VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS INFORMATIKOS KATEDRA
INFORMATIKOS INSTITUTAS

Bakalauro baigiamasis darbas

Jogos pratimų atpažinimas naudojant gilųjį mokymąsi

(Yoga exercises recognition using deep learning)

Atliko: 4 kurso bioinformatikos studentas
Alanas Kučinskij (parašas)

Darbo vadovas:
lekt. Irus Grinis (parašas)

Darbą recenzavo:
dr. Kliment Olechnovič (parašas)

Vilnius
2020

Turinys

Įvadas

Terminas gilusis mokymasis pasirodė 1986 metais, tačiau iš pradžių šita sritis nesulaukė didelio populiarumo. Laikui bėgant, 2000-ųjų metų viduryje prasidėjo tikrasis giliojo mokymosi populiarumo pakilimas. Informacijos saugyklos talpa tapo didesnė, atmintis pigesnė ir tai lėmė ne tik giliojo mokymosi, bet ir visų sistemų mokymosi vystymą. Įvykus duomenų apdorojimo ir jų saugojimo pažangai ir padidėjus sukuriamų, perduodamų ir saugomų duomenų kiekiui atsirado poreikis naudoti neuronų tinklus. Per kelis dešimtmečius dirbtinio intelekto populiarumas, aktualumas ir panaudojimo galimybės labai išaugo. Teksto vertimas, kalbos ir muzikos atpažinimas, transporto priemonių valdymas, vaizdo ir teksto generavimas, problemų identifikavimas ir sprendimų priėmimas – tai tik keletas dirbtinio intelekto panaudojimo galimybių. Šiuo metu visos didžiausios pasaulio kompanijos turi atskirą padalinį, skirta giliajam mokymuisi, sistemų ir galimybių tobulinimui, ir visai dirbtinio intelekto sričiai vystyti.

Prieš dešimt metų vaizdo atpažinimas buvo labai komplikuotas ir reikalaujantis daugybės investicijų procesas. Bet šiais laikais ženkliai pagerėjo kompiuterių kalbos ir vaizdo atpažinimo galimybės, kompiuterių skaičiavimo greitis reikšmingai išaugo, tai leidžia konstruoti giliojo mokymosi neuronų tinklo modelius, kurie atpažins objektus nenaudojant specifinių jutiklių, kamerų ar kitų instrumentų. Naudojant giliojo mokymosi algoritmus galima sukurti neuronų tinklo modelį, kuris galėtu atpažinti žmogaus judesius. Kompanija *Microsoft Corporation* 2010 metais pristatė judesių aptikimo įrangą ir pavadino ją *Kinect*. Naudojant specifinius jutiklius ir kameras *Kinect* galėjo atpažinti žmogų, naudojant tokį funkcionalumą buvo sukurta nemažai įvairių programų. Dabar, kai gilusis mokymasis yra vystomas tokių garsių technologinių kompanijų kaip *Google*, naudojant paprastą kamerą galima išbandyti tokį funkcionalumą ir savo namuose. Tam nereikia pirkti brangios įrangos, viską galima išbandyti naudojant giliojo mokymosi algoritmus.

Žmogaus atpažinimo vaizde technologijos gali padėti teisingai atlikti, pavyzdžiui, jogos pratimus. Egzistuoja daugybė jogos statinių, dinaminių ir tempimo pratimų. Naudojant giliojo mokymosi galimybes galima sukurti programą, padedančią žmonėms stebėti ir kontroliuoti savo kūno savijautą, teisingai ir laiku atlikti pratimus. Jogos pratimai gerina atmintį, koncentraciją ir intelekto produktyvumą: stiprina visus jutiminius sugebėjimus, lėtina senėjimo procesus, grąžina sąnarių lankstumą, mažina nugaros skausmus, pašalina stuburo išlinkimus, gydo galvos skausmą, padeda išlaikyti tinkamą kūno svorį, suteikia kokybišką atsipalaidavimą, stiprina imuninę sistemą. Kursiniuose darbuose buvo išbandytas, panaudotas ir išnagrinėtas vienas iš populiariausių giliojo mokymosi neuronų tinklo modelių algoritmas, kuris yra skirtas žmogaus aptikimui vaizdo įraše, nuotraukoje arba realiu laiku, naudojant vaizdo kamerą. Buvo sukurta programa, kuri galėjo atpažinti paprasčiausius žmogaus judesius, tokius kaip: rankos pakėlimas virš galvos, rankų tempimas, dilbių sukimas. Šio darbo tikslas - įsigilinti į žinomiausias giliojo mokymosi *Javascript* bibliotekas, suprasti dirbtinio intelekto veikimo principus, išnagrinėti klasifikavimo modelio veikimo principą ir panaudoti su *Posenet* apmokytu modeliu.

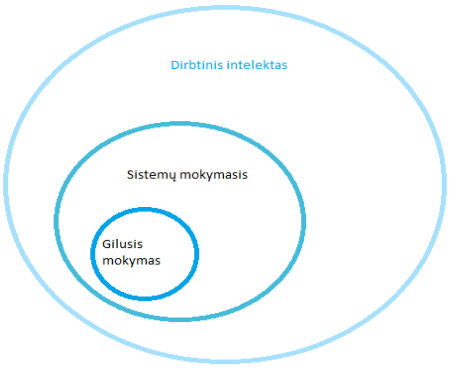
Uždaviniai

Išanalizuoti klasifikavimo modelių galimybes. Sukurti programą, gebančią atpažinti žmogaus apšilimo judesius naudojant du neuronų tinklo modelius.

1. Teorinė dalis

1.1. Įvadas į gilųjį mokymąsi

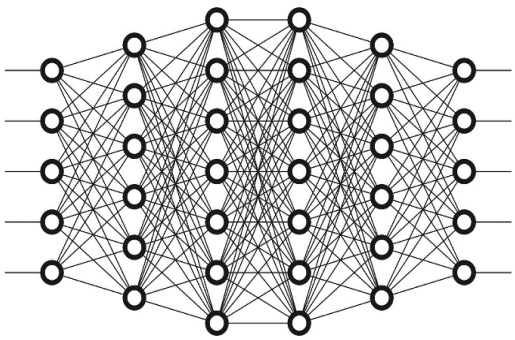
Geresniam supratimui, kas yra gilusis mokymasis, galima pavaizduoti kaip jis sąveikauja su dirbtiniu intelektu ir sistemų mokymusi.



1 pav. Sistemų ir giliojo mokymosi santykis

Pirmame paveikslėlyje matomas išorinis ratas, tai - dirbtinis intelektas, per vidurį - sistemos mokymasis, o mėlynas ratas - gilusis mokymasis.

Gilaus mokymosi metodo pagrinde yra sunkus neuronų tinklo modelis, sudarytas iš daugybės neuronų sluoksnių. Paveikslėlyje numeris 2 pavaizduotas daugybę sluoksnių turintis neuronų tinklas. Pirmasis neuronų tinklo sluoksnis (pirmas iš kairės) vadinamas įvedimu, o paskutinis neuronų tinklo sluoksnis - išvesties sluoksniu. Neuronas priima informaciją, įvertina ją ir aktyvuoja kitą sluoksnį. Vidutiniai sluoksniai neturi ryšio su išoriniu pasauliu, todėl tokie sluoksniai vadinami paslėptais.



2 pav. Gilaus mokymosi neuronų tinklo pavyzdys [Mah18]

1.1.1. Giliojo mokymosi veikimo principas

Vienas iš populiariausių giliojo mokymosi paaiškinimo pavyzdžių yra pavaizduotas paveikslėlyje numeris 3. Sąlygos: skaičius nuotraukoje visada turi būti vienas, o pats paveikslėlis turėtų būti juodai-baltas. Kiekvienas pikselis turi jam skirtą neuroną, kuris įvertina pikselio ryškumą nuo 0 iki 255, kai neuronų tinklas baigia savo įvertinimus - didelė tikimybė, kad jis geba atsakyti, koks skaičius yra pavaizduotas paveikslėlyje.

Kitas pavyzdys. Mažos raiškos paveikslėlyje rašomas skaičius nuo 0 iki 9. Prielaida: paveikslėlio raiška yra 64x64 pikselių. Sąlygos lieka kaip ir ankstesniame pavyzdyje: skaičius baltas, fonas juodas. Sudarytas giliojo mokymosi neuronų tinklas iš 4 neuronų sluoksnių: įvesties, dviejų paslėptų ir išvesties. Įvesties sluoksnyje neuronų skaičius bus lygus paveikslėlio aukščio ir pločio sandaugai. Kaip ir ankstesniame pavyzdyje, neuronai priima pikselių reikšmę: 0 arba 1. Nulinę reikšmę neuronas įgyja tuomet, kai priklausantis jam pikselis yra juodas, teigiamą reikšmę - kai pikselis yra baltas. Išvesties sluoksnis turi 10 neuronų, kurie ir duoda atsakymą į klausimą: koks yra skaičius? Pirmas ir antras paslėptieji sluoksniai gali veikti tokiu būdu: kiekvienas neuronas gali rodyti atskirą skaičiaus dalį: vertikalus brūkšnys, viršutinis ir apatinis apskritimai arba horizontalūs brūkšniai. Pavyzdžiui, skaičius 9 visada turės viršutinį apskritimą.

1.2. Giliojo mokymosi taikymo sritis

1.2.1. Kalbos atpažinimas

Gilaus mokymosi modeliai gali atpažinti ne tik kalbą, bet bandyti nuspėti dialektą.[Mar19] Ši užduotis labai sunki, todėl giliųjų neuronų tinklų naudojimas yra geriausias variantas, jei vertinti greičio atžvilgiu.

1.2.2. Teksto generavimas

Modelis gali išmokti tam tikros kalbos gramatikos, stiliaus ir skyrybos ženklų ypatumų ir bandyti generuoti taisyklingą tekstą. Teksto generavimas prasideda nuo kalbos modelio (angl. *language model*). Neuronų tinklas pasirenka žodį, kurį iš anksto žinojo, prideda prie sakinio ir taip kartoja iki kol nesigaus tekstas. Populiariausi teksto generavimo panaudojimo pavyzdžiai: *Twitter* pokalbių botai.

1.2.3. Vertimas

Garsiausias ir žinomiausias gilaus mokymosi, vertimų iš vienos į kitą kalbos srityje naudojimo pavyzdžių yra *Google Translate*.

1.2.4. Vaizdo generavimas

Giliojo mokymosi modeliai gali generuoti portretus ir peizažus. Vaizdo generavimo modeliai naudoja du neuroninius tinklus, veikiančius vienas paskui kitą. Pavyzdžiui, vienas neuronų tinklas generuoja portretą, o kitas modelis - jį įvertina. Antras neuronų tinklas arba atmeta sugeneruotą vaizdą, arba priima.

1.2.5. Autonominė transporto priemonė

Autonominės transporto priemonės yra viena iš žinomiausių taikymo sričių. Rizikos sumažinimo tikslais yra naudojama daug giliojo mokymosi algoritmų apmokytų modelių. Vienas modelis gali būti atsakingas už pėsčiųjų atpažinimą, kitas už kelio ženklų atpažinimą.

1.3. Giliojo mokymosi bibliotekos

Gilaus mokymosi algoritmai nėra paprasti. Norint konstruoti neuronų tinklų modelius, reikia turėti gerus matematikos ir programavimo pagrindus. Tačiau yra bibliotekos, kurios palengvina darbą su neuronų tinklais ir daro juos prieinamais net tiems žmonėms, kurie neturi specifinių anksčiau minėtų disciplinų žinių. Žinomiausios, prieinamiausios ir lengviausiai suvokiamos bibliotekos yra šios:

- *Tensorflow* – sistemų mokymosi biblioteka, skirta įvairiems programavimo įrankiams.[GR18]
- *PyTorch* – bilbioteka, skirta naudojimui su *Python* programavimo kalba. Dažnai naudojama žmogaus kalbos apdorojimui.
- *Sonnet* –- biblioteka, skirta veikti kartų su *TensorFlow*.
- *Keras* – abstrakti biblioteka, kurios pagrindu gali būti bet kuris iš šių programinių įrankių: *Microsoft Cognitive Toolkit*, *R*, *Theano*, *PlaidML* ir *TensorFlow*.
- *MXNet* –- biblioteka, leidžianti naudoti kelis *GPU* vienu metu. Galima kurti gerai optimizuotas programas.
- *Caffe* –- greita ir žinoma biblioteka.
- *Theano* –- leidžia optimizuoti ir apibrėžti matematines išraiškas.
- *FLearn* – biblioteka, sukurta *TensorFlow* pagrindu, siekiant suteikti aukšto lygio *API*.

1.3.1. TensorFlow.js

TensorFlow – viena iš populiariausių neuronų tinklo modelių kūrimo biblioteka. Buvo sukurta *Google*, skirta naudojimui su *Python* programavimo kalba.[16] *TensorFlow* galima naudoti kaip paprastų neuronų tinklų kūrimui, taip ir sunkiam giliam mokymuisi. Viskas, kas yra susiję su neuronų tinklais, reikalauja ne tik galingų kompiuterinių resursų, bet ir geros darbų optimizacijos. Yra žinoma, kad *Python* programavimo kalba nėra geriausias įrankis optimizuotų programų kūrimui. Siekiant užtikrinti gerą optimizaciją, buvo priimtas sprendimas parašyti *TensorFlow* biblioteką naudojant *C++* programavimo kalbą.[**tensorIntroSec**] Toks būdas leido sukurti greitą ir gerai optimizuotą biblioteką, kuri būtų naudojama su *Python* programavimo kalba. *Python* yra lengvai išmokstama programavimo kalba, kartu su *TensorFlow* ji leidžia naudoti ir kurti neuronų tinklus žmoniems, kurie nėra programavimo profesionalai. Norint palengvinti neuronų tinklų naudojimą platesniam žmonių ratui, buvo priimtas sprendimas sukurti biblioteką, kuri galėtų veikti su interneto naršyklės programavimo kalba – *JavaScript*. Taip buvo sukurtas *TensorFlow.js* - atviro kodo biblioteka, lengvai prieinama, duodanti galimybe kurti, apmokyti ir paleisti neuronų tinklų

modelius savo naršyklėje. Verta paminėti, jog jei reikia gerai optimizuotos programos, *Javascript* irgi nėra visiškai tinkama programavimo kalba. Kad visi modeliai veiktų sklandžiai ir greitai, *TensorFlow.js* buvo pagreitintas naudojant *WebGL API*. Didelių skaičiavimų kiekių perkeliamas iš *CPU* į *GPU* leidžia ženkliai pagreitinti programų veikimą.

1.3.1.1. Kaip instaliuoti *Tensorflow.js* ?

Pradėti naudoti *TensorFlow.js* savo programoje labai lengva, yra keli būdai tai padaryti. Pirmas: įdėti *ScriptTag* į savo *index.html* failą.

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js"></script>
```

3 pav. Tensorflow.js ScriptTag [20b]

Antras būdas: pridėti *TensorFlow* biblioteką prie savo programos – instaliuoti per *NPM* arba *YARN*.

```
yarn add @tensorflow/tfjs
// Arba
npm install @tensorflow/tfjs
```

Lengviausiai tai daryti naudojant *ScriptTag*. Bet šis variantas turi trūkumą: be interneto programa neveiks.

1.3.1.2. Ką gali TensorFlow.js ?

Štai išrašyti tik kelis variantai *TensorFlow.js* panaudojimo būdų:

- Naudojant naują informaciją apmokyti neuronų tinklo modelį.
- Importuoti iš anksto treniruotą neuronų tinklą ir naudoti savo programoje.
- Kurti naujus neuronų tinklus ir savarankiškai aprašyti jų architektūrą.

TensorFlow.js turi daugybę jau sukurtų ir apmokytų giliojo mokymosi algoritmo neuronų tinklo modelius. Pavyzdžiui:

- Vaizdo klasifikacija –- *MobileNet* modelis.
- Objektų aptikimas paveikslėlyje –- *COCO-SSD* modelis.
- Kūno segmentacija –- kūno vietų aptikimas, *BodyPix* modelis.
- Žmogaus pozicijos aptikimas –- *Posenet* modelis.
- Teksto toksiškumo nustatymas – *Toxicity* modelis.
- Kalbos komandų atpažinimas – *KNN* modelis.

1.3.2. Tenzoriai

Tenzorius – duomenų struktūra, kuri talpina skaliarą, arba vektorių, arba matricą. Paprasčiausi veiksmų pavyzdžiai su tenzoriais yra vektorinė sandauga, skaliarinė sandauga.

1.3.2.1. Kaip kurti tenzorius naudojant TensorFlow.js ?

Naudojant *TensorFlow* tenzorių kūrimas yra pakankamai lengvas veiksmas. Biblioteka turi gerą funkcijų rinkinį, kuris gerokai palengvina darbą. Viena iš tokių funkcijų yra *tf.tensor(value,shape?,dtype?)*. Funkcija priima tris parametrus, du paskutiniai iš kurių yra nebūtini. *Values* – duomenys, kuriuos norime išsaugoti. *Shape* – nustato, kokios formos bus tenzorius. *dtype* – įvedamų duomenų tipas. Pavyzdžiui, tikslas – įdėti į tenzorių reikšmes: 3,4,1,9

```
const vekt = tf.tensor([3,4,1,9]);
```

Universalumas palengvina darbą programuotojui, bet apsunkina programos skaitymą. Teisingiau būtų naudoti funkcijas, kurios buvo aprašytos tik skaliaro, arba tik vektoriaus, arba matricos kūrimui. Funkcijų pavyzdžiai:*tf.scalar()*, *tf.tensor1d()*, *tf.tensor2d()*.

```
const skal = tf.scalar(9);
const vekt = tf.tensor1d([3,4,1,9]);
const matrix = tf.tensor2d([[1,2,3],[4,5,6],[7,8,9]]);
```

Egzistuoja funkcija atskiram buferiui kurti: *tf.buffer()*. Buferį galima konvertuoti į tenzorių, naudojant funkciją *toTensor()*.[16] Sukurti buferi ir priskirti reikšmes:

```
const buffer = tf.buffer([2, 2]);
buffer.set(3, 0, 0);
buffer.set(5, 1, 0);
```

Konvertuoti buferį į tenzorių:

```
buffer.toTensor().print();
```

Tenzorių kopijavimas su funkcija *tf.clone()*: Sukuria naują tenzorių su tokia pati forma ir reikšmemis:

```
const z = tf.tensor([6, 9]);
z.clone().print();
```

Automatiškai užpildyti tenzorių galima naudojant *tf.fill()*:

```
tf.fill([2, 2], 5).print();
Atsakymas:
Tensor
    [5,5]
    [5,5]
```

1.3.2.2. Kaip kurti modelius naudojant TensorFlow.js ?

Egzistuoja du pagrindiniai modelių kūrimo būdai:

- *Sequential* -– paprasčiausias būdas, bet turi tam tikrų trūkumų.
- *Model* –- sunkesnis būdas, bet siūlo daugiau galimybių.

tf.sequential(config?) funkcija sukuria *sequential* modelį, priima vieną parametrą. Panaudojimo pavyzdys:[16]

Sukuriamas modelis tiesinei regresijai:

```
const model = tf.sequential();
model.add(tf.layers.dense({units: 1, inputShape: [1]}));
model.compile({loss: 'meanSquaredError', optimizer: 'sgd'});
```

Generuojami duomenis modeliui apmokyti:

```
const xs = tf.tensor2d([1, 2, 3, 4], [4, 1]);
const ys = tf.tensor2d([1, 3, 5, 7], [4, 1]);
```

Modelio mokymas ir bandymas su atsitiktine koordinate:

```
await model.fit(xs, ys);
model.predict(tf.tensor2d([5], [1, 1])).print();
Atsakymas:
Tensor
    [[-2.3440001],]
```

tf.model() funkcijos panaudojimo pavyzdys:[16]

Sukuriama įvestis:

```
const inp = tf.input({shape: [3]});
const denseLayer1 = tf.layers.dense({units: 5, activation: 'relu'});
const denseLayer2 = tf.layers.dense({units: 2, activation: 'softmax'});
```

Sukuriama išvestis:

```
const out = denseLayer2.apply(denseLayer1.apply(inp));
```

Sukuriamas modelis:

```
const model = tf.model({inputs: inp, outputs: out});
```

Modelio panaudojimas:

```
model.predict(tf.ones([10, 3])).print();
Atsakymas:
Tensor
    [[0.4423228, 0.5576772],
```

```
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772],
[0.4423228, 0.5576772]]
```

Pagrindinis skirtumas tarp *tf.model()* ir *tf.sequential()* yra tai, kad *tf.sequential()* palaiko tik tiesinį sluoksnių pluoštą. *tf.model()* turi daugiau galimybių.

1.3.3. ML5.js

Siekiant dar labiau supaprastinti darbą su giliojo mokymosi neuronų tinklais ir algoritmais, buvo sukurta aukšto lygio *JavaScript* biblioteka – *ML5.js*. Biblioteka yra lengvai suprantama net tiems žmonėms, kurie neturi gerų programavimo įgūdžių. Vienas iš svarbiausių privalumų yra neuronų tinklų kūrimas neliečiant tenzorių, daugiau nereikia galvoti apie sunkius tenzorinius skaičiavimus, viską atlieka ML5.

ML5 kūrėjai siekė padaryti sistemas ir gilųjį mokymąsi prieinama ne tik programuotojams, bet ir menininkams, studentams ir tiesiog kūrybingiems žmonėms. Biblioteka suteikia prieigą prie visų esamų *TensorFlow.js* apmokytų neuronų tinklų modelių. *ML5 API* yra lengvai suprantamas ir prieinamas būdas visiems, norintiems išbandyti neuronų tinklus savo kompiuteryje. Vienas iš būdų prijungti *ML5* prie *JavaScript* kodo yra panaudoti *CDN*:

```
https://unpkg.com/ml5@0.4.3/dist/ml5.min.js
```

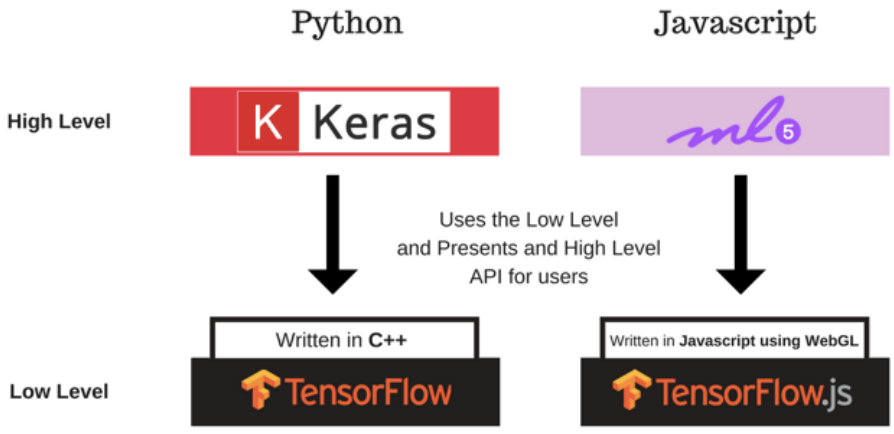
Greito ML5 prijungimo prie *JavaScript* kodo pavyzdys, paveikslėlis numeris 10:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js"></script>
  </head>
  <body>
    <script>
      //Kodas rašomas čia
      console.log('ml5 version:', ml5.version);
    </script>
  </body>
</html>
```

4 pav. JavaScript kodo ML5 bibliotėkos prijungimo pavyzdys.

Kaip jau buvo minėta anksčiau, *ML5* duoda lengvą ir prieinamą kiekvienam taikomąjį programinį interfeisą, leidžianti manipuliuoti *TensorFlow* galimybėmis. Tai reiškia, kad *ML5* naudoja *GPU* galimybes programos veikimui pagreitinti, kaip ir *TensorFlow*. Naršyklės atveju tai buvo įgyvendinta naudojant *WEBGL*.

ML5 veikimas labai panašus į *Keras* bibliotekos veikimą. *Keras* buvo sukurtas naudojimui su *Python* programavimo kalba ir turi tokią ideologiją: supaprastinti darbą su *TensorFlow* ir neuronų tinklais. *TensorFlow* yra labai žemo lygio bibliotekos (angl. low-level). Todėl buvo nuspręsta sukurti tokį aukšto lygio įrankį kaip *Keras*.



5 pav. ML5.js ir Keras palyginimas.[18]

2. Praktinė dalis

2.1. Tikslas

Patobulinti savo žinias neuronų tinklų temoje ir sukurti klasifikavimo modelį, naudojant *ML5.js*. Sukurti tinklalapį, panaudojus *Posenet* ir klasifikavimo neuronų tinklą, įgyvendinti tokį funkcionalumą: programa skaičiuoja žmogaus laiką be raumenų mankštinimo, skaičiuoja kiek laiko jis praleido tam tikrų raumenų mankštinimui. Visus duomenis galima atsiųsti bet kuriuo laiku paspaudus *download* mygtuką.

2.2. ML5.js ir Posenet

Posenet - tai giliojo mokymosi apmokytas neuronų tinklo modelis, kuris gali aptikti žmogaus figūrą vaizdo įraše arba nuotraukoje. Modelis negali pasakyti, koks konkrečiai žmogus yra nuotraukoje arba vaizde, bet jis gali nustatyti, kur yra žmogaus ranka, koja arba galva. Algoritmas išveda masyvą iš 17-os taškų, tai yra 34-ių koordinačių ir skaičių, vadinamų „numatymo balas“ (angl. pre-diction score), kuris rodo tikimybę, kad tai yra būtent ta žmogaus dalis, į kurią modelis rodo. Štai kaip atrodo išvedamas masyvas:[20a]

0. Nosis
1. Kairioji akis
2. Dešinioji akis
3. Kairioji ausis
4. Dešinioji ausis
5. Kairioji petys
6. Dešinioji petys
7. Kairioji alkūnė
8. Dešinioji alkūnė
9. Kairysis riešas
10. Dešinysis riešas
11. Kairioji dubens pusė
12. Dešinioji dubens pusė
13. Kairysis kelis
14. Dešinysis kelis
15. Kairysis kulkšnis
16. Dešinysis kulkšnis

2.3. Paprastų judesių aptikimas

Kursinio darbo tikslas buvo susipažinti su *Posenet* ir panaudoti apmokytą giliojo mokymosi modelį. Sukurti tinklalapį, kuriame žiniatinklio kamera galėtų stebėti žmogaus judesius ir skaičiuoti jų kiekį: rankų pakėlimus į šonus pečių lygyje ir dilbio sukimą ratu.

Posenet pakete yra du neuronų tinklų modeliai. Pirmasis yra skirtas vieno žmogaus aptikimui vaizde, o antrasis – kelių žmonių aptikimui. Pirmasis algoritmas sunaudoja mažiau kompiuterio resursų. Siekiant sumažinti kompiuterio apkrovą, buvo nuspręsta naudoti pirmąjį modelį.

2.3.1. Realizacija

Paprastų judesių aptikimo realizacija prisidėjo nuo *index.html* failo apibrėžimo, bei jau anksčiau minėtos *ML5* aukšto lygio bibliotekos importavimo. Buvo paimta 0.1.3 bibliotekos versija, naudojant tokį *CDN*:

```
https://unpkg.com/ml5@0.1.3/dist/ml5.min.js
```

Darbo palengvinimo tikslais kartu su *ML5* buvo naudojama ir *p5* biblioteka. *p5* buvo naudojama 0.7.2 versija, prijungta naudojant tokį *CDN*:

```
https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/p5.min.js
```

Pridėjus dar reikalingus mygtukus, gautas toks *index.html* kodas:

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/p5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/addons/p5.dom.min.js"></script>
    <script src="https://unpkg.com/ml5@0.1.3/dist/ml5.min.js"></script>
    <meta charset="utf-8" />
  </head>
  <body>
    <div id="control">
      <button id="handRais">Raise your hand</button>
      <button id="forearm">Forearm circle</button>
      <button id="armsAcrossChest">Arms across chest</button>
      <button id="sideBends">Side bends</button>
      <button id="armCicles">Full arm circles</button>
    </div>
    <script src="index.js"></script>
  </body>
</html>
```

6 pav. Kursinio darbo index.html failas.

Taip pat svarbu paminėti, jog buvo atliktas saugumo keitimas. Dabar naudotojų žiniatinklio kameros gali būti pasiekiamos tik naudojant https protokolą. Tai reiškia, kad paleisti *index.html* failą savo kompiuteryje ir naudoti žiniatinklio kameros nėra įmanoma, net per paprastą *HTTP* protokolą žiniatinklio kameros pasiekti negalima. *HTTPS* protokolas su *self-signed* sertifikatais nepadės apeiti šio saugumo.

Kitas realizacijos etapas: nustatymai ir logikos programavimas. Iš pradžių buvo sukurti visi *Posenet* programavimo kintamieji, taip pat aprašyta funkcija *setup()*, kuri *p5.js* bibliotekoje inicijuojama pirmiausiai ir tik vieną kartą. *Setup* funkcijoje nustatoma raiška, sukuriamas *canvas* ir įkeliamas *Posenet* giliojo mokymosi modelis.

```
function setup(){
    createCanvas(640,480);
    video = createCapture(VIDEO);
    video.hide();
    poseNet = ml5.poseNet(video,modelReady);
    poseNet.on('pose',getPoses);
}
```

7 pav. Kursinio darbo setup funkcija.

Kiekvienam judesiui buvo parašyta atskira funkcija, viena iš jų: *singlePersonRaiseHands()*. Funkcija buvo skirta suskaičiuoti vieno žmogaus rankų pakilimus į viršų virš galvos.

```
function singlePersonRaiseHands(poses){
    if(poses[0].pose.keypoints[8].score > 0.2 &&
    poses[0].pose.keypoints[7].score > 0.2){
        let rightElbY = poses[0].pose.keypoints[8].position.y;
        let leftElbY = poses[0].pose.keypoints[7].position.y;
        if(rightElbY < poses[0].pose.keypoints[6].position.y ||
            leftElbY < poses[0].pose.keypoints[6].position.y){
                if( movementFlag === 0){
                    counter += 1;
                    text.innerText = "Raised hand count:" + counter;
                    movementFlag = 1;
                }
        }else{
            movementFlag = 0;
        }
        warningText.style.visibility = "hidden";
    }else{
        warningText.style.visibility = "visible";
    }
}
```

8 pav. Kursinio darbo *singlePersonRaiseHands* funkcija.

Funkcijos logika: pirmiausiai yra tikrinamas spėjimo įvertis. Jeigu tikimybė, kad taškai priklauso žmogui, yra didesne už $0{,}2$, funkcija tikrina žmogaus alkūnių Y koordinates ir palygina su pečių taškų koordinatėmis. Jeigu alkūnių Y reikšmė yra mažesnė už pečių Y koordinatę, fiksuojama, jog ranka yra pakelta. Analogiškai, jeigu alkūnių Y reikšmės didesnės negu pečių, fiksuojama, jog rankos yra nuleistos.

Vienas iš sunkiausių pratimų buvo šoniniai lenkimai. Naudojant tik *Posenet* modelį, buvo sunku sugalvoti algoritmą, kuris galėtų aptikti šoninius lenkimus. Bet pavyko parašyti funkciją *singlePersonSideBends()*, kuri gali, nors ir ne labai tiksliai, aptikti šoninius lenkimus.

```
function singlePersonSideBends(poses){
    if(poses[0].pose.keypoints[6].score > 0.2 &&
    poses[0].pose.keypoints[5].score > 0.2){
        let rightShouldY = poses[0].pose.keypoints[6].position.y;
        let leftShouldY = poses[0].pose.keypoints[5].position.y;
        let noseY = poses[0].pose.keypoints[0].position.y;
        if(noseY > rightShouldY || noseY > leftShouldY){
            if( movementFlag === 0){
                counter += 1;
                text.innerText = "Side bends:" + counter;
                movementFlag = 1;
            }
        }else{
            movementFlag = 0;
        }
        warningText.style.visibility = "hidden";
    }else{
        warningText.style.visibility = "visible";
    }
}
```

9 pav. Kursinio darbo *singlePersonSideBends* funkcija.

Algoritmo logika: programa tikrina spėjimo įvertį. Jeigu įvertis didesnis už $0{,}2$, tikrinama, ar žmogaus nosies taško Y koordinatė yra didesnė už atitinkamą kairiojo arba dešiniojo peties koordinatę. Atitinkamai programa supranta, į kurį šoną buvo padarytas lenkimas.

2.4. Statiniai ir paprasti tempimo pratimai

Kursinio darbo projekto tikslas buvo panaudoti *Posenet* giliojo mokymo modelį statiniams bei paprastiems tempimo pratimams atpažinti. Statiniai pratimai stiprina sveikatą, jų esmė – maksimaliai ištempti raumenis ir išlaikyti tempimo padėtyje. Taip pat buvo naudotos *ML5.js* ir *p5.js* bibliotekos.

2.4.1. Realizacija

Pridėdamos bibliotekos, bet lieka tik du mygtukai. Vienas aktyvuoja tempimo pratimo atpažinimą, antras - statikos atpažinimą.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/p5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/addons/p5.dom.min.js"></script>
    <script src="https://unpkg.com/ml5@0.1.3/dist/ml5.min.js"></script>
    <meta charset="utf-8" />
  </head>
  <body>
    <button onclick="stretchHandBut()">Stretch hands</button>
    <button onclick="freezeBut()">Freeze</button>
    </br>
    <script src="index.js"></script>
  </body>
</html>
```

10 pav. Kursinio darbo projekto *index.html* failas.

Atsižvelgiant į pratimų paprastumą, buvo priimtas sprendimas padidinti vaizdo raišką iki *HD* standarto, tai 720×480 pikselių. Didesnė raiška gerokai apsunkina kompiuterio darbą ir padidina vaizdo ir pagrindinio procesorių apkrovą. Bet didesnė raiška duoda geresnius neuronų tinklo spėjimo rezultatus. Faile *index.js*, funkcijoje *setup()* buvo pakeista raiška.

```
function setup(){
    createCanvas(720,480);
    video = createCapture(VIDEO);
    video.hide();
    poseNet = ml5.poseNet(video,modelReady);
    poseNet.on('pose',getPoses);
}
```

11 pav. Antro kursinio darbo *setup()* funkcija.

Tempimo pratimo atveju buvo parašyta atskira funkcija *maxHandStretchCeck()*. Programa skaičiuoja dabartinį atstumą tarp riešų, jeigu atstumas didesnis negu maksimalus atstumas, kuris buvo užfiksuotas programos veikimo metu - dabartinis atstumas tampa maksimaliuoju. Jeigu dabartinis atstumas tarp riešų yra apytiksliai toks pats, koks ir maksimalus atstumas, programa perspėja naudotoją apie ne iki galo teisingai atliekamą pratimą. Analogiškai, jeigu atstumas yra nepakankamai didelis, programa praneša apie neteisingą pratimo atlikimą.

```
function maxHandStretchCeck(poses){
    let currentDistanceBetweenWrists = Math.abs(
        poses[0].pose.keypoints[keyPositions[5]].position.x
        - poses[0].pose.keypoints[keyPositions[6]].position.x);
    if(currentDistanceBetweenWrists >= MAX_DISTANCE_BETWEEN_WRISTS-50){
        MAX_DISTANCE_BETWEEN_WRISTS = currentDistanceBetweenWrists;
        drawColors = [0,255,0];
    }else if(currentDistanceBetweenWrists >= MAX_DISTANCE_BETWEEN_WRISTS - 100){
        drawColors = [255,140,0];
    }else{
        drawColors = [255,0,0];
    }
}
```

12 pav. Kursinio darbo projekto *maxHandStretchCeck()* funkcija.

Statinio pratimo atpažinimo funkcijos logika: žmogus užima tam tikrą kūno padėtį, programa įsimina žmogaus kūno dalių koordinates ir lygina su naujais gautais taškais. Jeigu naujos koordinatės skiriasi nuo senų, reiškia žmogus pajudėjo ir statinis pratimas nėra atliekamas. Svarbu tai, kad dėl blogo apšvietimo, žiniatinklio kamera gauna daug pašalinės informacijos, todėl tokiomis sąlygomis *Posenet* modelis gali blogiau dirbti. Todėl buvo nuspręsta tikrinti, ar naujos koordinatės patenka į tam tikrą intervalą.

```
function staticPose(poses){
    for(let i = 1; i < 8; i++){
        console.log("X"+ i +":"+ bodypointsX[i-1])
        console.log("Y"+ i +":"+ bodypointsY[i-1])
        if( bodypointsX[i-1] >= poses[0].pose.keypoints[i].position.x -STATIC_ERROR &&
            bodypointsX[i-1] <= poses[0].pose.keypoints[i].position.x +STATIC_ERROR &&
            bodypointsY[i-1] >= poses[0].pose.keypoints[i].position.y -STATIC_ERROR &&
            bodypointsY[i-1] <= poses[0].pose.keypoints[i].position.y +STATIC_ERROR){
            drawColors = [0,255,0];
        }else{
            drawColors = [255,0,0];
            bodypointsX[i-1] = poses[0].pose.keypoints[i].position.x;
            bodypointsY[i-1] = poses[0].pose.keypoints[i].position.y;
            break;
        }
    }
}
```

13 pav. Kursinio darbo projekto *staticPose()* funkcija.

Intervalas skaičiuojamas taip: imamos pirmos koordinatės ir pridedama paklaida. Antro kursinio programoje paklaida buvo lygi 15-ai pikselių. Pavyzdys: koordinatė (50,100), patekimo intervalas bus lygus pagal X ašį [35,65], pagal Y ašį [85,115].

2.5. Pratimų atpažinimas naudojant du neuronų tinklus

2.5.1. Tikslas

Sukurti *WEB* programą, kuri galėtų atpažinti sunkesnius tempimo pratimus, bei skaičiuoti, kiek laiko buvo atliekamas tam tikras pratimas. Viskas turėtų būti skaičiuojama automatiškai, be jokių mygtukų. Taip pat reikalingas toks funkcionalumas: bet kuriuo metu galima atsiųsti tempimo rezultatus *JSON* formatu.

2.5.2. Duomenų surinkimas

Tokio funkcionalumo įgyvendinimui, buvo nuspręsta sukurti dar vieną neuronų tinklą, kuris galėtų dirbti kartu su *Posenet* giliojo mokymosi modeliu. Kaip jau yra žinoma, *Posenet* analizuoja vaizdą ir išduoda 17-ą koordinačių.

14 pav. *Keypoints* masyvas.[20a]

Antras neuronų tinklų modelis turėtų priimti 34-is skaičius (17-a koordinačių) ir pagal juos klasifikuoti žmogaus padėtį vaizde. Kaip ir anksčiau, buvo naudojami *ML5* ir *p5* bibliotekos, skirtos *JavaScript* programavimo kalbai.

Sukuriamas neuronų tinklas su 34-iais įvedimo neuronais ir 6-iais išvesties. Taip pat buvo pasirinkta tokia neuronų tinklo mokymosi užduotis: klasifikavimas. [cha20]

```
let options = {
    inputs: 34,
    outputs: 6,
    task: 'classification',
    debug: true
}
```

Kiekvienas iš išvesties neuronų rodo, kokioje padėtyje dabar yra žmogus. Išvedimo neuronai koduojami taip: Q,W,E,R,T,G.

- Q – kaklo tempimas į dešinę pusę,
- W – kaklo tempimas į kairę pusę,
- E – kaklo tempimas į priekį,
- R – dešinės rankos tempimas už nugaros,
- T – kairės rankos tempimas už nugaros,
- G – ramiai sėdintis žmogus.

Duomenų surinkimo algoritmo kodas buvo panaudotas iš online mokymų *The Coding Train*. [cha20]

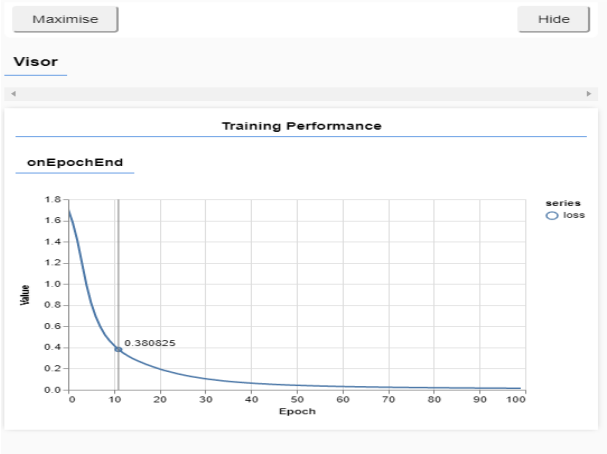
```
if (poses.length > 0) {
  pose = poses[0].pose;
  skeleton = poses[0].skeleton;
  if (state == 'collecting') {
    let inputs = [];
    for (let i = 0; i < pose.keypoints.length; i++) {
      let x = pose.keypoints[i].position.x;
      let y = pose.keypoints[i].position.y;
      inputs.push(x);
      inputs.push(y);
    }
    let target = [targetLabel];
    brain.addData(inputs, target);
  }
}
```

15 pav. Duomenų surinkimo algoritmas.

Algoritmo logika: programa tikrina, ar *Posenet* modelis aptiko nors vieną žmogų vaizde. Jeigu žmogus buvo aptiktas, visos koordinatės yra talpinamos į masyvą. Toks procesas kartojasi kelis kartus ir visos reikšmės išsaugomos kompiuterio faile *PoseData.json*.

2.5.3. Neuronų tinklų modelio mokymas

Neuronų tinklų modelio mokymas užtrūko apie 2 valandas. Kaip jau buvo minėta anksčiau, modelis turi 34-is įvedimo neuronus ir 6-is išvedimo. Buvo nustatyta padaryti 100 epochų. Epocha - tai visų duomenų praleidimas per neuronų tinklą pirmyn ir atgal vieną kartą.[SHA17] Duomenų mokymuisi skirtas failas užima 1398Kb. Paveikslėlyje numeris 22 pavaizduoti mokymosi proceso rezultatai.



16 pav. Neuronų tinklo mokymosi proceso rezultatai.

2.5.4. Naujo neuronų tinklo įkėlimas ir paruošimas darbui

Prieš panaudojant sukurtą neuronų tinklų modelį, reikia aprašyti reikiamus kintamuosius ir parametrus:

```
let neuralNet;
let parameters = {
    inputs: 34,
    outputs: 6,
    task: 'classification',
    debug: true
};
```

Taip pat reikia nurodyti kelią iki apmokyto modelio:

```
const modelInfo = {
    model: 'myModel/model.json',
    metadata: 'myModel/model_meta.json',
    weights: 'myModel/model.weights.bin',
};
```

Galutinai, setup() funkcijoje reikia perduoti visus parametrus ir įkelti modelį:

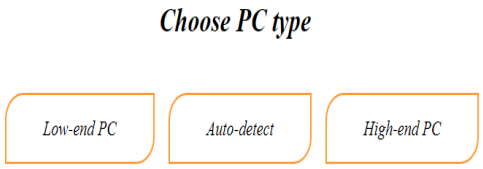
```
neuralNet = ml5.neuralNetwork(parameters);
neuralNet.load(modelInfo, neuralNetLoaded);
```

Dabar modelis yra paruoštas naudojimui ir gali klasifikuoti žmogaus padėtį.

2.5.5. Funkcionalumo įgyvendinimas

Baigiamojo darbo programa sudaryta iš dviejų variantų. Pirmas variantas *low-end* yra skirtas silpnesniems kompiuteriams, neturintiems greitų *GPU* ir *CPU*. Toks variantas gerai tinka

programos naudojimui foniniame režime, tokiu atveju didelio kompiuterio apkrovimo nėra ir galima laisvai daryti savo kasdienius darbus. Antras variantas *high-end* skirtas našiems ir greitiems kompiuteriams. Programa veikia daug greičiau, per vieną sekundę padaro 30 kartų daugiau skaičiavimų negu *low-end* variantas. Žinoma, šiuo atveju kompiuterio apkrova ženkliai išauga. Jeigu naudotojas nežino savo kompiuterio specifikacijų, jis gali paspausti mygtuką *Auto-detect* ir programa pati pasirinks atitinkamą variantą.



17 pav. Varianto pasirinkimas.

Turint apmokytą ir prijungtą klasifikavimo neuronų tinklą, galima klasifikuoti žmogaus padėtį vaizde. Pirmiausiai, reikia surinkti duomenis iš *Posenet* giliojo mokymosi neuronų tinklo.

```
function classifyFrame() {
  if(pose){
    let inputs = [];
    for (let i = 0; i < pose.keypoints.length; i++) {
      inputs.push(pose.keypoints[i].position.x);
      inputs.push(pose.keypoints[i].position.y);
    }
    neuralNet.classify(inputs, getResult);
  }else {
    setTimeout(classifyFrame, 100);
  }
}
```

18 pav. Duomenų surinkimas ir perdavimas neuronų tinklui.

Funkcija *classifyFrame()* tikrina, ar *Posenet* modelis aptiko žmogų vaizde. Jeigu žmogus buvo aptiktas, algoritmas įrašo visus pagrindinius taškus į atskirą masyvą. Kai masyvas buvo užpildytas, jis perduodamas antram neuronų tinklų modeliui.

```
function getResult(error, results) {
  if (results[0].confidence > 0.75) {
    poseLabel = results[0].label.toUpperCase();
    workoutCheck(poseLabel);
  }
  classifyFrame();
}
```

19 pav. Funkcija *getResult*.

Funkcija *getResult()* tikrina klasifikavimo tikslumą, jeigu tikslumas didesnis už 0,75 – algoritmas perduoda rezultatus *workoutCheck()* funkcijai.

```
function workoutCheck(poseLabel){
    switch(poseLabel){
        case "G":
            if(timerFlags[0] == 0){
                saveTime(timerFlags);
                dropTimer();
                dropFlags();
                timerFlags[0] = 1;
                startNewTimer("G");
            }
            timeSign.innerText = "Time without warmup: " + getTimerDifference(0);
            checkTimerAlert(getTimerDifference(0,1));
            break;
```

20 pav. Dalis funkcijos *workoutCheck()*.

Funkcijos *workoutCheck* pagrindas yra *switch-case* konstrukcija, kuri suteikia galimybę patogiai aprašyti algoritmą. Kaip jau buvo minėta anksčiau, antras neuronų tinklas rezultate duoda tam tikrą raidę, kuri reiškia tam tikrą žmogaus poziciją vaizde. Algoritmas priima raidę ir aktyvuoja skirtingus laikmačius, skirtus kiekvienam iš pratimų atskirai. Visa informacija apie laikmačius yra kaupiama, ir paspaudus mygtuką *Download* galima išsaugoti visus savo rezultatus.

Taip pat programa, atsižvelgiant į atliekamą pratimą, pratimo pavadinimą nuspalvina žaliai. Jeigu žmogus neatlieka jokio pratimo, laikmatis skaičiuoja ir fiksuoja, kiek laiko praleista be raumenų tempimo. Jeigu laikas bus didesnis nei viena valanda, programa perspėja naudotoją, kad atėjo laikas atlikti tempimo pratimus.

2.5.6. Ar įmanoma tokį funkcionalumą įgyvendinti kitaip?

Pavyzdžiui, mes neturim antro neuronų tinklo modelio, kuris naudodamas kūno taškų koordinates gali klasifikuoti žmogaus padėtį. Ar įmanoma būtų sukurti tokią programą naudojant tik *Posenet* modelį? Vienas iš įmanomų sprendimų – trikampių panašumo požymių ieškojimas. Darant šoninį kaklo lenkimą: petys, alkūnė ir riešo taškų koordinatės sudaro trikampį.

Galima išsaugoti tokį trikampį ir ieškoti trikampių panašumo požymių tarp iš anksto išsaugoto ir realiu laiku gaunamų trikampių. Egzistuoja trys trikampių panašumo požymiai: pagal du kampus, pagal dvi kraštines ir kampą tarp jų, pagal tris kraštines. Jeigu programa rado panašų trikampį, galima teigti, kad žmogus atlieka šoninį kaklo lenkimą.

Apibendrinant, lengvesnis būdas sukurti žmogaus pratimų atpažinimo programą – sukurti, apmokyti ir panaudoti klasifikavimo neuronų tinklų modelį. Toks metodas užtikrina greitą ir teisingą programos veikimą.

Išvados

Laikui bėgant, *IT* sritis tampa vis svarbesne žmogaus gyvenime. Gilusis ir sistemų mokymasis tapo neatskiriama progreso dalimi. Kuo kompiuteriai greitesni ir stipresni, tuo geriau veikia dirbtinis intelektas.

Posenet giliojo mokymosi neuronų tinklų algoritmas suteikia galimybę atpažinti žmogaus figūrą vaizde. Kartu su turinčia daug galimybių ML5.js biblioteka tai leidžia realizuoti daugybę kūrybinių idėjų. ML5.js biblioteką galima panaudoti kartu su TensorFlow.js biblioteka ir kurti sunkius neuronų tinklus, kurie veiks naršyklėje. Šie tinklai gali veikti greitai ir sklandžiai, nes WEBGL leidžia optimizuoti jų veikimo greitį.

Šiame darbe buvo išnagrinėti tokie giliojo mokymosi neuronų tinklų kūrimo įrankiai kaip *ML5* ir *TensorFlow.js*. Parodyta ir paaiškinta kaip kurti skirtingo sudėtingumo tenzorius, kaip atlikti paprastus veiksmus su tenzoriais ir kaip naudoti jau apmokytą giliojo mokymosi neuronų tinklų modelį. Buvo išanalizuoti klasifikavimo modelių galimybės, bei sukurtas vienas modelis naudojant *ML5* įrankį.

Dirbtinio intelekto sritis auga ir tai duoda dideles galimybes panaudoti giliojo mokymosi algoritmus savo kūrybinėms idėjoms įgyvendinti. Gilusis mokymas gali pakeisti daugumą dabar egzistuojančių technologijų.

Santrauka

Dirbtinio intelekto sritis auga ir tai suteikia daug naujų galimybių realizuoti anksčiau neįmanomą funkcionalumą. Šiame darbe buvo aptarti ir panaudoti keli giliojo mokymosi įrankiai, bei parodyta, kaip galima panaudoti jau sukurtą ir apmokytą neuronų tinklų modelį. Taip pat buvo sukurtas jogos pratimų atpažinimui ir jų rezultatų surinkimui bei apdorojimui skirtas tinklalapis. Jogos pratimai mažina nugaros skausmus ir suteikia kokybišką atsipalaidavimą. Sukurta programa padės nepamiršti apie mankštinimo pratimus.

Raktiniai žodžiai: *gilusis mokymasis, sistemos mokymasis, klasifikavimo modeliai, neuronų tinklas*.

Summary

Artificial intelligence field is growing fast and this provides a large amount of opportunities to realize previously impossible functionality. In this work, several deep learning tools were discussed and used, it was clarified how to work with tensors and how the already developed and trained neural network model could be used. Also, website was developed to identify yoga exercises and calculate results. Yoga exercises reduces back pain and provides relaxation. The conceived software can help a potential user not to obliterate physical activity.

Key words: *deep learning, machine learning, clasification models, neural network.*

Literatūra

[16] Tensorflow.js api. https://js.tensorflow.org/api/latest/, 2016. Accessed 2020.05.

[18] Introduction to ml5.js. https://towardsdatascience.com/introduction-to-ml5-js-3fe51d6a4661, 2018. Accessed 2020.05.

[20a] Pose estimation. https://www.tensorflow.org/lite/models/pose_estimation/overviewhttps://creativecommons.org/licenses/by/4.0/, 2020. Accessed 2020.05.

[20b] Tensorflow.js setup. https://www.tensorflow.org/js/tutorials/setuphttps://creativecommons.org/licenses/by/4.0/, 2020. Accessed 2020.05.

[cha20] The Coding Train channel. Ml5.js: pose classification with posenet and ml5.neuralnetwork(). https://www.youtube.com/watch?v=FYgYyq-xqAw&t=141s, 2020. Accessed 2020.03.

[GR18] Josh Gordon and Sara Robinson. Introducing tensorflow.js: machine learning in javascript. https://medium.com/tensorflow/introducing-tensorflow-js-machine-learning-in-javascript-bf3eab376db, 2018. Accessed 2020.05.

[Mah18] Sambit Mahapatra. Why deep learning over traditional machine learning? https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063, 2018. Accessed 2020.05.

[Mar19] Bernard Marr. 10 amazing examples of how deep learning ai is used in practise. https://www.forbes.com/sites/bernardmarr/2018/08/20/10-amazing-examples-of-how-deep-learning-ai-is-used-in-practice/#73c3a684f98a, 2019. Accessed 2020.05.

[SHA17] SAGAR SHARMA. Epoch vs batch size vs iterations. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9, 2017. Accessed 2020.05.

Priedai

There are 2 ways to use this program:

First way

Use program online, it was uploaded on VU MIF server:

- Open link: https://karklas.mif.vu.lt/alku3851/Bakalauras/index.html

Second way

If you want to use this program on your local machine, here are the steps you need to follow:

- Make sure you have installed Python 3 on your device.
- Download project from git: https://git.mif.vu.lt/alku3851/Bakalauras
- Open Terminal/CMD (depends on the system).
- Go to the project folder: cd /path/to/project/Bakalauras
- Run the localhost server: python -m http.server
- In Google Chrome open: localhost:8000