta slika predstavlja
 - Labela („mačka“, „pas“, „sto“,...)
 - Ovo je problem klasifikacije
 - Moramo obučiti algoritam da raspoznaće različite klase
 - Koristimo mašinsko učenje
 - Npr. treniraćemo model na hiljadama slika koje sadrže mačke i hiljadama slika koje ne sadrže mačke
 - Algoritam će prepoznavati samo klase koje je naučio da prepoznaće
- 
- | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 58 | 32 | 22 | 97 | 38 | 15 | 00 | 40 | 00 | 75 | 04 | 05 | 07 | 78 | 52 | 32 | 50 | 77 | 91 | 00 |
| 49 | 49 | 99 | 40 | 37 | 81 | 18 | 57 | 60 | 87 | 17 | 40 | 98 | 43 | 69 | 41 | 04 | 56 | 62 | 00 |
| 81 | 49 | 31 | 73 | 55 | 79 | 14 | 29 | 93 | 71 | 40 | 67 | 51 | 18 | 30 | 03 | 49 | 13 | 36 | 65 |
| 52 | 70 | 95 | 23 | 04 | 60 | 11 | 42 | 60 | 14 | 68 | 56 | 01 | 32 | 56 | 71 | 37 | 02 | 36 | 91 |
| 22 | 31 | 16 | 71 | 51 | 67 | 09 | 89 | 41 | 92 | 36 | 54 | 22 | 40 | 40 | 28 | 66 | 33 | 13 | 80 |
| 24 | 47 | 34 | 60 | 99 | 03 | 45 | 02 | 44 | 75 | 33 | 53 | 78 | 36 | 64 | 20 | 35 | 17 | 12 | 50 |
| 52 | 98 | 81 | 28 | 64 | 23 | 67 | 10 | 26 | 38 | 40 | 67 | 59 | 54 | 70 | 66 | 18 | 38 | 64 | 70 |
| 67 | 26 | 20 | 68 | 02 | 62 | 12 | 20 | 95 | 63 | 94 | 39 | 63 | 08 | 40 | 91 | 66 | 49 | 94 | 21 |
| 24 | 55 | 58 | 05 | 66 | 73 | 99 | 26 | 97 | 17 | 78 | 78 | 96 | 83 | 14 | 88 | 34 | 89 | 63 | 72 |
| 21 | 36 | 23 | 09 | 75 | 00 | 76 | 44 | 20 | 45 | 35 | 14 | 00 | 61 | 33 | 97 | 34 | 31 | 33 | 95 |
| 78 | 17 | 53 | 28 | 22 | 75 | 31 | 67 | 15 | 94 | 03 | 80 | 04 | 62 | 16 | 14 | 09 | 53 | 56 | 92 |
| 16 | 39 | 05 | 42 | 96 | 35 | 31 | 47 | 55 | 58 | 88 | 24 | 00 | 17 | 54 | 24 | 36 | 29 | 85 | 57 |
| 86 | 56 | 00 | 48 | 35 | 71 | 89 | 07 | 05 | 44 | 44 | 37 | 41 | 60 | 21 | 58 | 51 | 54 | 17 | 58 |
| 19 | 80 | 81 | 68 | 05 | 94 | 47 | 69 | 28 | 73 | 92 | 13 | 86 | 52 | 17 | 77 | 04 | 89 | 55 | 40 |
| 04 | 52 | 08 | 83 | 97 | 35 | 99 | 16 | 07 | 97 | 57 | 32 | 16 | 26 | 79 | 33 | 27 | 98 | 66 | |
| 09 | 46 | 68 | 87 | 57 | 62 | 20 | 72 | 03 | 46 | 33 | 67 | 46 | 55 | 12 | 32 | 63 | 93 | 53 | 69 |
| 04 | 42 | 16 | 73 | 38 | 97 | 39 | 11 | 24 | 94 | 72 | 18 | 08 | 46 | 29 | 32 | 40 | 62 | 76 | 36 |
| 20 | 69 | 36 | 41 | 72 | 30 | 23 | 88 | 31 | 05 | 89 | 69 | 82 | 67 | 59 | 85 | 74 | 04 | 36 | 16 |
| 20 | 73 | 35 | 29 | 78 | 31 | 90 | 01 | 74 | 31 | 49 | 71 | 48 | 55 | 81 | 16 | 23 | 57 | 05 | 54 |
| 01 | 70 | 54 | 71 | 83 | 51 | 54 | 69 | 16 | 92 | 33 | 48 | 61 | 43 | 52 | 01 | 89 | 11 | 67 | 48 |
- Šta kompjuter vidi
- semantic gap*
- Klasifikacija slike
- 82% cat
15% dog
2% hat
1% mug

Izazovi: varijacija u položaju kamere

Challenges: Viewpoint variation



This image by Nikita is
licensed under CC-BY 2.0

Izazovi: osvetljenje

Challenges: Illumination



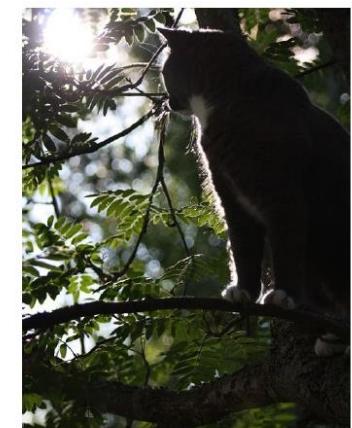
[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)



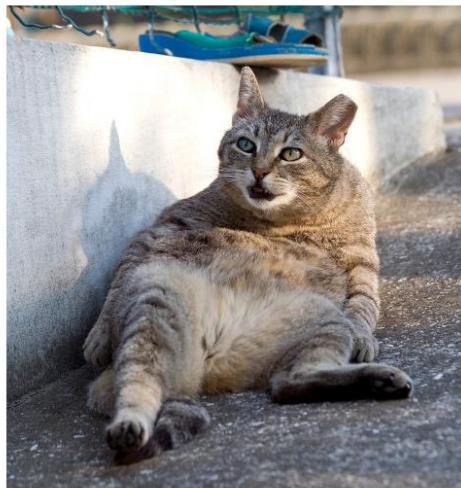
[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

Izazovi: deformacija objekta

Challenges: Deformation



This image by Umberto Salvagnin
is licensed under CC-BY 2.0



This image by Umberto Salvagnin
is licensed under CC-BY 2.0



This image by sare bear is
licensed under CC-BY 2.0



This image by Tom Thai is
licensed under CC-BY 2.0

Izazovi: zaklonjenost objekta

Challenges: Occlusion



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)



This image by [jonsson](#) is licensed under [CC-BY 2.0](#)

Izazovi: pozadina

Challenges: Background Clutter



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

Izazovi: varijacija u klasi

Challenges: Intraclass variation



This image is CC0 1.0 public domain

Klasifikacija slika

An image classifier

```
def classify_image(image):  
    # Some magic here?  
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for
recognizing a cat, or other classes.

Manuelno kodirana pravila

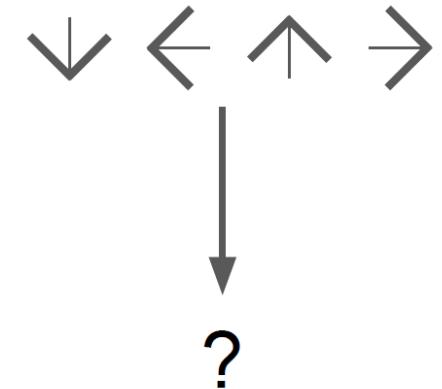
Attempts have been made



Find edges



Find corners



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

Pristup zasnovan na učenju iz podataka

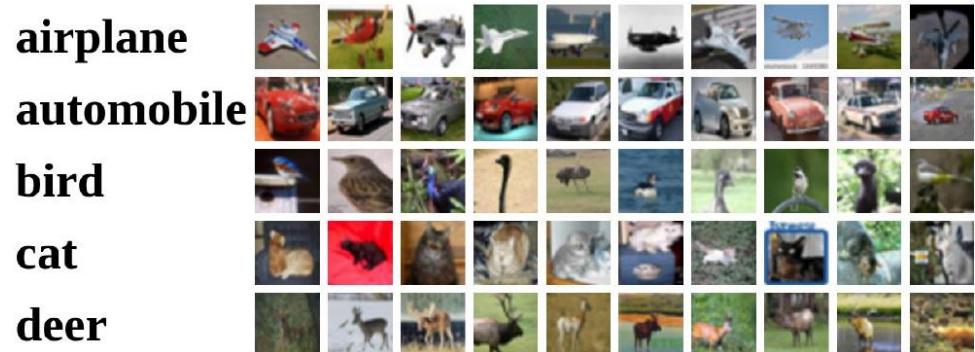
Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

Example training set

```
def train(images, labels):
    # Machine learning!
    return model

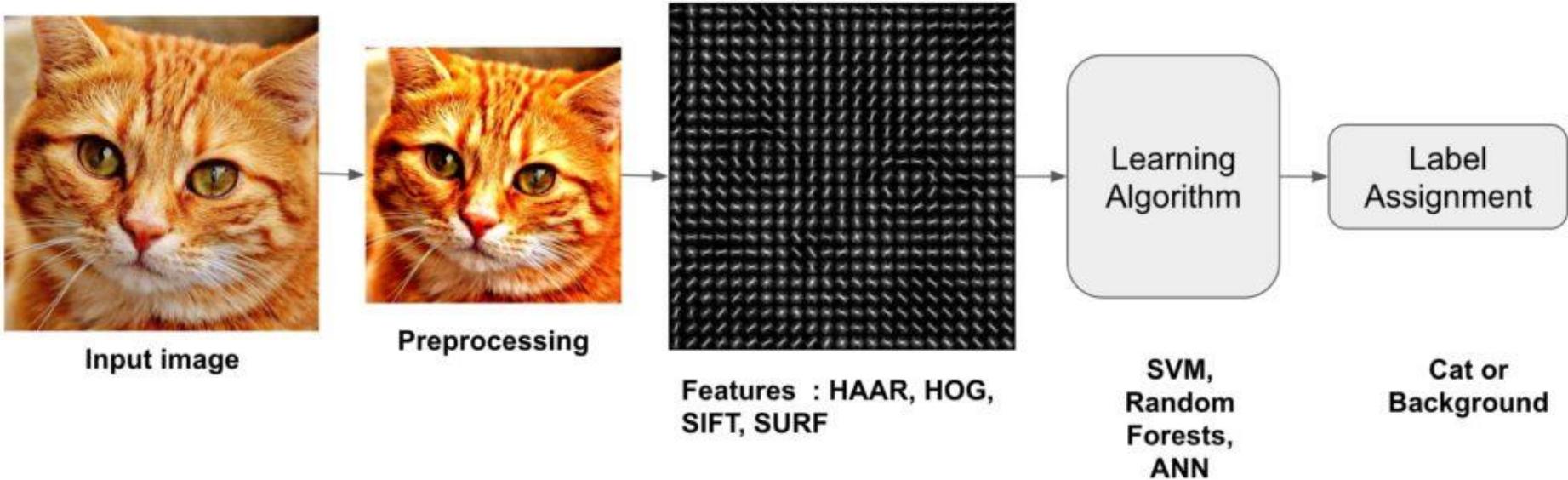
def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```



Istorija i trenutno stanje

- 2001. Paul Viola and Michael Jones
 - Detekcija lica na video snimku u realnom vremenu
 - Njihovo rešenje je implementirano u OpenCV i Viola-Jones je postao sinonim za detekciju lica
- 2005. Navneet Dalal i Bill Triggs
 - HOG (Histogram of Oriented Gradients) deskriptor
 - Pokazao se značajno bolji u odnosu na alternativne metode u detekciji pešaka
- Sada – *Deep Learning*
 - 2012. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) – značajno bolje performanse u odnosu na ostale algoritme (85%, za 11% bolji od drugog mesta)
 - 2013. Svi pobednici koriste *Deep Learning*
 - 2015. Konvolucione neuronske mreže (*Convolutional Neural Networks*) prevazilaze performanse čoveka (95%)

Konceptualni pregled sistema



- Tradicionalni *computer vision* pristupi prate ovaj *pipeline*
- *Deep Learning* preskače korak ekstrakcije obeležja

1. Preprocesiranje

- Česti postupci preprocesiranja:
 - Normalizacija kontrasta i efekata osvetljenja
 - Normalizacija podataka (oduzeti *mean* i podeliti standardnom devijacijom)
 - Nekada *gamma correction* daje nešto bolje rezultate
 - Kada radimo sa slikama u boji, transformacija reprezentacije (npr. RGB u LAB) može da daje bolje rezultate
 - ...
- Nije unapred jasno koji koraci preprocesiranja su dobri
 - Morate da isprobate šta radi na vašim podacima
- Važan deo preprocesiranja jeste da se slika svede na fiksnu veličinu (isecanje ili skaliranje)
 - Ovo je esencijalno – sledeći korak (ekstrakcija obeležja) se izvršava na slikama fiksne veličine

2. Ekstrakcija obeležja

- Ulazna slika ima mnoge nerelavantne informacije sa aspekta klasifikacije – moramo je pojednostaviti izvlačenjem važnih (relevantnih) informacija
- Npr. na slici želimo da detektujemo novčiće



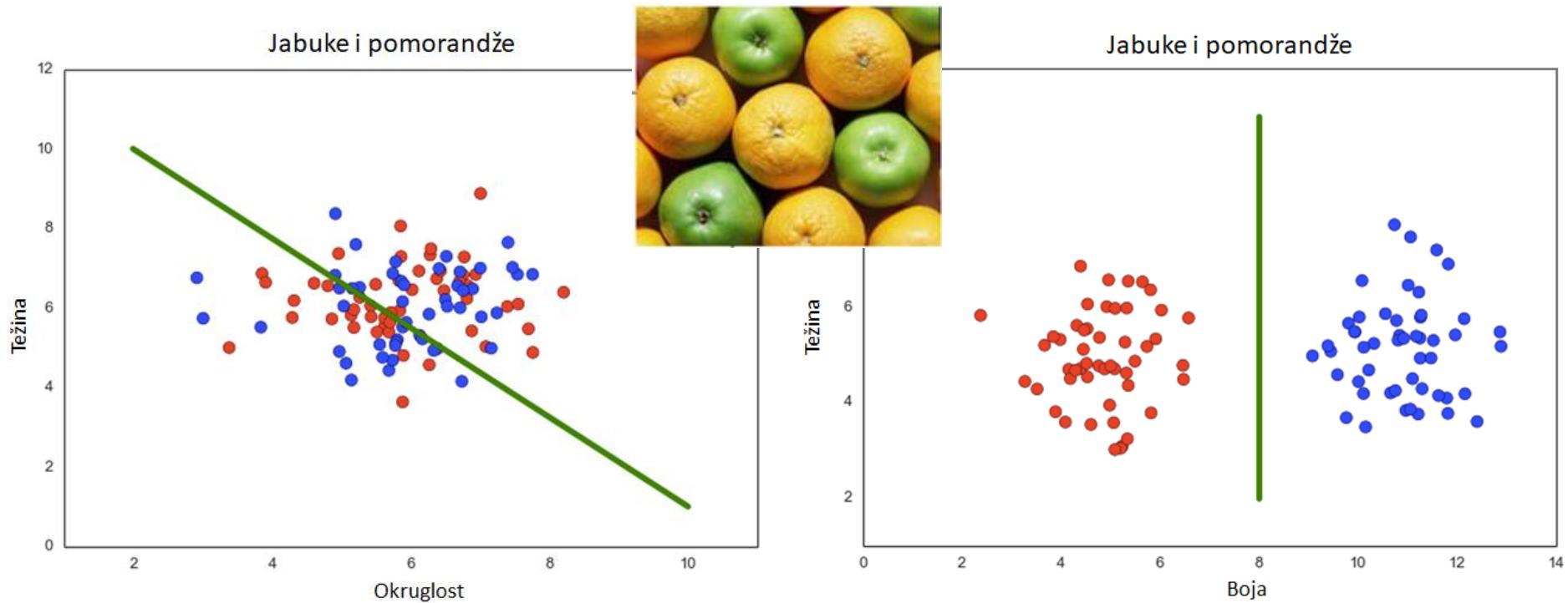
Velike varijacije u RGB vrednostima piksela

Slika je pojednostavljena, a i dalje vidimo kružne oblike koje predstavljaju novčiće.

Nadamo se da smo zadržali ključne informacije neophodne za razlikovanje novčića od ostalih objekata, a odbacili šum

2. Ekstrakcija obeležja

- U tradicionalnim pristupima ovaj korak je od ključne važnosti za performanse algoritma
 - Potrebna su nam pouzdana obeležja
 - Npr. u primeru sa novčićima, dobar postupak ekstrakcije obeležja će ne samo uhvatiti kružni oblik koji predstavlja novčić, već i informacije koje nam omogućavaju da novčić razlikujemo od drugih kružnih objekata (dugmad, točkovi,...)



2. Ekstrakcija obeležja

- Poznate metode za ekstrakciju obeležja:
 - Haar-like features (Viola-Jones)
 - Histogram of Oriented Gradients (HOG)
 - Scale-invariant Feature Transform (SIFT)
 - Speeded Up Robust Feature (SURF)

3. Obučavajući algoritam za klasifikaciju

- Potrebno je da obučimo algoritam za klasifikaciju
 - Biće nam potrebni pozitivni primeri (slike koje sadrže objekat od interesa, npr. mačku) i negativni primeri (slike koje ne sadrže objekat od interesa)
- Postoje različiti algoritmi za klasifikaciju
 - Generalan princip je da se vektori obeležja tretiraju kao tačke u visokodimenzionom prostoru
 - U tom prostoru pokušavamo da pronađemo ravni/površine koje će dobro razdvojiti jednu klasu od druge
 - Ovo ćemo lakše razumeti na konkretnom algoritmu, npr. SVM

Model k najbližih suseda (k-NN)

Distance Metric to compare images

L1 distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

- = add → 456

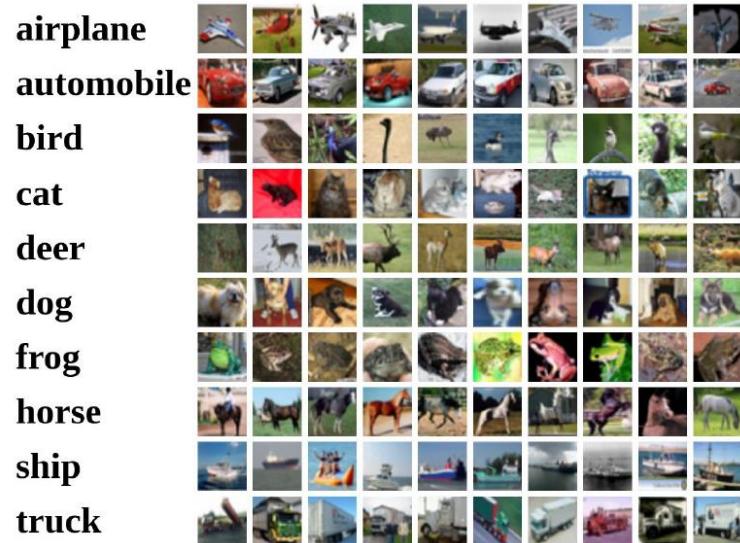
Model k najbližih suseda (k-NN)

Example Dataset: CIFAR10

10 classes

50,000 training images

10,000 testing images



Test images and nearest neighbors



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Model k najbližih suseda (k -NN)

- N primera u trening skupu:
 - Vreme treniranja: $O(1)$
 - Vreme testiranja: $O(N)$
 - Ovo je loše – voleli bismo da naš klasifikator bude *brz* prilikom predikcije. Sporo treniranje je u redu
- <http://vision.stanford.edu/teaching/cs231n-demos/knn/>
- Hiper-parametri:
 - Metrika rastojanja
 - k

Model k najbližih suseda (k-NN)

- Može se primeniti na bilo koji tip podataka
 - Samo treba da definišemo meru udaljenosti
 - Generalno, ovaj algoritam je dobra početna tačka kada razmatramo novi problem
- Međutim, retko ga koristimo u praksi
 - Spor u trenutku testiranja
 - Nezgodan za primenu na slike – distance definisane nad pikselima ne moraju da budu informativne

Original



Boxed



Shifted



Tinted

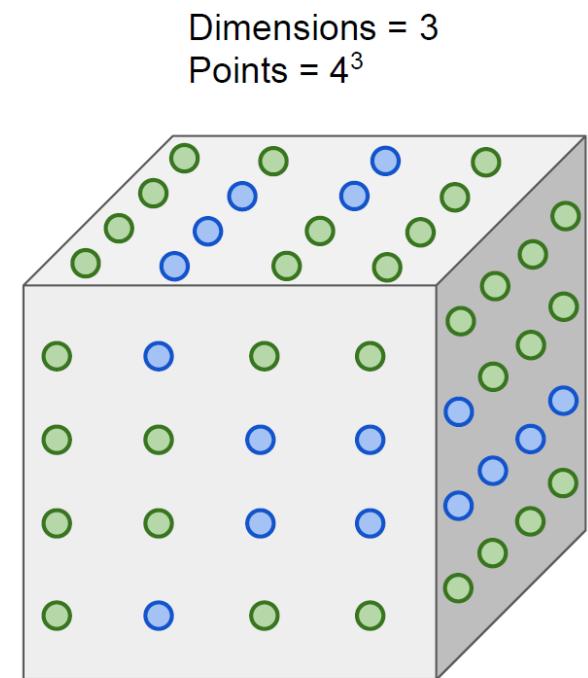
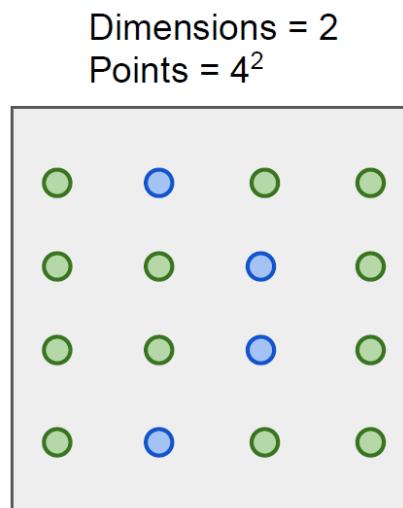
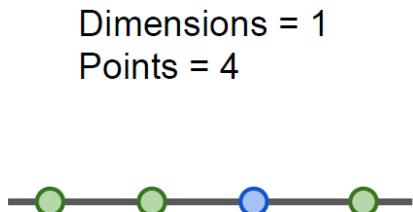


(all 3 images have same L2 distance to the one on the left)

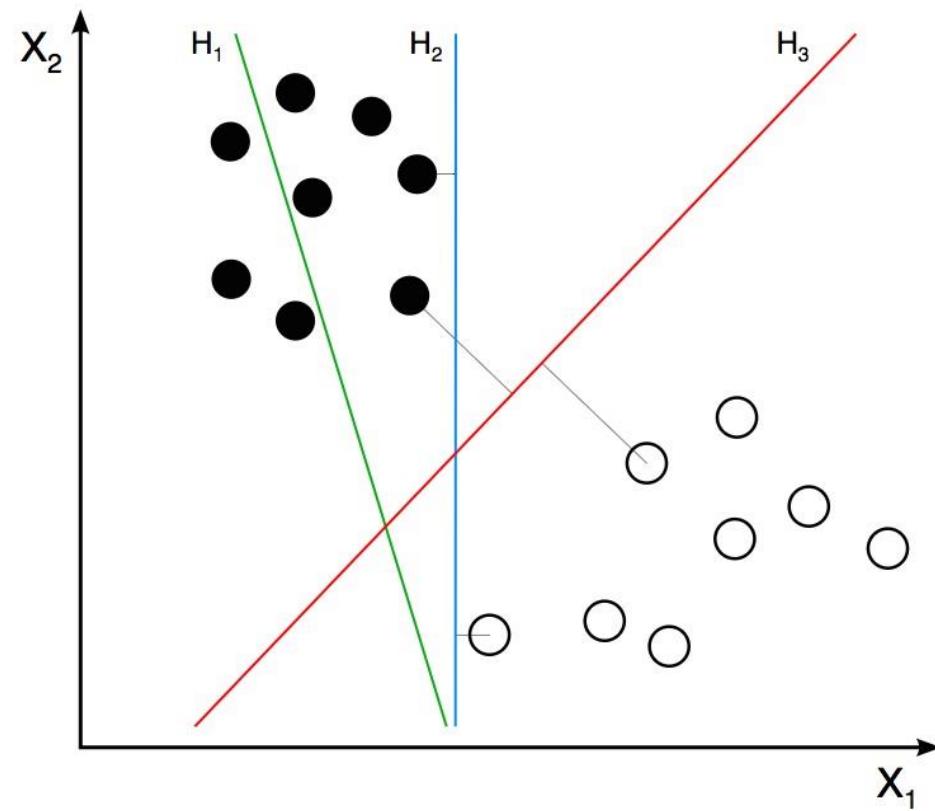
Model k najbližih suseda (k -NN)

k-Nearest Neighbor on images **never used.**

- Curse of dimensionality

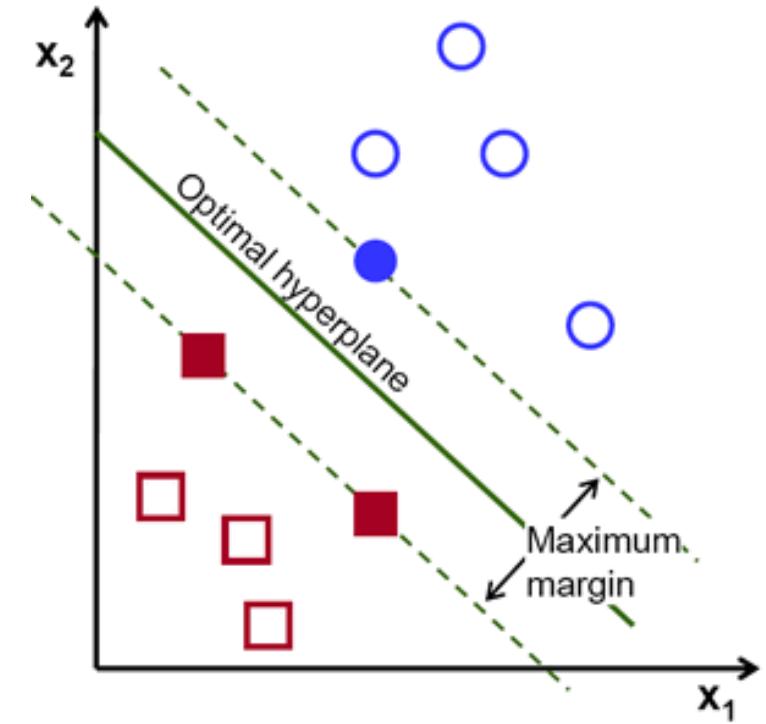
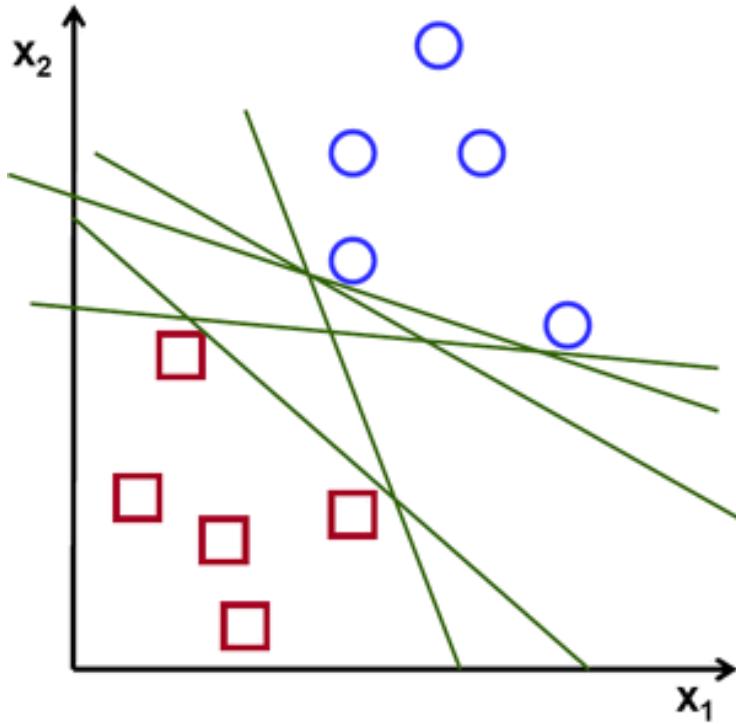


SVM (Support Vector Machines)



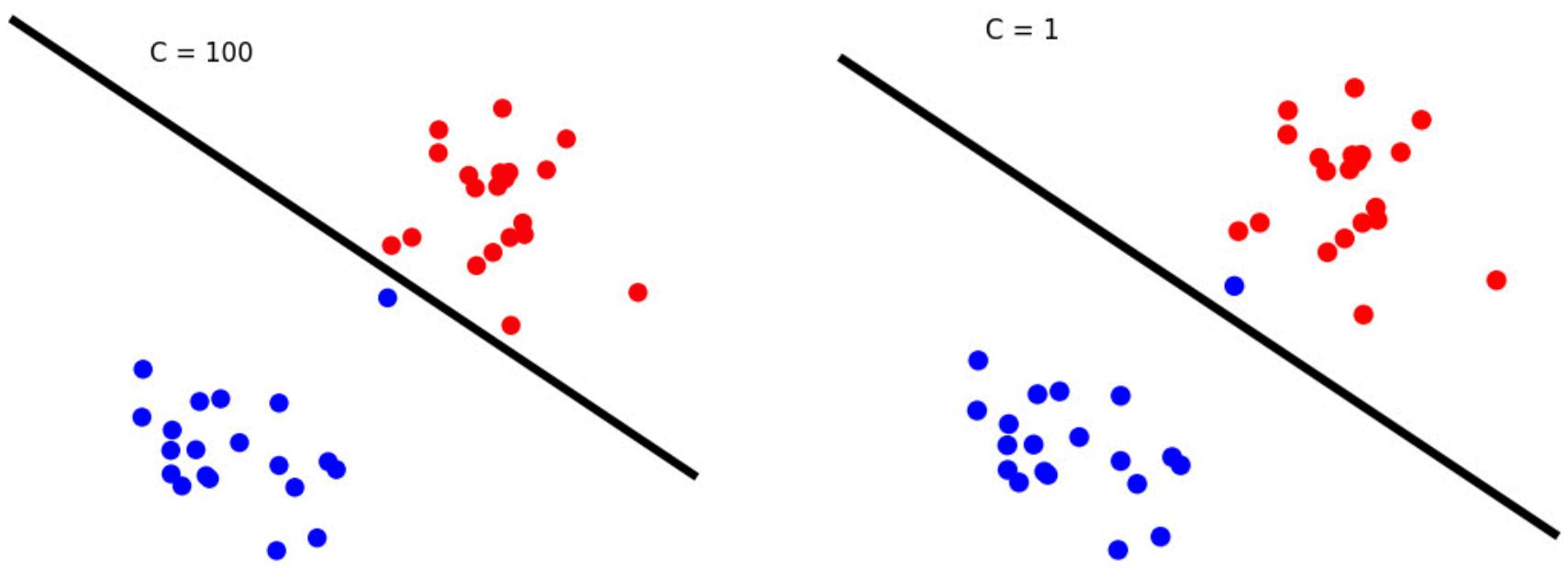
- Pojednostavljena predstava:
 - Vektor obeležja sastoji od samo dve vrednosti: x_1 i x_2
 - Imamo dve klase (crna i bela) i tražimo liniju koja najbolje razdvaja ove dve klase u (x_1, x_2) prostoru
- Linija H_1 - ne razdvaja klase dobro
 - Neke „crne“ primere svrstava u „belu“ klasu
- Linije H_2 i H_3 obe perfektno razdvajaju klase. Koja je bolja i zašto?

SVM: Pronalaženje maksimalne margine



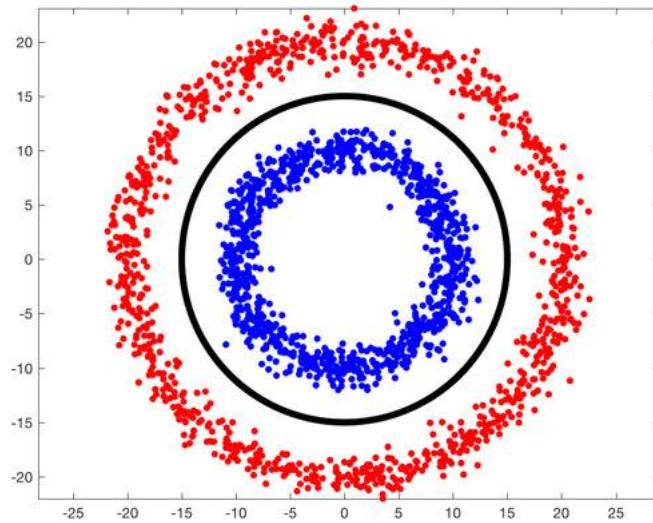
- Margina razdvajajuće hiperravni je minimum rastojanja od te hiperravni do neke od tačaka skupa podataka
- Među svim razdvajajućim hiperravnima (slika levo), SVM pronalazi optimalnu – onu sa **najvećom marginom** (slika desno)

SVM hiper-parametri

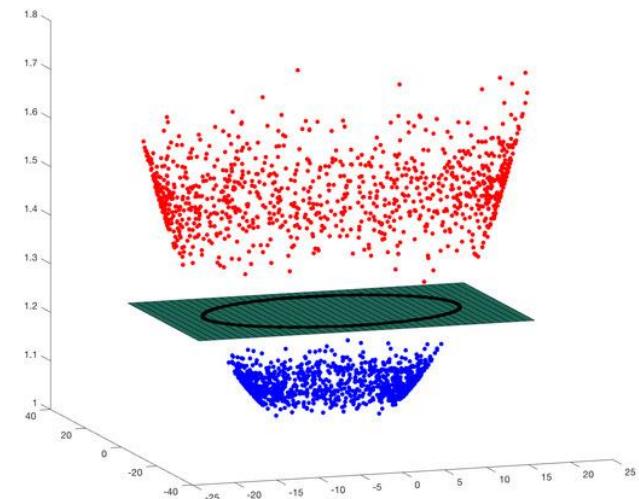


- Koja margina je bolja?
- Parametar C kontoliše nagodbu između veličine margine i broja grešaka:
 - Malo C : velika margina, ali dozvoljavamo da dosta primera bude pogrešno klasifikovano
 - Veliko C : mala margina, ali mali broj primera je pogrešno klasifikovano

SVM hiper-parametri



Šta ako podaci nisu **linearno separabilni** (željena granica odluke nije hiperravan)?

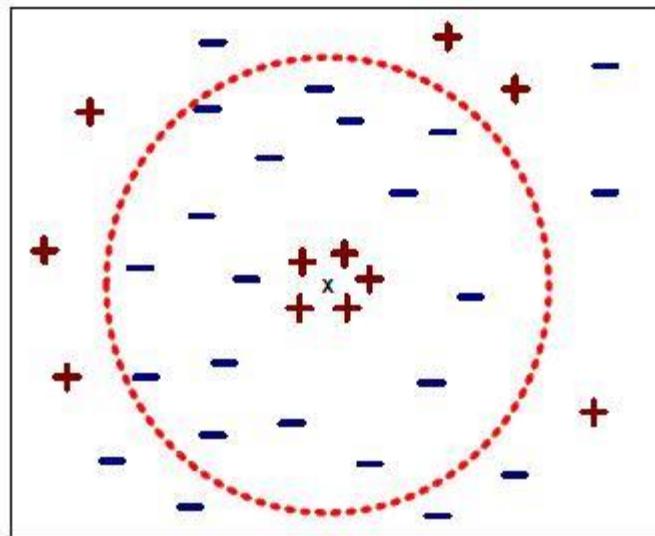


Transformisamo ih u prostor gde jesu i primeniti SVM

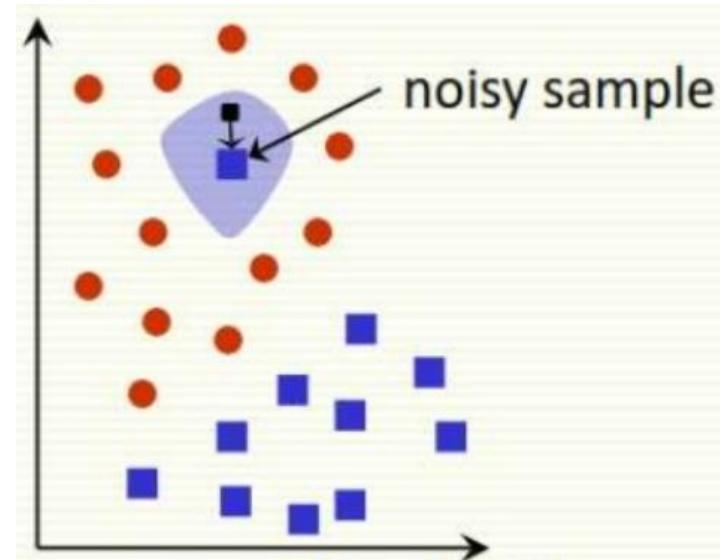
- Ovo možemo postići korišćenjem RBF kernela i podešavanjem njegovog parametra γ
- SVM ima još mnogo hiper-parametara (različite opcije za kernele gde svaki kernel ima svoje parametre)
 - Više o ovome na predmetu Mašinsko učenje

Kako optimizovati hiper-parametre modela?

- Gotovo svaki model ima parametre koje treba optimizovati
 - Npr. kod K -NN algoritma treba da odredimo optimalno K (broj najbližih suseda koje uzimamo u obzir prilikom klasifikacije primera)
 - Kod SVM algoritma treba da odaberemo C i γ
- Optimizacija je od izuzetne važnosti za dobre performanse



Preveliko K



Premalo K – osetljivost
na šum (outliere)

Kako optimizovati hiper-parametre modela?

- Hiper-parametri
 - Ne učimo ih direktno iz podataka, već ih unapred zadajemo
 - Problem: veoma zavise od problema
- Najčešći pristup: isprobati šta radi najbolje
 - Za svaku kombinaciju vrednosti parametra obučiti model i izmeriti njegove performanse. Odabratи one parametre koji rezultuju najboljim performansama
 - Npr. treniramo K -NN model za $K = 1, 2, \dots$ i odaberemo K koje daje najbolje performanse
 - Za SVM bismo isprobavali različite kombinacije vrednosti C i γ
- Ali na kojim podacima trenirati model a na kojim meriti performanse modela?

Optimizacija hiper-parametara

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: $K = 1$ always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train

test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train

validation

test

Optimizacija hiper-parametara

Setting Hyperparameters

Your Dataset

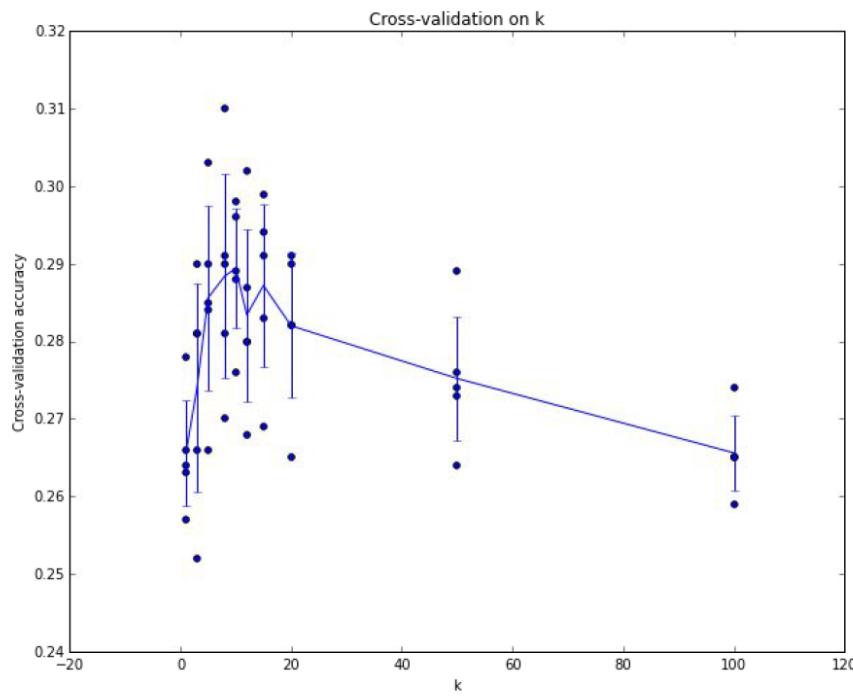
Idea #4: Cross-Validation: Split data into **folds**,
try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Optimizacija hiper-parametara

Setting Hyperparameters



Example of
5-fold cross-validation
for the value of **k**.

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

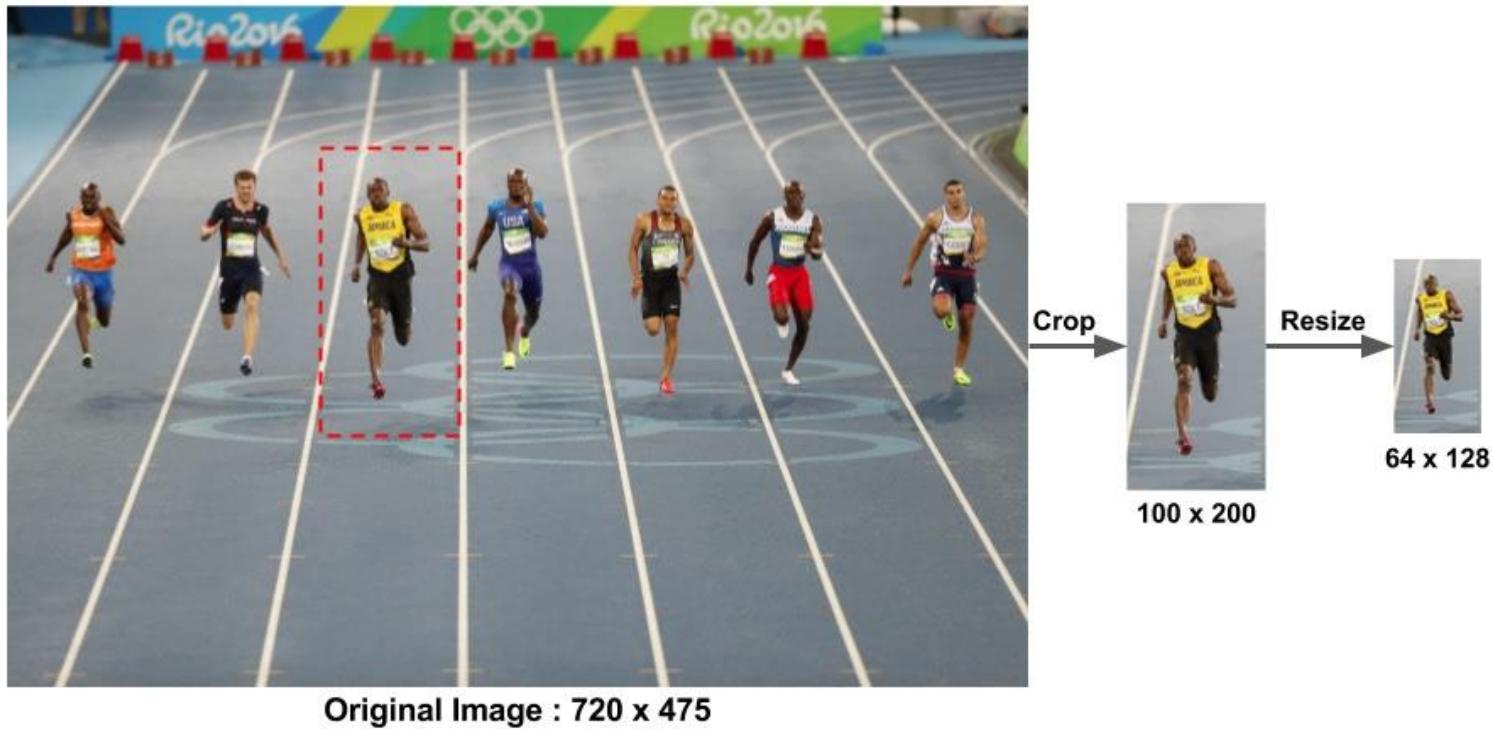
(Seems that $k \approx 7$ works best
for this data)

Ekstrakcija obeležja – HOG deskriptor

HOG (Histogram of Oriented Gradients)

- Slika (fiksne veličine) se transformiše u vektor obeležja (fiksne veličine)
 - Tipično, slika veličine širina \times visina \times 3 se transformiše u vektor dužine n
 - Na primer, videćemo kako pomoću HOG deskriptora sliku veličine $64 \times 128 \times 3 = 24\,576$ vrednosti možemo reprezentovati pomoću samo 3780 vrednosti, a da ta reprezentacija bude korisna
 - Šta znači „korisno“ zavisi od primene – mi želimo obeležja koja su nam korisna za razlikovanje objekata na slikama
- Pogled iz teorije informacija (*information theory*): ivice kodiraju promenu, a promene su teške za predviđanje. Zato, ivice efikasno kodiraju sliku
- Tamo gde se nalazi ivica, imaćemo veliku magnitudu gradijenta intenziteta piksela
- Ideja HOG: lokalni izgled objekta se može efektivno opisati distribucijom (histogram) smera ivica (orientacija gradijenta)

Preprocesiranje



- Potrebni su nam isečci slike fiksne veličine

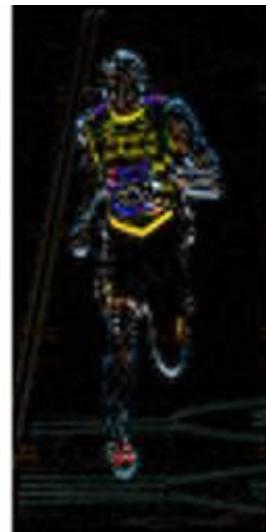
HOG postupak

1. Računanje gradijenta

- Izračunati gradijent po obe ose g_x i g_y (na jednom od prethodnih predavanja smo videli različite kernele za računanje gradijenta, npr. Sobel)
- Izračunati magnitudu $\sqrt{g_x^2 + g_y^2}$ gradijenta i orientaciju $\theta = \arctan \frac{g_y}{g_x}$ (između 0 i 180°)



Gradijent po x osi
(vertikalne ivice)



Gradijent po y osi
(horizontalne ivice)



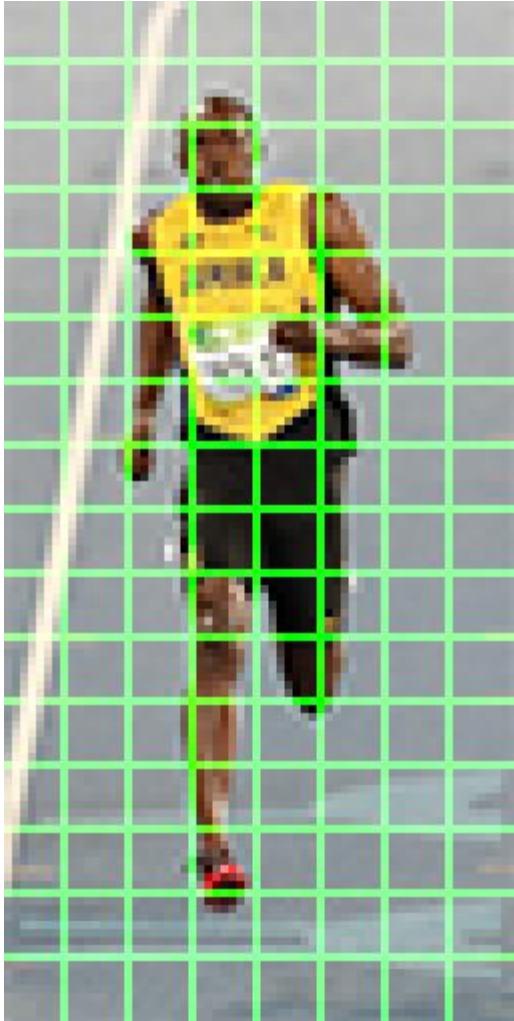
Magnituda
gradijenta

HOG postupak

- Kod slika u boji gradijent se računa za svaki od tri kanala
- Za magnitudu gradijenta piksela se uzima maksimalna magnituda (od tri kanala) i ugao koji odgovara tom gradijentu

HOG postupak

2. Slika se deli na ćelije veličine $N \times N$

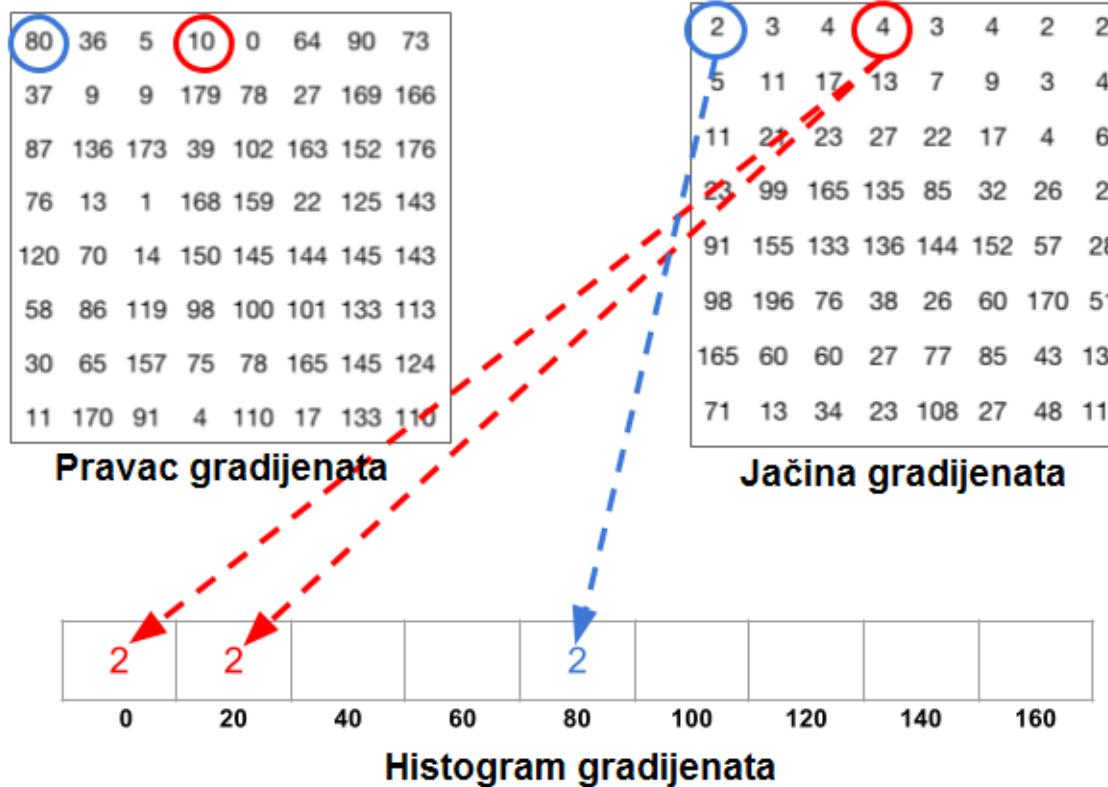


- Za svaku ćeliju ćemo izračunati histogram gradijenata
- Računanje histograma na ćelijama čini reprezentaciju robustnijom na šum
 - Individualni gradijenti sadrže šum, ali histogram preko ćelije $N \times N$ čini reprezentaciju manje osetljivom na šum

HOG postupak

3. Izračunati histogram gradijenata u ćelijama $N \times N$

- Za svaki piksel ćelije računamo magnitudu i gradijent (npr. za veličinu ćelije 8×8 imamo 64 magnituda i 64 gradijenata – 128 brojeva)
- Kompaktnija reprezentacija ovih brojeva bi bilo histogram



- Napravićemo histogram sa 9 podeoka (*bins*) koji odgovaraju orientacijama $0^\circ, 20^\circ, \dots, 160^\circ$
- Svaki piksel “glasa” za podok koji odgovara njegovoj orientaciji gradijenta. Jačina glasa odgovara magnitudi gradijenta

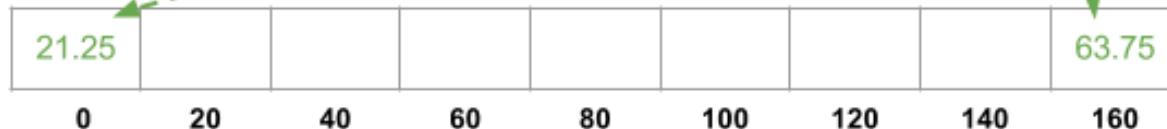
HOG postupak

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

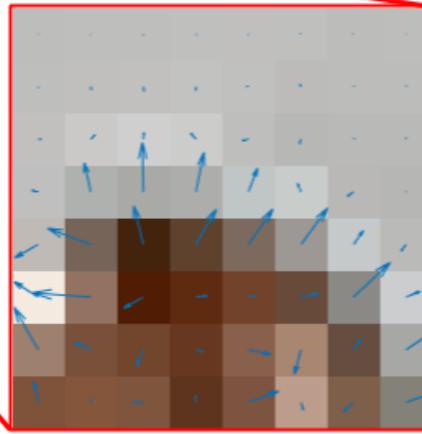
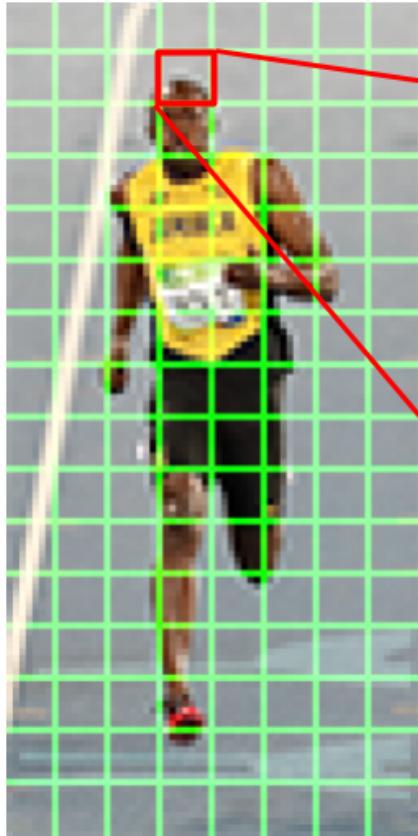
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude



Histogram of Gradients

HOG postupak



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

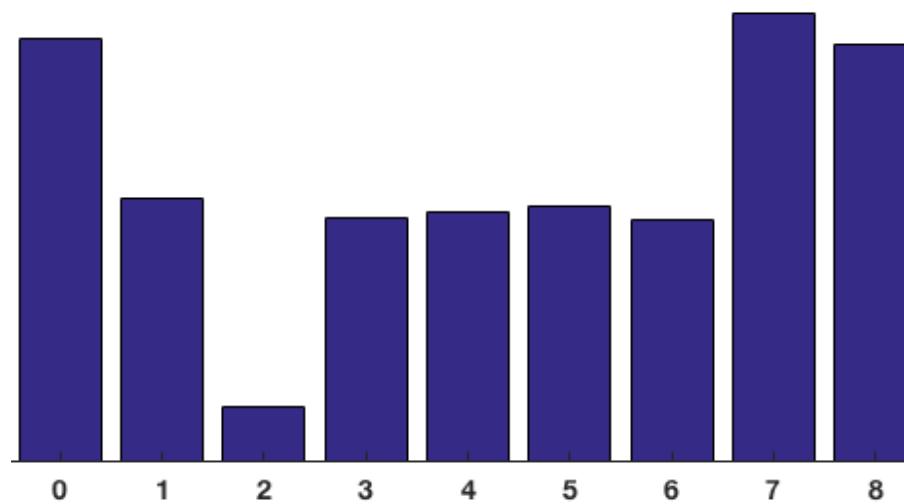
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

- Koriste se *unsigned gradients* – ugao se kreće između 0 i 180 stepeni
- U radu koji predstavlja HOG je empirijski pokazano da to radi bolje za detekciju pešaka
 - Neke biblioteke dozvoljavaju da specifikirate da li računate *signed* ili *unsigned*

HOG postupak



- Rezultujući histogram
 - Vidimo da ima dosta težina blizu 0 i 180 stepeni – to je drugi način da kažemo da ima dosta ivica koje ukazuju dole ili gore
- Dobićemo kompaktiju reprezentaciju
 - Ako su ćelije veličine 8×8 , svaka sadrži $8 \times 8 \times 3 = 192$ vrednosti
 - Gradijent sadrži $8 \times 8 \times 2 = 128$ vrednosti
 - Histogramom ovih 128 vrednosti redukujemo na 9

HOG postupak

- Kako odabratи broj segmenata (*bins*)?
 - Autori HOG deskriptora su predložili 9 za hvatanje gradijenata od 0° do 180° u inkrementima od 20°
 - Možete probati da eksperimentišete sa većom granulacijom, kao i izborom da li ćete koristiti *signed* ili *unsigned* gradijente

HOG postupak

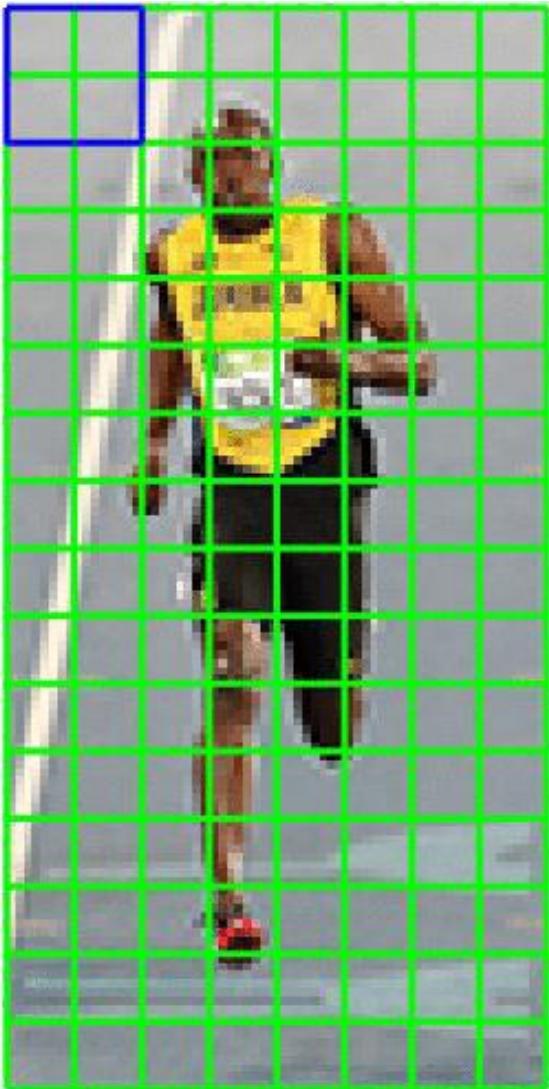
- Kako odrediti veličinu ćelije?
 - Izbor zavisi od veličine obeležja koja tražimo
 - U originalnom radu koji je predstavio HOG detektovani su pešaci i korišćene su ćelije 8×8 . Ćelije te veličine na slikama pešaka skalirane na 64×128 su dovoljno velike da „uhvate“ interesantna obeležja poput lica, vrha glave,...
 - Previše male ćelije će rezultovati ne toliko kompaktnom reprezentacijom
 - Previše velike ćelije mogu da ne obuhvate relevantne informacije

HOG postupak

4. Normalizacija bloka

- Konstruisan histogram je baziran na gradijentu slike
- Gradijent je osetljiv na osvetljenje – ako napravimo sliku tamnijom tako što prepolovimo intenzitet svakog piksela, magnituda gradijenta će se takođe prepoloviti
- Ovo čini naš postupak osetljiv na osvetljenje, a, idealno, želimo da naš deskriptor bude invarijantan na varijacije u osvetljenju
- Da bismo se suprotstavili ovim efektima, normalizujemo histogram
 - Razmišljajte o histogramu kao o vektoru vrednosti
 - Svaku vrednost ćemo podeliti magnitudom vektora

HOG postupak



- Umesto da normalizujemo histogram jedne ćelije 8×8 (9 vrednosti), normalizujemo histogram većeg bloka, npr. 16×16
 - Ovaj blok sadrži 4 histograma koja se mogu konkatenirati u vektor od 36 elemenata
- Prozor od 16×16 zatim pomerimo za 8 piksela i ponavljamo ovaj postupak

HOG postupak

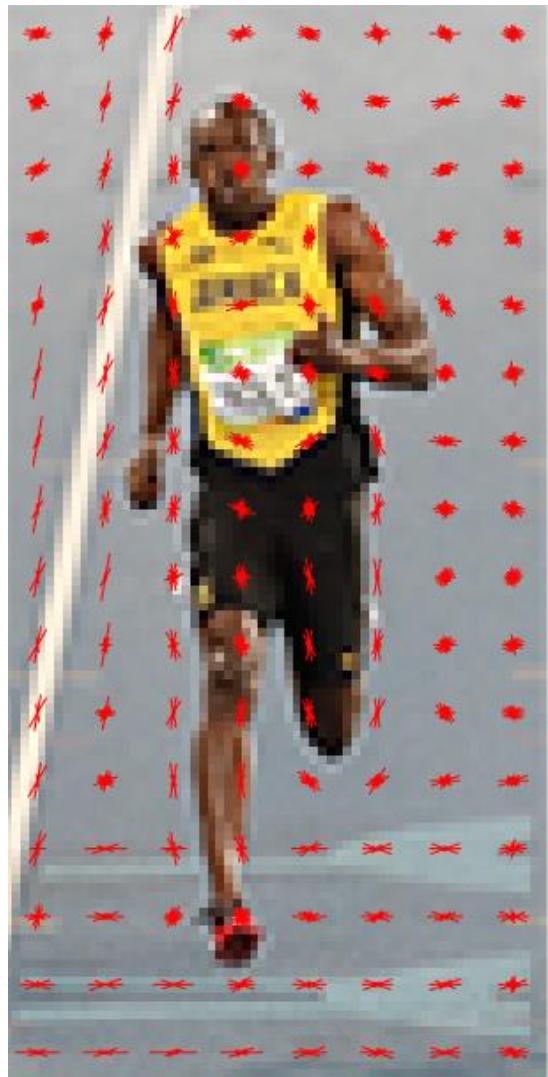
- Kako odabrati veličinu bloka za normalizaciju?
 - Blok postoji radi rešavanja problema varijacije osvetljenja
 - Veliki blok: lokalne promene su manje značajne
 - Tipično, blok se uzima da bude $2 \times$ veličina ćelije, ali treba isprobati i druge vrednosti
 - Tipično, blok se pomera za 50% svoje veličine

HOG postupak

5. Reprezentaciju cele slike dobijamo tako što konkateniramo dobijene vektore

- Za sliku veličine 64×128 imamo 7 horizontalnih i 15 vertikalnih pozicija za blokove 16×16 . Ukupno $7 \times 15 = 105$ pozicija
- Svaki blok 16×16 je reprezentovan vektorom od 36 elemenata
- Dakle, kada konkateniramo ove vektore, reprezentacija slike će biti vektor dimenzije $36 \times 105 = 3780$
- Ovo je značajno manje od originalnih dimenzija isečka slike $64 \times 128 \times 3 = 24\,576$

Vizuelizacija HOG deskriptora



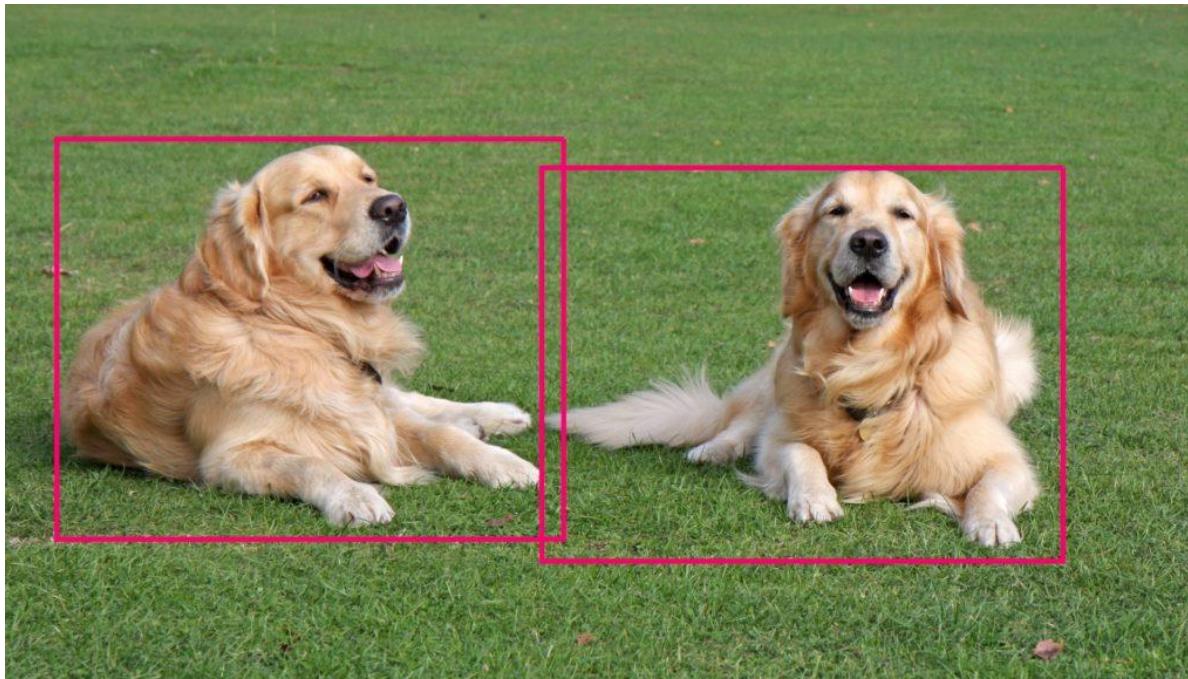
- Nacrtani su histogrami (normalizovani vektori od 9 elemenata) u 8×8 ćelijama
- Primetite da dominantna orientacija u histogramu prati oblik osobe

Detekcija objekata pomoću HOG deskriptora

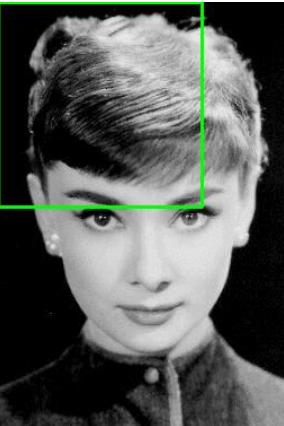
- Dakle, ideja je:
 1. Sastaviti trening skup od
 - P pozitivnih primera (slike koje sadrže objekat)
 - N negativnih primera (slike koje ne sadrže objekat)
 - U praksi $N \gg P$
 - Svaki primer (sliku) reprezentovati pomoću HOG deskriptora
 2. Trenirati model (npr. SVM) na trening skupu

Detekcija objekata pomoću HOG deskriptora

- Ali, trenirani model može da prepozna samo objekte na isečcima fiksne veličine
 - Selektovaćemo podregije (isečke) originalne slike i primeniti trenirani model da vidimo da li ti isečci sadrže sliku
 - Objekat je lociran tamo gde klasifikator vrati najveću verovatnoću da isečak sadrži objekat

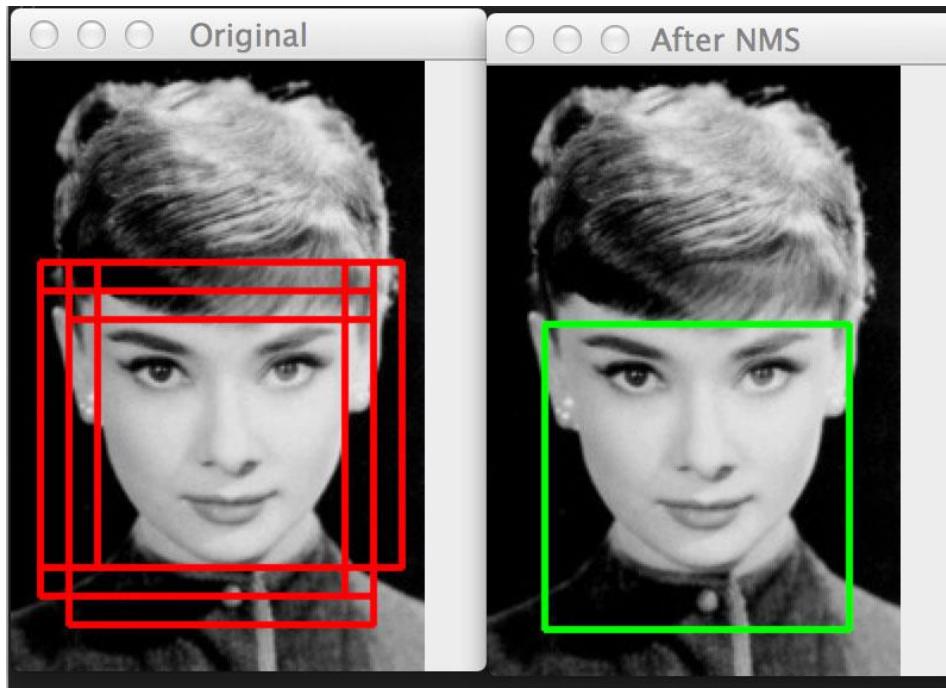


Sliding window



- Najjednostavniji način da ovo uradimo je *Sliding Window Algorithm*
 - Pomeramo „prozor“ preko slike da selektujemo isečak
 - Za isečak izračunamo HOG deskriptor i primenimo trenirani model da vidimo da li isečak sadrži objekat ili ne
- Nedostaci:
 - Ne samo da moramo da pretražimo sve moguće lokacije na slici već da isto uradimo za njene skalirane verzije
 - Model je treniran za objekte određene veličine – ako je objekat previše velik da bude obuhvaćen isečkom, model ga neće prepoznati
 - Ovaj algoritam podrazumeva da su svi objekti fiksnih proporcija
 - Slika je 2D projekcija 3D objekta – proporcije i oblik mogu da variraju u zavisnosti od ugla iz kog je objekat uslikan
 - Možemo primeniti pristup za više različitih proporcija, ali ovo uvećava kompleksnost

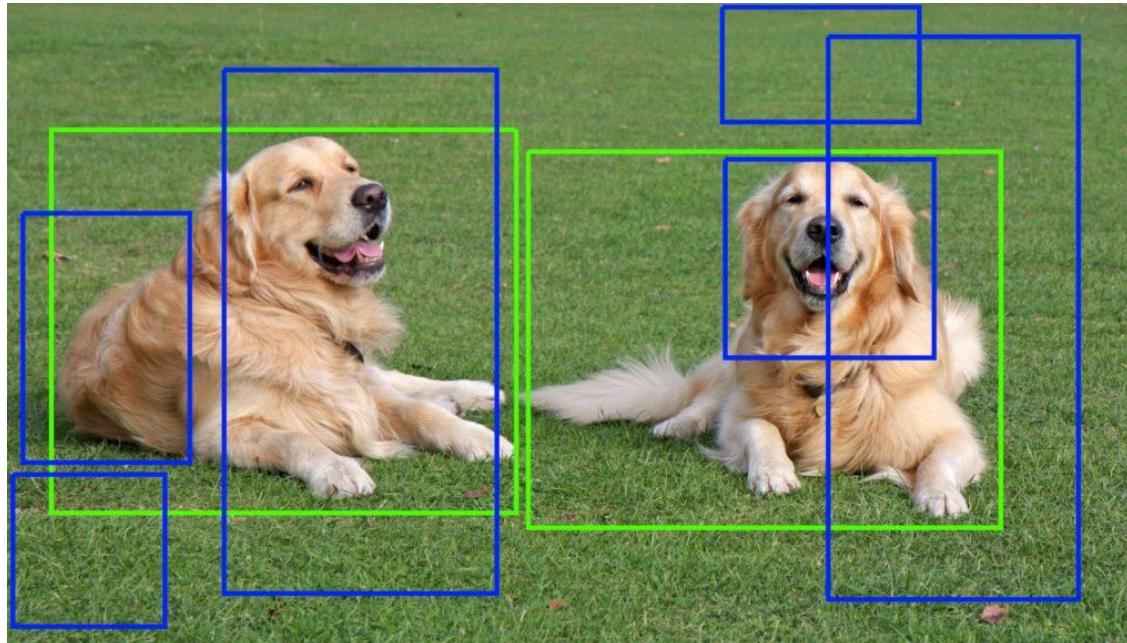
Sliding window



- Na slici je prikazan *overlapping bounding box problem*
 - Dobili smo više isečaka na kojima je (korektno) prepoznato lice
 - Ali svi se odnose na isto lice
 - Ovo se rešava primenom nekog *non-maximum suppression* algoritma
<https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/>

Region Proposal Algorithms

- Alternativa *sliding window* pristupu
- Ulaz u algoritam je slika, a izlaz svi isečci-kandidati koji verovatno sadrže objekat
- Za svaki ovakav isečak izračunamo HOG deskriptor i primenimo obučeni klasifikacioni model



Region Proposal Algorithms

- Isečci-kandidati se određuju segmentacijom
 - Segmentacija grupiše piksele koje su međusobno slični po nekom kriterijumu (boja, tekstura,...) u povezane regije
 - Budući da smo grupisali piksele slike u manji broj segmenata Imaćemo značajno manje isečaka na koje moramo primeniti klasifikator u odnosu na *sliding window* pristup
 - Isečci će biti različite veličine i proporcija
- Segmentacija bi trebala imati veoma veliki odziv
 - Objekti se moraju naći među isečcima-kandidatima
 - *False positives* su manji problem od *false negatives* (njih filtriramo naknadno pomoću klasifikatora)

Segmentacija

- Postoji mnogo pristupa za segmentaciju
 - Objectness
 - Constrained Parametric Min-Cuts for Automatic Object Segmentation
 - Category Independent Object Proposals
 - Randomized Prim
 - Selective Search
- Najčešće korišćen pristup je *Selective Search*
 - Brz i ima veoma veliki odziv
 - <https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/>

Literatura

- <http://www.cs.duke.edu/courses/fall17/compsci527/notes/hog.pdf>
- <https://www.learnopencv.com/image-recognition-and-object-detection-part1/>
- <https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/>
- Rad u kome je predstavljen HOG:
 - Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.
<http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>
- State of the Art of Object Recognition Techniques
https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/SS2016_AS_ObjectRecogn.pdf