



VILNIAUS GEDIMINO  
TECHNIKOS UNIVERSITETAS  
**ELEKTRONIKOS FAKULTETAS**  
**ELEKTRONINIŲ SISTEMŲ KATEDRA**

Domantas Skuja

**LICENCIJŲ TIPAI IR JŲ PANAUDOJIMAS PROGRAMINĖJE  
ĮRANGOJE, SKIRTOJE GILIAJAM MOKYMUISI**

Atvirojo kodo įranga mokslui, verslui ir valdymui modulio referatas

**Atliko**

Domantas Skuja

(vardas, pavardė)

(parašas, data)

## TURINYS

|  |    |
|--|----|
| Paveikslų sąrašas  | 4  |
| Ivadas   | 5  |
| Licencijos   | 6  |
| 1.1. Licencijavimo tipai .....   | 6  |
| 1.2. Laisvosios licencijos tipai .....   | 6  |
| 1.2.1. MIT .....   | 6  |
| 1.2.2. BSD .....   | 6  |
| 1.2.3. Apache .....  | 7  |
| Giliojo mokymosi Algoritmų apžvalga  | 8  |
| 1.3. Sąsūkos dirbtiniai neuronų tinklai (SDNT) .....                                       | 8  |
| 1.4. Regionais pagristas sasūkos dirbtinis neuronų tinklas (R-CNN).....                    | 9  |
| 1.5. Greitas regionais pagristas sasūkos dirbtinis neuronų tinklas (Fast R-CNN).....       | 10 |
| 1.6. Greitesnis regionais pagristas sasūkos dirbtinis neuronų tinklas (Faster R-CNN) ..... | 11 |
| 1.7. Regionais pagristas vien tik sasūkų dirbtinis neuronų tinklas (R-FCN).....            | 12 |
| 1.8. YOLOv2 .....  | 13 |
| 1.9. Single Shot Detector (SSD) .....  | 14 |
| 1.10. Orientuotų gradientų histograma (HOG) .....  | 15 |
| Programinės įrangos ir Jų licencijos   | 16 |
| 1.11. Programiniai paketai .....   | 16 |
| 1.11.1. OpenCV .....   | 16 |
| 1.11.2. TensorFlow .....   | 16 |
| 1.11.3. CMake .....  | 16 |
| 1.11.4. CUDA .....   | 17 |
| 1.12. Python bibliotekos.....  | 17 |
| 1.12.1. NumPy .....  | 17 |
| 1.12.2. PyTorch.....   | 17 |
| 1.13. Brūkšninio kodo dekodavimo bibliotekos .....   | 18 |
| 1.13.1. ZBar .....   | 18 |
| 1.13.2. ZXing .....  | 18 |
| Išvados  | 19 |
| Literatūra   | 20 |

## PAVEIKSLŲ SĄRAŠAS

|   |                                     |
|---|-------------------------------------|
| pav. 1 SDNT architektūros struktūrine diagrama .....  | 9                                   |
| pav. 2 R-CNN algoritmo sekā .....   | 10                                  |
| pav. 3 Fast R-CNN architektūros struktūrinė diagrama .....  | 11                                  |
| pav. 4 Faster R-CNN architektūros struktūrinė diagrama .....  | 12                                  |
| pav. 5 R-FCN architektūros struktūrinė diagrama.....  | 13                                  |
| pav. 6 YOLO architektūros veikimo pavyzdys .....  | 14                                  |
| pav. 7 SSD arcitektūros struktūrine diagrama .....  | 14                                  |
| pav. 8 HOG struktūrinė schema.....  | 15                                  |
| pav. 9 1D brūkšninio kodo aptikimo su integruotu gilioju mokymusi ir geometriniu modeliu struktūrinė schema ..... | <b>Error! Bookmark not defined.</b> |
| pav. 10 Kampo histogramos pavydys .....   | <b>Error! Bookmark not defined.</b> |

## ĮVADAS

Brūkšninių kodų technologija yra viena pagrindinių sėkmingos įmonės priemonių. Tad šiandien juos galime surasti visur. Pavyzdžiu i ligoninėse vaistų brūkšniniai kodai užtikrina, kad vaistai skiriami tinkamam pacientui. Pašto tarnybos plėčiai naudoja brūkšninius kodus, kad galėtų stebėti siuntas visame pasaulyje. Taip pat industrinės įmonės sandėliavimo tvarkai ir atsekamumui bei daug kitų sričių.

Vartotojai dažnai susiduria su QR kodais norėdami atsisiusti mobiliają programėlę ar užsisakyti maistą restorane. Tad nėra nei dienos, kai žmogus vienaip ar kitaip nesusidurtu su brūkšniais kodais.

Tai ir nulėmė mano pasirinkta magistrinio darbo tema kuri yra „QR ir brūkšninių kodų paieška ir atpažinimas vaizduose taikant giliojo mokymosi metodus“. Šio referato tikslas yra atskleisti keletą pagrindinių licencijų tipų bei giliojo mokymosi programinių įrangų kurios bus naudojamos pasirinktame baigiamajame darbe.

## LICENCIJOS

### 1.1. Licencijavimo tipai

Šiame darbe naudojamos programinės įrangos yra skirstomos į 2 pagrindinės licencijų rūšys:

**Nuosavybinė licencija** (*angl. Proprietary Licence*) – teisės į programą ar kitą autorinį kūrinį priklauso jį užsakiusiai įmonei ar asmeniui. Vartotojui leidžiama naudotis programa licencijoje nustatytomis sąlygomis.

**Laisvoji licencija** (*angl. Free Licence*) – licencija, leidžianti vartotojui kopijuoti bei tobulinti kūrinį, platinti jį bei juo remiantis sukurtus kūrinius, tačiau gali riboti arba drausti toliau platinamų kūrinii licencijos keitimą[1].

Kadangi darbe daugiau skiriama dėmesio laisvosioms licencijoms tad šiame skyriuje apžvelgsime kokio tipo šių licencijų galime sutikti nagrinėjant QR ir brūkšninių kodų paieškos ir atpažinimo vaizduose temą.

### 1.2. Laisvosios licencijos tipai

#### 1.2.1. MIT

MIT yra viena paprasčiausiu ir lengviausiai įsisavinamų laisvųjų licencijų. Kurios apibrėžime yra tik 172 žodžiai. Pagrindiniai reikalavimai yra tai jog į kodo kopiją ir (arba) modifikaciją reikia įtraukti du dalykus:

- Originalus pranešimas apie autorių teises
- Pačios licencijos kopija

#### 1.2.2. BSD

BSD atvirojo kodo programinės įrangos licencijų šeima turi ilgą istoriją. Pirmoji versija buvo sukurta UC Berkeley 1980 m., Siekiant licencijuoti naują universiteto UNIX operacinę sistemą „Berkeley Software Distribution“ arba BSD OS. Ši pradinė versija, žinoma kaip 4-Clause BSD licencija, kuri šiandien jau nėra naudojama taip pat kaip ir 0-Clause variantas (kuris iš esmės yra lygiavertė viešosios nuosavybės licencija). Tad šiai dienai yra likę 2 naudojami variantai: 2-Clause ir 3-Clause. BSD licencijai iš programinės įrangos kūrėjo daug nereikalauja. Tiesą sakant, BSD 3-Clause licencijos reikalavimai yra labai panašūs į plačiai naudojamas MIT licencijos reikalavimus. Kaip ir MIT licencija, BSD 3-Clause licencijos straipsnis yra labai trumpas. BSD 3-Clause licencijoje yra du pagrindiniai reikalavimai tiems, kurie nori naudoti licencijuotą kodą. Jie iš tikrujų yra tokie patys kaip MIT licencijos.

Jei planujate kopijuoti, modifikuoti ar platinti bet kurį kodą, licencijuotą pagal BSD, turite įtraukti:

- Originalų pranešimą apie autorijų teises
- Pačios licencijos kopiją

### 1.2.3. Apache

Apache License 2.0 yra laisvoji licencija, o tai reiškia, kad vartotojai, naudodami kodą, gali daryti (beveik) viską, ko nori, išskyrus keletą išimčių. Tai suteikia programinės įrangos vartotojui laisvę ją naudoti bet kokiems tikslams, platinti, modifikuoti ir platinti modifikuotas tos programinės įrangos versijas. Tačiau Apache licencija nereikalauja, kad modifikuotos programinės įrangos versijos būtų platinamos naudojant tą pačią licenciją, arba net jos turi būti platinamos kaip nemokama programinė įranga. Apache licencijoje reikalaujama, kad gavėjai būtų informuojami tik apie tai, jog platinant buvo naudojamas Apache licencijos kodas, priešingai nei Copyleft licencijose [2]. Pagrindiniai licencijos reikalavimai:

- Originalus pranešimas apie autorijų teises
- Pačios licencijos kopija
- Paskelbti apie svarbius pradinio kodo pakeitimus
- NOTICE failo su priskyrimo pastabomis kopija (jei tokią yra originalioje bibliotekoje) Trečias aukščiau išvardytas reikalavimas ir yra pagrindinis Apache licencijos 2.0 lyginant su kitomis laisvosiomis licencijų skirtumas. Jei atliksite kokių nors esminius licencijuoto kodo pakeitimus, turėsite juos paskelbti viešai sekančioje atnaujintoje versijoje, kurią platsite.

Kaip ir „MIT“ licencijoje, „Apache“ licencijoje 2.0 reikalaujama naudojant kodą įtraukti originalų pranešimą apie autorijų teises bei pilną licencijos kopija. Tačiau tai nėra vienintelis reikalavimai. „Apache“ licencijoje 2.0 taip pat teigama, kad visi, kurie žymiai modifikuoja kodą, privalo aprašyti savo pakeitimus. Be to, atvirojo kodo bibliotekoje turi būti failas „NOTICE“ su priskyrimo pastabomis. Jei nusprendi naudoti kodą su šia licencija taip pat privalai įtraukti failą „NOTICE“. Kartu su šiais apribojimais yra keletas papildomų privalumų. „Apache 2.0“ aiškiai suteikia autorijų teisių savininkams teisę reikalauti savo darbų patentų. (Nors daugelis ekspertų teigia, kad MIT licencijos tekste pateikiama panaši patentų apsauga) Be to, kiekvienas, kuris naudojas kodu, gali suteikti licencijuotos programinės įrangos garantiją.

## GILIOJO MOKYMO ALGORITMŲ APŽVALGA

Kompiuterinė rega yra tarpdisciplininė sritis, kuri pastaraisiais metais sulaukia daugybės dėmesio. Viena iš neatsiejamų kompiuterinės regos sričių yra objektų aptikimas. Ši sritis nagrinėja žmonių, daiktų ir kitų objektų aptikimą ir klasifikavimą [1].

Pagrindinis skirtumas tarp aptikimo ir klasifikavimo algoritmu yra tai, kad pirmojo algoritmo rezultatas yra vaizde nubrėžtas keturkampis langas vaizduojantis objekto vietą. Be to, objekto aptikimo atveju nebūtinai reikia nupiešti tik vieną keturkampį, paveikslėlyje gali būti daugybė ribojančių langelių, vaizduojančių skirtingus objektus.

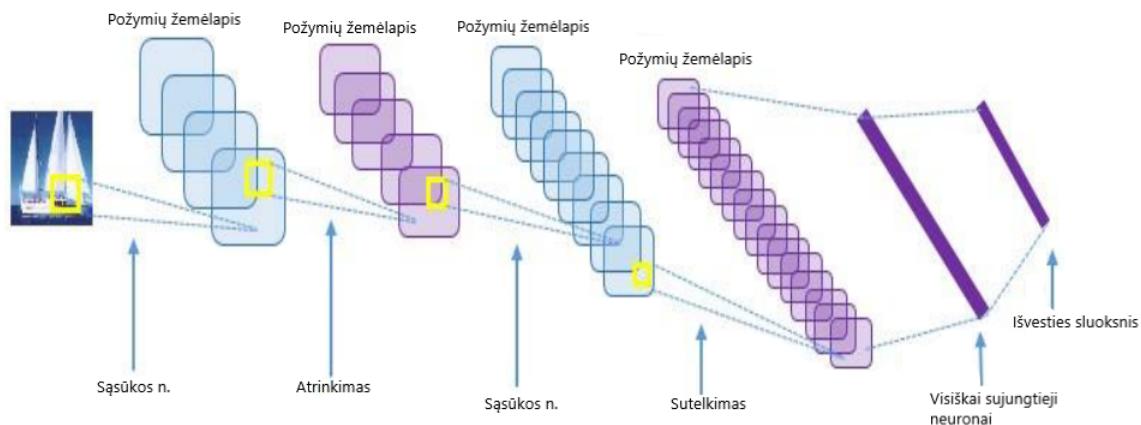
Priežastis, kodėl negalime spręsti šios problemos kurdami standartinį sąsūkos tinklą, po kurio eina visiškai prijungtas sluoksnis, yra ta, kad nežinome kokia turėtų būti išvestis. Ar aptiksime tik viena objektą ar jų grupę, o gal daugybe skirtingų objektų? Paprasčiausias būdas išspręsti šią problemą būtų pasirinkti įvairius dominančius regionus iš įvesties vaizdo ir jiems pritaikyti sąsūkos dirbtinius neuronų tinklus (angl *convolutional neural networks*), kad klasifikuotų objektą pasirinktame regione. Bet šio būdo didžiulė problema yra ta, kad dominantys objektai gali turėti skirtingą vietą ir proporcijas vaizde. Todėl būtume priversti pasirinkti daug regionų ir atlikti begalę matematinių skaičiavimų kuriems reikėtų daug kompiuterinių resursų. Spręsti šiai problemai buvo sukurti algoritmai kaip HOG, R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, YOLO, SSD ir kt., kuriuos šiame skyriuje ir aptarsiu. Kadangi visi metodai naudoja sąsūkos dirbtinius neuronų tinklus juos apžvelgsiu pirmiausiai.

### **1.3. Sąsūkos dirbtiniai neuronų tinklai (SDNT)**

Sąsūkos dirbtinių neuronų tinklai (SDNT) – tai dirbtinis neuronų tinklas sudarytas iš sąsūkos sluoksniių ir veikiantis tiesioginiu sklidimu, sąsūkos sluoksniuose atskiri neuronai apdoroja įvesties vaizdą slenkančiu langu, apskaičiuojant lange esančių verčių sandaugų su sąsūkos filtru vertėmis sumą. SDNT sukurtas remiantis biologiniais procesais ir yra daugiasluoksnio percepsono variacija, kuri pasiekia daug mažesnius skaičiavimo resursus bei leidžianti iš vaizdo išgauti dvimačius arba trimačius požymius. SDNT naudojami kompiuterinės regos, automatinio kalbos atpažinimo ir natūralios kalbos apdorojimo, garso atpažinimo ir bioinformatikos srityse, kuriose pasiekia itin gerų rezultatų [2].

SDNT naudoja atgalinio klaidos sklidimo algoritmą (angl. *back-propagation algorithm*) neuroninio tinklo mokymosi procesui testi, o jo spartinimui dažniausiai klaidos sklidimo funkcija yra parenkama ReLU. Tinklo svoriams atnaujinti yra naudojamas stochastinis gradientas (angl. *stochastic gradient*). O optimizuotiems požymiams (angl. *optimized features*) surasti yra pasitelktos sąsūkos operacijos ir sutelkimo procesai (angl. *pooling*) [1].

Paveiksle 1 yra pavaizduotas sasūkos dirbtinio neuronų tinklo struktūra. Kuri yra sudaryta iš sasūkos, sutelkimo arba atrinkimo (angl. *subsampling*) ir visiškai sujungtujų neuronų. SDNT



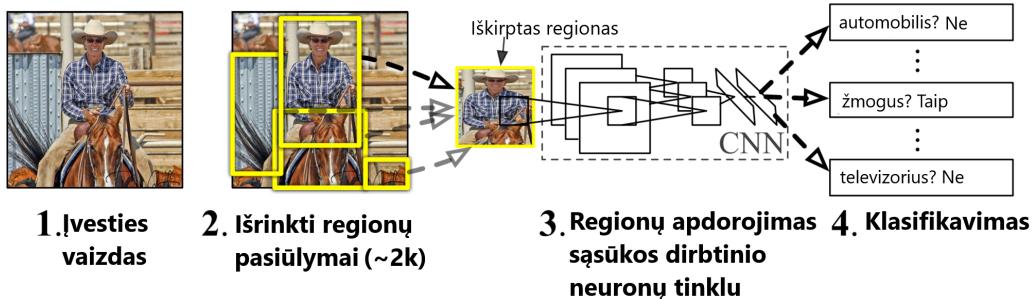
pav. 1 SDNT architektūros struktūrine diagrama

#### 1.4. Regionais pagrīstas sasūkos dirbtinis neuronų tinklas (R-CNN)

Vienas iš pirmųjų metodų sukurtas spręsti didelio langu skaičiaus problemai – regionais pagrīstas sasūkos dirbtinis neuronų tinklas (angl. *region-based convolutional neural networks*) ir licencijuotas **MIT** licencija. Buvo pasiūlyta naudoti atrankinę paiešką, kad iš įvesties vaizdo išrinktu apie 2000 langų kurie tolimesnėje eigoje pavadinti regionų pasiūlymai (angl. *region proposals*) [3]. Ši atrankinė paieška yra vykdoma pasitelkiant selektyvios paieškos algoritmu kuris sudarytas iš 3 etapų[4]:

1. Iš smulkiai langų sukuriama regionų pogrupiai
2. Rekursyviai sujungiami panašūs regionai naudojant godujį algoritmą (angl. *greedy algorithm*)
3. Sujungtujų regionų pagalba sugeneruojami galutiniai kandidatai.

Gauti du tūkstančiai regionų yra iškerpami ir apdorojami sasūkos dirbtinio neuronų tinklo kuris klasifikuoja kiekvieno iškirpto regiono objektą kaip pavaizduota 2 paveiksle.



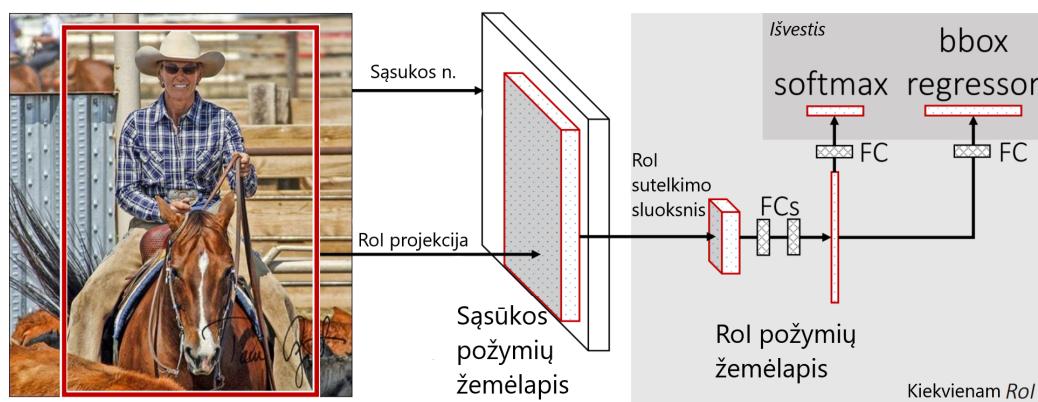
pav. 2 R-CNN algoritmo seka

Pagrindinės šio metodo problemas :

- Tinklo mokymas vis dar užima daug laiko, nes reikia klasifikuoti 2000 regionų vienam vaizdui.
- Negalima įgyvendinti realiuoju laiku, nes kiekvienam bandomajam vaizdui reikia maždaug 47 sekundžių.
- Selektyvios paieškos algoritmas yra fiksujotis ir jo apmokinimas negalimas todėl yra nemažai pasiūloma prastų regionų.

## 1.5. Greitas regionais pagrįstas sasūkos dirbtinis neuronų tinklas (Fast R-CNN)

Ankstesniojo metodo autorius (R-CNN) išsprendė kai kuriuos R-CNN trūkumus ir sukurė greitesnį objektų aptikimo algoritmą kurį pavadino – greitas regionais pagrįstas sasūkos dirbtinis neuronų tinklas (angl. *Fast region-based convolutional neural networks*). Metodas yra panašus į R-CNN algoritmą. Tačiau užuot pateikę visus regiono pasiūlymus sasūkos dirbtiniams neuronų tinklui, metodas tinklui pateikia visą įvesties vaizdą, kad sukurtų sasūkos požymių žemėlapį. Pagal šį žemėlapį yra pasiūlomi regionai ir sujungiami į kvadratus tuo pačiu selektyviosios paieškos metodu. Toliau naudojant „RoI“ sutelkimo sluoksniu juos pertvarko į fiksujoto dydžio ir panaudoja kaip įvesti visiškai sujungtiems neuronams. Iš RoI požymių vektoriaus „softmax“ klasifikatoriaus pagalba nustato siūlomo regiono klasę ir ribojančio lanelio poslinkio vertes. Viso šio proceso architektūra pavaizduota 3 paveiksle [5].



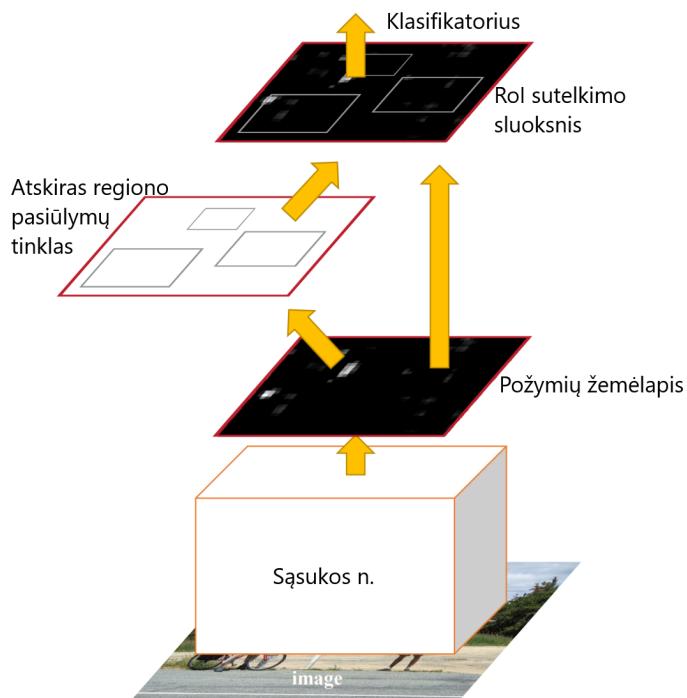
pav. 3 Fast R-CNN architektūros struktūrinė diagrama

Viena iš priežasčių kodėl metodas yra greitesnis nei R-CNN ta, kad nereikia kiekvieną kartą tiekti 2000 regiono pasiūlymų į sąsūkos neuronų tinklą. Vietoj to, sąsūkos operacija atliekama tik vieną kartą kiekvienam paveikslėliui ir iš jo sukuriamas funkcijų žemėlapis. Algoritmas licencijuojamas **MIT** licencija.

## 1.6. Greitesnis regionais pagrįstas sąsūkos dirbtinis neuronų tinklas (Faster R-CNN)

Abu aukščiau paminėti R-CNN algoritmai naudoja atrankinę paiešką, kad surastų regionų pasiūlymus. Tačiau šis procesas reikalaujantis labai daug laiko. Tad buvo sugalvota dar labiau patobulinti algoritmą pašalinant selektyvų paieškos algoritmą ir leidžiant tinklui pačiam išmokti regionų pasiūlymus.

Panašiai kaip Fast R-CNN vaizdas pateikiamas kaip įvestis į sąsūkos dirbtinių neuronų tinklą kuris sudaro požymių žemėlapį. Priešingai nei prieš tai aptartame metode šis nenaudoja selektyvaus paieškos algoritmo regiono pasiūlymus nustatyti, o naudoja atskirą tinklą. Pakeista architektūros struktūra pavaizduota pav. 4 [6]. Algoritmas licencijuojamas **MIT** licencija.

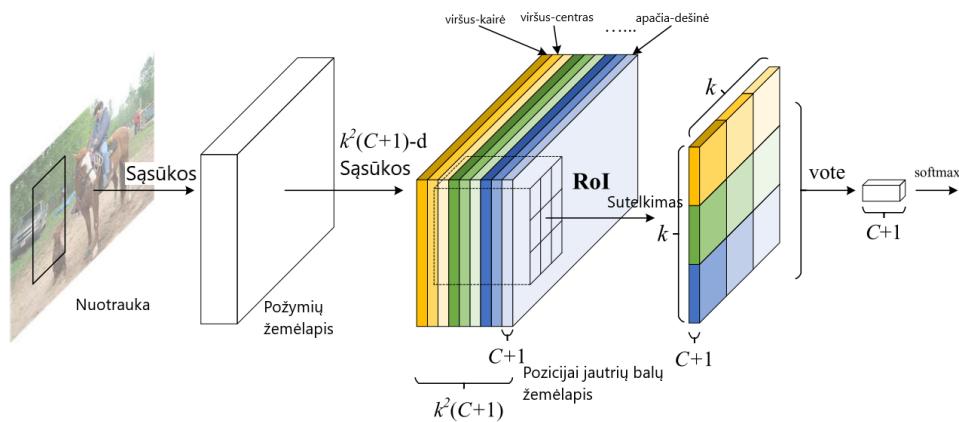


pav. 4 Faster R-CNN architektūros struktūrinė diagrama

## 1.7. Regionais pagristas vien tik sasūkų dirbtinis neuronų tinklas (R-FCN)

Regionais pagristas vien tik sasūkų dirbtinis neuronų tinklas (angl. *region-based fully convolutional network*) yra dar viena R-CNN atmaina skirta tiksliam ir efektyviam objektų aptikimui. Skirtingai nuo anksčiau aptartų regionais pagrįstų detektorių šis detektorius yra visiškai paremtas sasūkomis ir beveik visi skaičiavimai dalijami visame įvesties vaizde. Šis principas pavaizduotas paveiksle 5.

Pasielkti šiam tikslui yra naudojami pozicijai jautrūs balų žemėlapiai (angl. *position-sensitive score maps*), kad būtų išspręsta dilema tarp vertimo invariantišumo (angl. *translation-invariance*) klasifikuojant vaizdą ir vertimo variacijos (angl. *translation-variance*) nustatant objektus [7]. Šis metodas taip pat licencijuotas **MIT**.

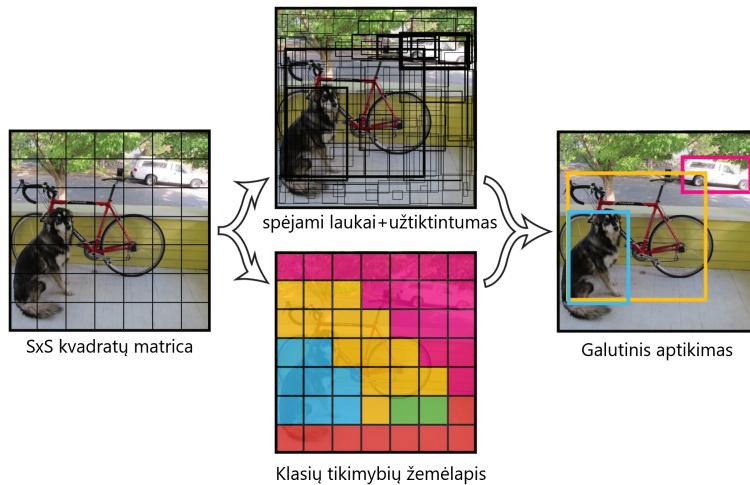


pav. 5 R-FCN architektūros struktūrinė diagrama

## 1.8. YOLOv2

Tai visiškai kitoks vaizdo apdorojimo principas, nuo prieš tai aptartų. Šiame metode nėra naudojami pasiūlytieji regionai ir jų paieškų. Išmetus šias architektūros dalis buvo sukurta pilnai tik sąsūkų dirbtiniai neuronų tinklais paremta sistema, galinti objektus vaizde atpažinti realiu laiku. Iš pradžių galėjės atpažinti kelias dešimtis skirtingų klasių, vėliau, antrojoje versijoje patobulintas ir dabar atpažstantis net 9000 skirtingų klasių su aukštesniu tikslumu bei spartesniu atlikimu nei pirmojo versijoje.

YOLO algoritmas sukurtas taip, kad iš karto atliktų skaičiavimus visam paveikslėliui. Iš to ir kilo pavadinimas (angl. *You Only Look Once*). Tam jėjimo paveikslėlis padalinamas į  $S \times S$  dydžio kvadratus. Jeigu objekto centras yra atitinkamoje laštelėje, klasifikuojamas objektas. Vienas langelis gali būti reikšmingas spėliojant B kiekj objektą apibrėžiančią stačiakampių, bei kiekvienam tokiam spėjimui pateikiamas skaičius, įvertinant, kaip užtikrintai tinklas siūlo tokį klasifikavimo rezultatą – objektą viduje ir stačiakampio koordinates. Jeigu nerandamas joks objektas, užtikrintumas dėl klasifikavimo yra lygus nuliui. Veikimo principas pavaizduotas paveiksle 6 [8]. Projektas **nėra licencijuotas** ir su juo galima daryti ką nori.

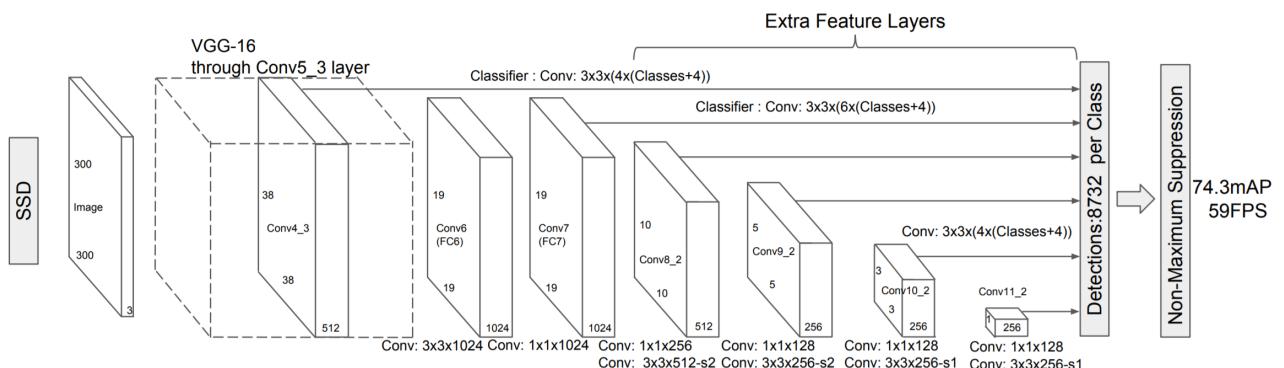


pav. 6 YOLO architektūros veikimo pavyzdys

### 1.9. Single Shot Detector (SSD)

SSD metodas pagrįstas sasūkų dirbtinių neuronų tinklu, kuris sukuria fiksuoto dydžio rinkinį ribojančių dėžių ir įvertina ar objektas tuoose laukeliuose atitinka turimus pavyzdžius, o po to atlieka ne maksimalus slopinimo žingsnį (angl. *non-maximum suppression step*), kad būtų galima gauti galutinius aptikimus. Pirmieji tinklo sluoksniai remiasi standartine architektūra, naudojama aukštos kokybės vaizdų klasifikavimui. Tada prie tinklo pridedame pagalbinę struktūrą (paveiksle 7 pavaizduota „Extra Feature Layers“), kad būtų galima aptiki šiuos pagrindinius požymius:

- Sasūkos veiksniai skirti aptikimui (angl. *convolutional predictors*)
- Daugiamastelis požymių žemėlapis (angl. *multi-scale feature maps*)
- Numatytieji langeliai ir proporcijos (angl. *default boxes and aspect ratios*)



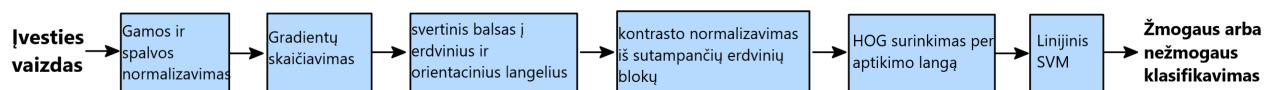
pav. 7 SSD arcitektūros struktūrine diagrama

Metodo licencija **MIT**.

## 1.10. Orientuotų gradientų histograma (HOG)

Orientuotų gradientų histograma (angl. *Histogram of Oriented Gradients (HOG)*) objektų aptikimui ir atpažinimui sukurtas metodas siekiant vaizdo medžiagoje atpažinti pėsčiuosius. Šis metodas yra paremtas vaizdo gradienčio orientacijos tankiame tinkelyje (angl. *image gradient orientations in a dense grid*) normalizuotos histogramos vertinimu. Pagrindinė mintis yra, kad bandomo atpažinti objekto išvaizdą ir forma remiantis intensyvumo gradientų (angl. *Intensity gradients*) arba krašto krypčių (angl. *edge directions*) pasiskirstymu [9].

Praktiškai tai yra įgyvendinama vaizdą dalijant į mažus nuotraukos langus. Kiekvienam langui yra sukaupiamas vienmatė histograma kuri yra sudaryta iš kiekvieno pikselio intensyvumo gradienčio ir krašto krypties vertės. Šių principų struktūrinė schema ir pavaizduota paveiksle 8 [10].



pav. 8 HOG struktūrinė schema

Sistema platinama **Apache 2.0** licencija.

## PROGRAMINĖS ĮRANGOS IR JŲ LICENCIJOS

### 1.11. Programiniai paketai

#### 1.11.1. OpenCV

Biblioteka turi daugiau nei 2500 optimizuotų algoritmų kuriuos sudaro klasikiniai bei pažangiausi mašininio mokymosi algoritmai. Šie algoritmai gali būti naudojami veidams, objektams aptikti ir atpažinti. Taip pat klasifikuoti žmogaus veiksmus vaizdo įrašuose, stebėti fotoaparato judestis, sekti judančius objektus, išgauti 3D objekto modelius, sukurti 3D taškinius debesis iš keletos kamerų ir sujungti vaizdus, kad būtų sukurta didelė skiriamoji geba. OpenCV biblioteka turi daugiau nei 54 tūkstančių vartotojų bendruomenę ir 18 milijonų atsiuntimų. Pagrindinės sėsajos kurias turi biblioteka yra C++, Python, Java ir MATLAB kurias palaiko Windows, Linux, Android ir Mac OS. OpenCV daugiausia naudojama realaus laiko programose ir jei įmanoma naudoja MMX ir SSE instrukcijas. Šiuo metu biblioteka aktyviai pritaikoma CUDA ir OpenCL sėsajoms. Dėl šių privalumų kartu su gerai žinomomis kompanijomis, tokiomis kaip „Google“, „Yahoo“, „Microsoft“, „Intel“, „IBM“, „Sony“, „Honda“, „Toyota“, kurios naudoja šią biblioteką, yra daugybė startuolių „Applied Minds“, „VideoSurf“ ir „Zeitera“, kurie plačiai pritaikė OpenCV biblioteką. Taip pat lengvam sistemos panaudojimui prisideda ir jos lecencijavimas. OpenCV 4.5.0 versija ir aukštesnė yra licencijuota Apache 2 licencija, o OpenCV 4.4.0 ir žemesnė turi **3-clause BSD** licencija. Kurios leidžia lengva panaudojimą ir prisaikymą.

#### 1.11.2. TensorFlow

TensorFlow yra mašininio mokymosi sistema, veikianti didelio masto, nevienalytėse aplinkose. Ji turi išsamią, lanksčią ekosistemą kuri leidžia piketuotojams sukurti ir lengvai įdiegti pažangias ML sistemas. Šią biblioteką sukūrė tyrėjai ir inžinieriai, dirbantys „Google Brain“ komandoje „Google“ mašinų intelekto tyrimų organizacijoje, kad palengvintų mašininio mokymosi ir giliųjų neuroninių tinklų tyrimų atlikimą. Sistema yra pakankamai bendra, tad ją galima pritaikyti ir įvairiose kitose srityse[8]. TensorFlow suteikia stabilų Python ir C++ aplikacijų programavimo sėsaja (angl. application programming interface – API ), taip pat yra pritaikymai ir kitoms kalbos tačiau jų stabilumas nėra garantuojamas. Biblioteka yra licencijuojama **Apache 2.0** licencija.

#### 1.11.3. CMake

CMake yra atviro kodo įrankių grupė pritaikyta kelioms platformoms bei skirta kurti, išbandyti ir glaudinti programinę įrangą. Ji yra naudojama programinės įrangos kompliliavimo procesui valdyti pasitelkiant paprasčiausią platformą ir kompiliatoriaus nepriklausomus konfigūracijos failus. CMake

įrankių rinkinį sukūrė „Kitware“, reaguodama į galingos daugiašalės platformos poreikį atvirojo kodo projektams, tokiemis kaip ITK ir VTK. Šiandien CMake generuoja šiuolaikines sistemas, tokias kaip „Ninja“, taip pat projekto failus integravimai kūrimo aplinkai (*angl. Integrated Development Environment - IDE*) pvz., „Visual Studio“ ir „Xcode“. CMake palaiko Windows, MacOS, Linux, FreeBSD, OpenBSD, Solaris ir AIX ir yra licencijuota **BSD 3-clause** licencija.

## 1.11.4.CUDA

CUDA tai NVIDIA sukurta lygiagreti skaičiavimo platforma bei programavimo modelis (*angl. Compute Unified Device Architecture – CUDA*), skirtas bendram skaičiavimui grafiniuose procesoriuose (*angl. graphics processing unit GPU*). Visa tai padeda išnaudoti GPU branduolius lygiagrečiam skaičiavimui kurių yra tūkstančiai. Naudodami CUDA, kūrėjai turi galimybę programuoti tokiomis kalbomis kaip C, C++, Fortran, Python ir MATLAB kur lygiagrečius skaičiavimus gali išreikšti per plėtinius kelių pagrindinių raktinių žodžių pavidalu. CUDA palaiko visas pagrindines Nvidia vaizdo plokštės bei Windows ir Linux operacines sistemas ir yra licencijuota **nuosavybine** licencija.

## 1.12.Python bibliotekos

### 1.12.1. NumPy

NumPy yra pagrindinis Python skaičiavimo paketas skirtas mokslininkams. Tai yra biblioteka, leidžianti naudoti objektus sudarytus iš daugiamatių masyvų taip pat įvairius išvestinius objektus (pvz., užmaskuotas masyvus ir matricas) bei pagrindinius matematinius veiksmus greitoms masyvų operacijoms, įskaitant matematinių, loginių, formos manipuliavimą, rūšiavimą, pasirinkimą, įvestį / išvestį, diskrečias Furjė transformacijas, tiesinė algebra, statistiką, atsitiktinį modeliavimą ir daug daugiau. NumPy licencijuota **BSD 3-Clause** licencija.

### 1.12.2. PyTorch

PyTorch yra optimizuota tensorių biblioteka giliam mokymuisi naudojanti procesorius ir GPU. Ši biblioteka yra Python paketas, teikiantis dvi aukšto lygio funkcijas:

- Tensoriaus skaičiavimas (toks kaip „NumPy“) su dideliu GPU našumu.
- Gilūs neuroniniai tinklai, sukurti ant „tape-based autograd“ sistemos.

Pagrindinis PyTorch panaudojimas yra pakeisti NumPy biblioteką, kad būtų išnaudojama GPU galia bei suteiktų maksimalų lankstumą ir našumą. PyTorch licencijuota **BSD** licencija.

## 1.13. Brūkšnio kodo dekodavimo bibliotekos

### 1.13.1. ZBar

ZBar yra atviro kodo programinė įranga, skirta brūkšniniams kodams dekoduoti iš įvairių šaltinių, tokį kaip vaizdo srautai, vaizdo failai ir neapdoroto intensyvumo jutikliai. Jis palaiko daugelį populiařių simbolij (brūkšninių kodų tipus), išskaitant EAN-13 / UPC-A, UPC-E, EAN-8, 128 kodą, 39 kodą, „Interleaved 2 of 5“ ir QR kodą. Biblioteka palaiko Python, Perl, C ++ programavimo sasajas bei turi labai supaprastintą C bibliotekos versija kuri yra tinkamą naudoti įterptinėse sistemoose. Pagrindiniai sistemos privalumai:

- Pritaikytą daugeliui platformų - Linux/Unix, Windows, iPhone, embedded ir t.t.
- Didelis greitis – real-time nuskaitymas iš vaizdo
- Mažas kodo dydis - pagrindinis skaitytuvas ir EAN dekoderis užima tik 1 tūkstantj eilučių.
- Jokių slankiojo kablelio operacijų

ZBar turi **GNU LGPL 2.1** licenciją, kad būtų galima plėtoti tiek atvirojo kodo, tiek komercinius projektus.

### 1.13.2. ZXing

ZXing ("zebra crossing") yra atvirojo kodo, kelių formatų 1D / 2D brūkšninių kodų dekodavimo biblioteka, įgyvendinta su Java, bet prieinama ir kitoms programavimo kalboms. Biblioteka palaiko brūkšninius kodus pavaizduotus lentelėje žemiau. ZXing licenzijuota **Apache 2.0** licencija.

#### Supported Formats

| 1D product            | 1D industrial | 2D           |
|-----------------------|---------------|--------------|
| UPC-A                 | Code 39       | QR Code      |
| UPC-E                 | Code 93       | Data Matrix  |
| EAN-8                 | Code 128      | Aztec        |
| EAN-13                | Codabar       | PDF 417      |
| UPC/EAN Extension 2/5 | ITF           | MaxiCode     |
|                       |               | RSS-14       |
|                       |               | RSS-Expanded |

## IŠVADOS

Pagrindinis šio referato tikslas atskleisti informaciją apie vyraujančias atvirojo kodo licencijas, bei kurios yra naudojamos giliojo mokymosi algoritmuose kurie yra naudojami objektams vaizde aptikti. Referate apžvelgtos MIT, Apache ir BSD licencijos. Taip pat keleta Python bibliotekų, brūkšninių kodų dekodavimo programų bei programinių paketų.

## LITERATŪRA

- [1] S. Hayat, S. Kun, Z. Tengtao, Y. Yu, T. Tu, and Y. Du, “A Deep Learning Framework Using Convolutional Neural Network for Multi-Class Object Recognition,” *2018 3rd IEEE Int. Conf. Image, Vis. Comput. ICIVC 2018*, pp. 194–198, 2018, doi: 10.1109/ICIVC.2018.8492777.
- [2] E. Šabanovič, *Sąsūkos dirbtinių neuronų tinklų įgyvendinimas spartinančiuose įrenginiuose vaizdams analizuoti realiuoju laiku*, no. March. 2019.
- [3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.
- [4] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-0620-5.
- [5] R. Girshick, “Fast R-CNN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.
- [6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
- [7] Y. Li, K. He, J. Sun, and others, “R-fcn: Object detection via region-based fully convolutional networks,” *Adv. Neural Inf. Process. Syst.*, no. Nips, pp. 379–387, 2016, [Online]. Available: <http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf>.
- [8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
- [9] C. Tomasi, “Histograms of Oriented Gradients,” *Comput. Vis. Sampl.*, pp. 1–6, 2012, [Online]. Available: <https://www.cs.duke.edu/courses/fall15/compsci527/>.
- [10] T. Surasak, I. Takahiro, C. H. Cheng, C. E. Wang, and P. Y. Sheng, “Histogram of oriented gradients for human detection in video,” *Proc. 2018 5th Int. Conf. Bus. Ind. Res. Smart Technol. Next Gener. Information, Eng. Bus. Soc. Sci. ICBIR 2018*, pp. 172–176, 2018, doi: 10.1109/ICBIR.2018.8391187.
- [11] Y. Xiao and Z. Ming, “1D barcode detection via integrated deep-learning and geometric approach,” *Appl. Sci.*, vol. 9, no. 16, pp. 1–12, 2019, doi: 10.3390/app9163268.