Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Skaitinio intelekto metodai (T125B117)

Kursinis darbas

Aplinkos garsų klasifikavimas naudojant konvoliucinį neuroninį tinklą

Žygimantas Marma EVS 8/1
Studentas

Prof. Vidas Raudonis
Dėstytojas

Kaunas, 2022

Turinys

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

DNT - dirbtinis neuroninis tinklas.

ESC - angl. Environmental Sound Classification

MFCC – angl. Mel Frequency Cepstral Coefficients

CRP - angl. Cross Recurrence Plot

ReLU – angl. Rectified Linear Unit

ESC – angl. Environmental Sound Classification

KNT – konvoliucinis neuroninis tinklas

Terminai:

Fonema – mažiausias kalbos vienetas, turintis skiriamąją reikšmę.

Įvadas

Gyvename pasaulyje, apsuptame skirtingų garsų iš skirtingų šaltinių. Mūsų smegenys kartu su klausos sistema nuolat dirba identifikuojant kiekvieną girdimą garsą evoliucija paremtu optimaliu būdu. Be to, mūsų smegenys nuolat apdoroja gautus garso signalus ir suteikia mums atitinkamų žinių apie supančią aplinką. Akivaizdu, kad žmonės gali lengvai atskirti garsus, tačiau kompiuterizuotoms sistemoms ši užduotis nėra tokia paprasta. Nors buvo bandymų skirtingų algoritmų pagalba sukurti išmaniuosius įrenginius, kurie galėtų išgauti reikiamą informaciją iš garso signalo, tačiau smegenų tikslumo lygio pasiekti mokslininkams vis dar nepavyksta. Būtent pastarąją garsų klasifikavimo problemą ir nagrinėsime šiame darbe.

Garso klasifikavimas – tai garso įrašų klausymosi ir analizės procesas. Šis procesas, taip pat žinomas kaip audio signalų klasifikavimas, yra pagrindas daugelių šiuolaikinių DI technologijų, tokių kaip: virtualieji asistentai, automatinės kalbos atpažinimo sistemos ir teksto į kalbą vertimo aplikacijos. Garsų klasifikavimas jau daugelį metų yra didelės svarbos tyrimų sritis. Šioje srityje buvo išbandyta daug įvairių metodų su skirtingais modeliais ir funkcijomis, kurie pasirodė esą naudingi ir tikslūs. Taip pat garsų klasifikavimą galima rasti išmaniųjų namų apsaugos sistemose, kalbos vertimo realiu laiku iš vienos į kitą, įsibrovėlių aptikimo laukinės gamtos zonose ir garso stebėjimo srityse.

Aplinkos garsų klasifikavimas yra viena iš svarbiausių problemų garsų atpažinimo srityje. Palyginus su įprastais ir struktūriškais garsais, tokiais kaip kalba ar muzika, aplinkos garsai neturi nei statinių laiko modelių, kaip melodijos ar ritmai, nei semantinių sekų, kaip fonemos. Todėl sunku rasti universalių bruožų, galinčių reprezentuoti įvairių tamprų modelius. Be to, aplinkos garsuose yra daug triukšmo ir pašalinių garsų nesusijusių su nagrinėjamu. To pasėkoje susidaro sudėtinga kompozicijos struktūra su nepastovumu, įvairumais ir nestruktūrizuotomis savybėmis. Siekiant išspręsti ankščiau išvardytas problemas, nekalbių aplinkos signalų klasifikavimo (ESC) užduotims atlikti buvo naudojami įvairūs signalų apdorojimo metodai ir mašininio mokymosi metodai.

Pastaraisiais ESC tyrimams, kurie daugiausia skirti tokiems garso įvykiams kaip šunų lojimo, ginklų šūvių, stiklo dužimo ar žmogaus griuvimo garsų atpažinimui, sulaukia vis daugiau dėmesio. Tyrimo rezultatai buvo panaudoti daugelyje praktinių pritaikymų, įskaitant robotizuotą klausą, išmaniuosius namus, garso stebėjimo sistemą garso kraštovaizdžio vertinimui. Modeliai taip pat naudojami apsaugos sistemose aptikti tokius garsus kaip dūžtantį stiklą. Pramonėje naudojamas atsitiktiniams nelaimingiems atsitikimams, nustatant garso neatitikimus gamyklinėse mašinose. Modeliai netgi naudojami atskirti gyvūnų poravimosi garsus ir taip saugoti laukinę gamtą. Taip pat tobulėjant robotikos galimybės kuriami išmanūs slaugos robotai galintys reaguoti į žmogaus nugriuvimą ir suteikti arba bent iškviesti pagalbą.

Svarbu paminėti kad, projektų, susijusių su garso klasifikavimu, duomenų rinkinio kokybė gali nulemti ir dažniausiai nulemia projekto rezultatų kokybę. Todėl, norint užtikrinti tikslų garso klasifikavimo lygį, mums reikės daug aukštos kokybės, tiksliai anotuotų duomenų. Todėl šiame darbe eksperimentams pasirinkome viešą duomenų rinkinį – UrbanSound8K.

Galiausiai šiame projekte mes pasiūlysime konvoliucinio neuroninio tinklo modelį (KNT), kurio dėka būtų galima klasifikuoti nekalbius garso signalus naudojant Mel’o spektrogramas.

1. Literatūros apžvalga

1.1 Muzikos žanrų klasifikavimas (Derek A. Huang)

Darbe [1] mokslininkai klasifikavo muzikos žanrus įvairiais metodais. Buvo atlikti eksperimentai su RBF branduolio atraminių vektorių mašina (angl. RBF Kernel Support Vector Machines), k-artimiausių kaimynų metodu, paprastu tiesinio skleidimo (angl. basic feed-forward) tinklu ir galiausiai pažangiu konvoliuciniu neuroniniu tinklu.

Treniravimui buvo naudotas GTZAN duomenų rinkinys [2]. Šis duomenų rinkinys susideda iš tūkstančio 30 sekundžių garso klipų, kurie visi yra pažymėti kaip vienas iš 10 galimų žanrų ir pateikti .au failų formatu. Mokslininkai taip pat naudojo Mel‘o spektrogramas duomenų analizei. Galiausiai buvo prieita išvados, kad tinkamiausias yra konvoliucinis neuroninis tinklas. Šio tipo tinklas pasiekė 82% tikslumą su testavimo duomenimis (naudojant apdorotus duomenis), kai kiti metodai pasiekė 54% bei 60% tikslumą.

Table 1: Accuracy of predictions by model used.

	With data processing			Without data processing		
	Train	CV	Test	Train	CV	Test
Support Vector Machine	.97	.60	.60	.75	.32	.28
K-Nearest Neighbors	1.00	.52	.54	1.00	.21	.21
Feed-forward Neural Network	.96	.55	.54	.64	.26	.25
Convolution Neural Network	.95	.84	**.82**	.85	.59	.53

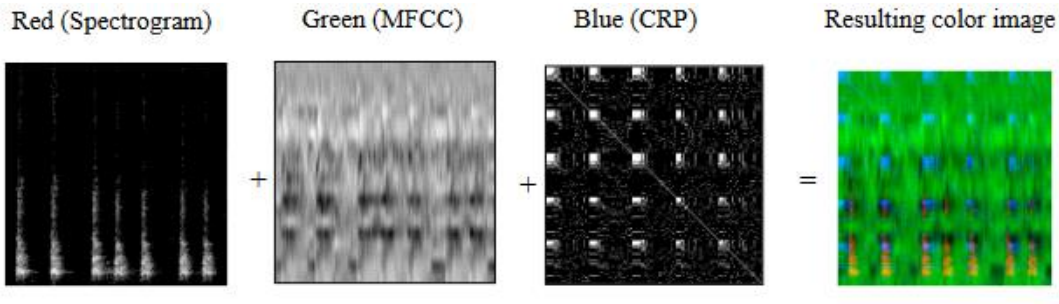
1 pav. Mokslinio darbo testavimo rezultatai

1.2 Aplinkos garsų klasifikavimas naudojant vaizdus atpažinančius tinklus (Venkatesh Boddapatia)

Darbe [3] yra naudojami gilieji konvoliuciniai neuroniniai tinklai („AlexNet“ [4] ir „GoogLeNet“ [5]) vaizdų atpažinimui nekalbių signalų klasifikavimui. Buvo dirbama su ESC-10, ESC-50 ir UrbanSound8K garsų duomenų rinkiniais. Kadangi ESC duomenų bazės [6] yra ganėtinai mažos (ESC-50 sudaryta iš 2000 penkių sekundžių įrašų, o ESC-10 iš 400 įrašų) todėl mokymosi duomenų bazė buvo praplėsta. Nauji garso įrašai, gauti modifikuojant pradinius duomenis įvairiais metodais: įtraukiant atsitiktinius vėlavimus, audio signalo laikas prailginamas, garso įrašų pakeičiamas greitis – jie pagreitinami arba sulėtinami tam tikru koeficientu. Šiame darbe yra palyginamos skirtingos vaizdinių rūšys (spektrogramos, Mel‘o dažnio cepstraliniai koeficientai (MFCC) ir kryžminiai pasikartojimo grafikai (CRP)) siekiant naudojant vaizdus gauti geriausią garsinių signalų klasifikavimo tikslumą.

Mokslininkai įrodė, kad gilieji konvoliuciniai neuroniniai tinklai, specialiai sukurti objektų atpažinimui vaizduose, gali būti sėkmingai išmokyti klasifikuoti aplinkos garsų spektrinius vaizdus. Geriausias klasifikavimo tikslumas ESC-50, ESC-10 ir UrbanSound8K duomenų rinkiniuose buvo atitinkamai 73 %, 91 % ir 93 % naudojant GoogLeNet. Daugeliu atvejų, kuriuos ištyrė mokslininkai, „GoogLeNet“ tinklas klasifikavimo tikslumas buvo didesnis nei „AlexNet“. Jų manymų to pagrindinė priežastis yra ta, kad „GoogLeNet“ yra daug gilesnis nei „AlexNet“ (22, palyginti su 8 sluoksniais). Taip pat autoriai nustatė, kad kelių

skirtingų vaizdo rūšių kombinavimas (spektrogramos, MFCC, ir CRP) kaip to paties vaizdo skirtingų spalvų kanalų nepagerino klasifikavimo tikslumo.

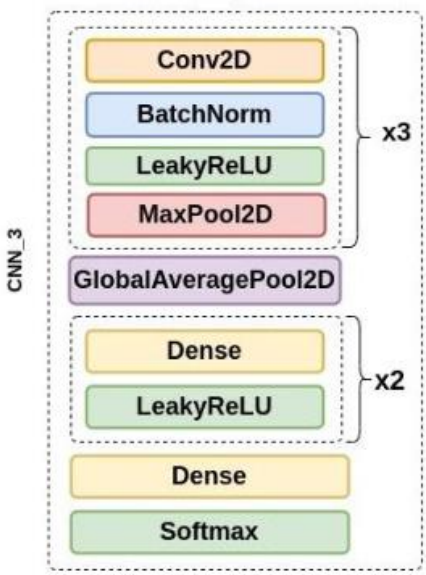


2 pav. Darbe naudojamas trijų tipų vaizdų derinys, norint gauti vienos spalvos vaizdą.

1.3 Buitinių veiksmų garsų atpažinimo identifikavimo sistemos sukūrimas (Artem Sazonov)

Šiame darbe yra pateikiamas garso įrašų apdorojimo metodas, kuris suskirsto aplinkos garsus į vieną iš 11 buitinės veiklos klasių. Mokslininkų aprašytas metodas pagrįstas bruožų išskyrimu iš Mel'o kepstrumo į 224 pikselių pilkos spalvos kvadratinį vaizdą. Šie vaizdai buvo klasifikuojami naudojant 3 sluoksnių konvoliucinį neuroninį tinklą.

Geriausiu atveju buvo pasiektas 92,60 % atpažinimo tikslumas naudojant DASEE duomenų bazę. Šie rezultatai pranoko 1 dimensijos konvoliuciniu modeliu pagrįstus metodus, kuriuose naudojami neapdoroti garso signalo duomenys. DASEE duomenų bazės [7] tarpklasių disbalanso problema buvo išspręstas taikant klasės kodavimo etikečių išlyginimą (angl. labels smoothing) ir klasių svėrimą kategoriškoje kryžminės entropijos (angl. categorical cross-entropy) praradimo funkcijoje.

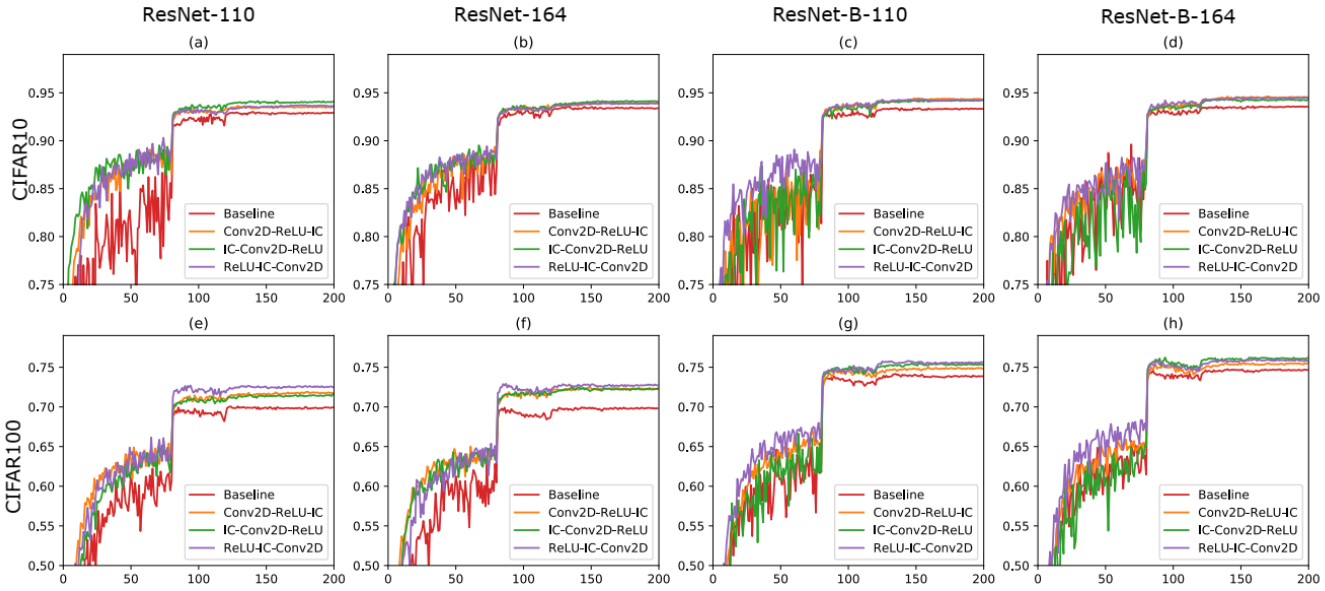


3 pav. Darbe naudota neuroninio tinklo struktūra

1.4 Partijos normalizavimo ir išmetimo technikų analizė naudojant giliuosius neuroninius tinklus (Guangyong Chen)

Šiame darbe [8] mokslininkai siūlo naują techniką, skirtą padidinti neuroninio tinklo mokymo efektyvumui. Jų darbas grindžiamas idėja, kad apdorojus neuroninių tinklų įvestis galima pasiekti greitesnę tikslumo konvergenciją. Pasiekti šiems rezultatus mokslininkai analizavo partijos normalizavimo ir išmetimo technikų naudojimą. Autoriai teigia, kad neturėtume dėti partijos normalizavimo prieš ReLU aktyvacijos funkciją, nes dėl neneigiamų ReLU funkcijos išėjimų svorio sluoksnis bus atnaujintas neoptimaliu būdu. Tačiau mokslininkai, rašo, kad jie gali pasiekti geresnį našumą derindami partijos normalizavimą ir išmetimą kaip nepriklausomų komponentų sluoksnį.

Atlikus bandymus su CIFAR10/100 duomenų rinkiniais [9], kaip parodyta 4 pav., tradicinis partijos normalizavimo naudojimas vis dar lemia nestabilų optimizavimo procesą, palyginti su autorių įgyvendintu sprendimu. Todėl, autoriai abejoja įprasta praktika, kai partijos normalizavimas yra naudojamas prieš aktyvinimo sluoksnį.



4 pav. „ResNet“ ir „ResNet-B“ testavimo tikslumo su nepriklausomų komponentų sluoksniu bandymų palyginimas

Galiausiai šiame straipsnyje mokslininkai pateikia kitokį požiūrį tradicinis partijos normalizavimo ir išmetimo naudojimui treniruojant DNT ir nustatė, kad jie turėtų būti sujungti kaip bendras sluoksnis. Šių technikų kombinacija turėtų būtų paversta nepriklausomais komponentais, ir įdėti šį sluoksnį tiesiai prieš svorio sluoksnį. Performuluodami „ResNets“ tinklą [10] su nepriklausomais komponentais, autoriai pasiekė stabilesnį mokymo procesą, greitesnį konvergavimą ir geresnį apibendrinimo našumą. Autoriai siūlo, kad ateityje turėtume apsvarstyti galimybę įtraukti kitus pažangesnius normalizavimo metodus. Tokius kaip sluoksnių normalizavimas [12] egzempliorių normalizavimas [13] grupės normalizavimas [14] ir kitus pažangesnius statistinius metodus nepriklausomų komponentų sluoksniais sukurti.

1.5 Environmental sound classification using temporal-frequency attention based convolutional neural network (Wenjie Mu)

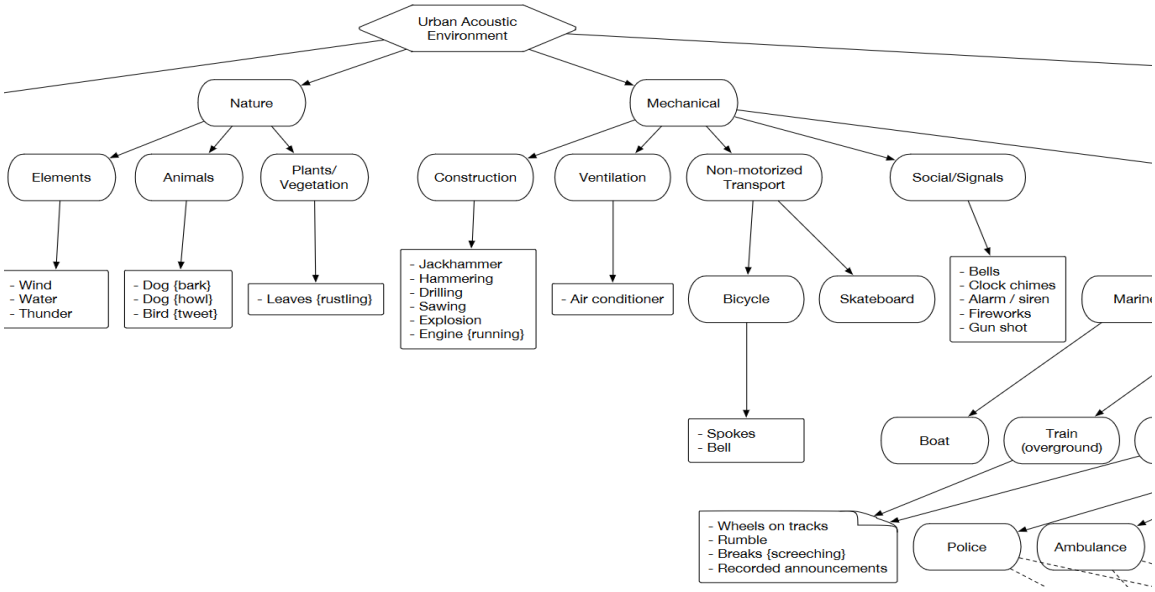
Šiame darbe [15] aplinkos garso klasifikavimui siūlomas naujas laiko dažniu dėmesiu pagrįstas konvoliucinio neuroninio tinklo modelis (TFCNN). Įdiegus sukurtą laiko ir dažnių dėmesio mechanizmą pagrindinėje KNT architektūroje, vaizdinių mokymuisi naudojami skaičiavimai gali būti sutelkti į konkrečias sritis, kuriose yra diskriminacinė informacija, taip efektyviai fiksuojant kritines laiko ir dažnio ypatybes.

Eksperimentai su UrbanSound8K ir ESC-50 duomenų rinkiniais parodė, kad modelio klasifikavimo tikslumas yra didesnis atitinkamai 93,1 % ir 84,4 %. Palyginti su ankstesniais šio duomenų rinkinio modeliais, mokslininkų modelis pasižymi nesudėtinga tinklo struktūra ir paprastų funkcijų apdorojimo privalumai, tuo pačiu užtikrinant tikslumą. Be to, šiame darbe vertinamas modelio klasifikavimo veikimas pagal kelis skirtingus dėmesio mechanizmus ir aptariamas jų poveikis kiekvienam garso įvykiui. Ateityje mokslininkai planuoja ir toliau optimizuoti svertinio derinimo strategiją, atsižvelgdami į skirtingų garso įvykių tipų priklausomybės nuo laiko ir dažnio ypatybių laipsnį, o tada pasirinktinai nustatyti šiai kategorijai tinkamus sintezės parametrus, kad dar labiau pagerintume modelio našumą.

2. Matematinis siūlomo sprendimo pagrindimas

2.1 Analizuojamas duomenų rinkinys

Aplinkos garsams klasifikuoti, kaip minėjome anksčiau, naudosime UrbanSound8K duomenų bazę. Šiame duomenų rinkinyje yra 8732 pažymėtos miesto aplinkos garsų ištraukos (kurių trukmė neilgesnė nei 4s) iš 10 klasių: oro kondicionierius, automobilio garsinis signalas, vaikų žaidimo šurmulys, šuns lojimas, gręžimas, variklio darbas tuščiąja eiga, ginklo šūvis, gręžimo kūjo garsai, sirenos garsas ir gatvės muzika. Klasės sudarytos iš miesto garso taksonomijos 5 pav. Išsamų duomenų rinkinio ir jo sudarymo aprašymą rasti moksliniame darbe [11].



5 pav. Urbansound8k duomenų rinkinio taksonomijos iškarpa

Rinkinyje visos ištraukos paimtos iš įrašų, įkeltų į www.freesound.org tinklalapį. Failai yra iš anksto surūšiuoti į dešimt aplankalų (aplankai, pavadinti fold1-fold10), kad būtų lengviau atkurti ir palyginti su automatinio klasifikavimo rezultatais, aprašytais aukščiau minėtame straipsnyje [11]. Duomenų rinkinyje taip pat pateikiami metaduomenys (2.1 lentelė) kiekvienam garso įrašui, kurių pagalba galime lengviau naudotis duomenų baze.

2.1 lentelė. Urbansound8k metaduomenų fragmentas

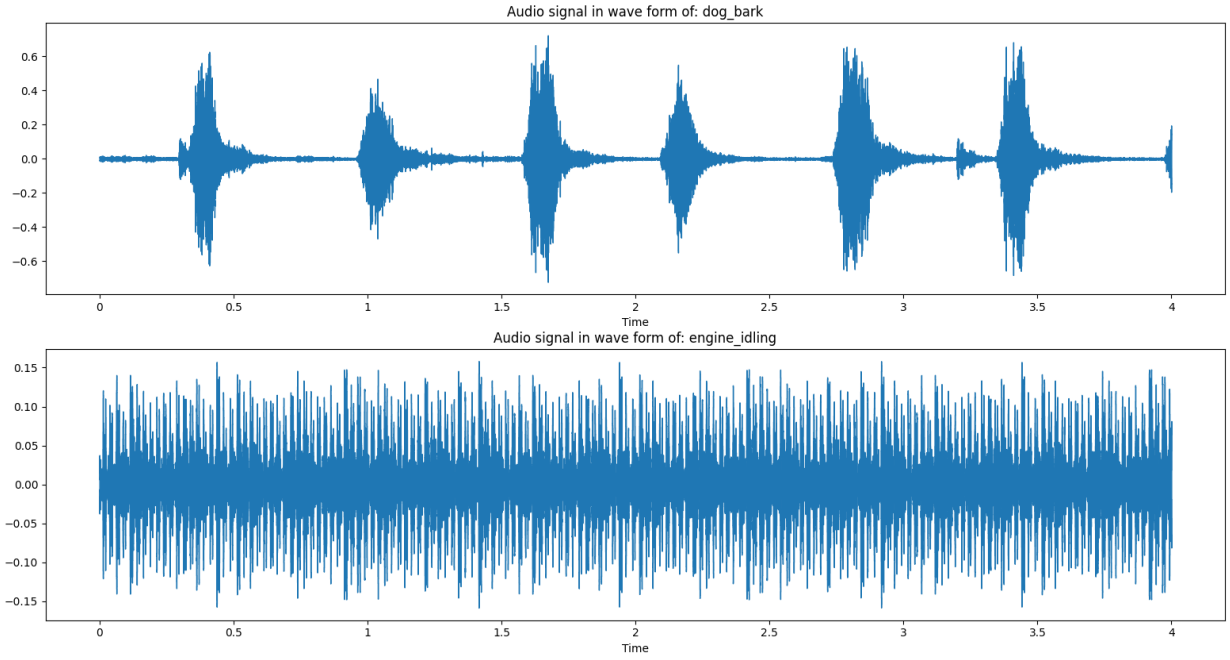
slice_file_name	fsID	start	end	salience	fold	classID	class
100032-3-0-0.wav	100032	0	0.317551	1	5	3	dog_bark
100263-2-0-117.wav	100263	58.5	62.5	1	5	2	children_playing
100263-2-0-121.wav	100263	60.5	64.5	1	5	2	children_playing
100263-2-0-126.wav	100263	63	67	1	5	2	children_playing
100263-2-0-137.wav	100263	68.5	72.5	1	5	2	children_playing
100263-2-0-143.wav	100263	71.5	75.5	1	5	2	children_playing

100263-2-0-161.wav	100263	80.5	84.5	1	5	2	children_playing

Norėdami pamatyti savo garso signalus grafine formą pasirašome Python funkciją. Naudodami „Librosa“ biblioteką pasiekiame diske esančius .wav formato failus ir nubrėžiame signalo amplitudės priklausomybes nuo laiko dviem atsitiktiniam įrašam.

```
def plot_wave_from_audio(df, base_path):
  for j in range(1, 3):
    i = np.random.randint(0, 8732)
    path = base_path  + "//audio//fold" + str(df["fold"][i]) + '//' + df["slice_file_name"][i]
    data, sr = librosa.load(path)
    plt.subplot(2, 1, j)
    librosa.display.waveshow(data, sr=sr, x_axis='time', offset=0.0, marker='', where='post')
    plt.title("Audio signal in wave form of: " + str(df["class"][i]))
  plt.show()
```

6 pav. Funkcija nubrėžti signalo amplitudės priklausomybes nuo laiko



7 pav. Audio signalų amplitudės priklausomybes nuo laiko, viršuje šuns lojimas, apačioje mašinos variklio ūžimas

2.2 Bruožų išgavimas iš audio failo

Yra trys pagrindiniai būdai, kaip išgauti bruožus iš garso signalo:

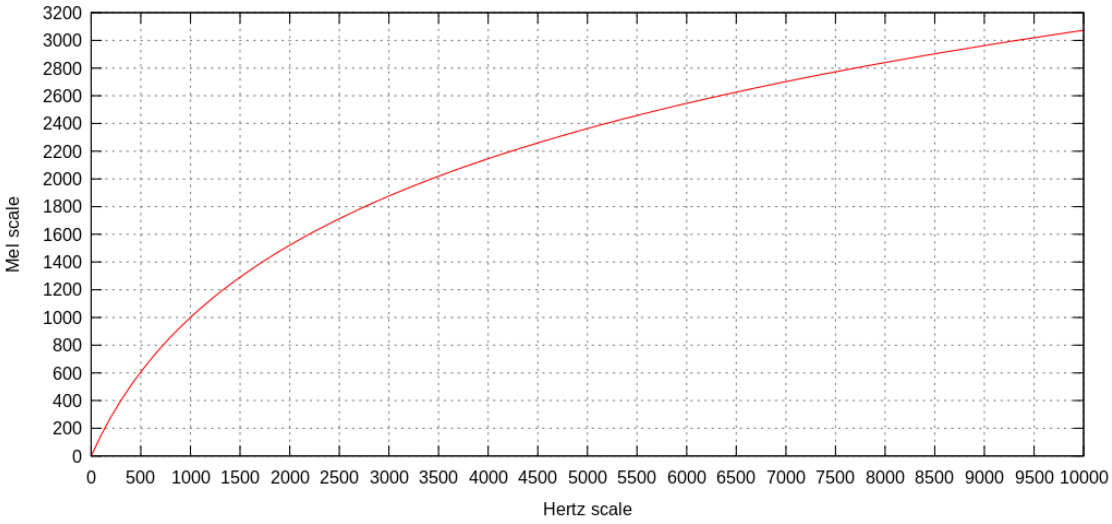
1) Naudojant garso failų MFCC ar kitų bruožų duomenis.

2) Garso spektrogramų vaizdo naudojimas. Turint vaizdus, kaip įvestis į neuroninį tinklą, vienas iš pranašumų yra tai, kad galima naudoti konvoliucinius sluoksnius. To pasėkoje įmanomas geresnis modelio tikslumas.
3) Naudoti keletą bruožų išgavimo technikų ir jas kombinuoti į vieną paveikslėlį.

Šiame darbe nagrinėsime garsų apžinimą naudojantis Mel‘o sprektogramomis (angl. Mel spectrogram). Pirmiausia Mel‘o skalė yra garso dažnio (arba natų) suvokimo skalė, kuriuos klausytojai vertina kaip esančius vienodais atstumais vienas nuo kito. Ši skalė buvo įvesta, kadangi žmonių klausa yra jautresnė garsų pasikeitimui žemų dažnių zonoje nei aukštųjų. Atskaitos taškas tarp šios skalės ir normalaus dažnio matavimo nustatomas priskiriant 1000 mels suvokimo aukštį lygų 1000 Hz tonui. Pastebime kad, viršijus 500 Hz, vis didesni intervalai klausytojų yra vertinami tarsi tolygūs padidėjimai. Nors oficialios formulės pakeisti dažnį į melus nėra, populiariausiai pripažinta yra formulė iš O'Shaughnessy's knygos [16]:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$

Šią išraišką atvaizduojame grafiškai 8 pav.

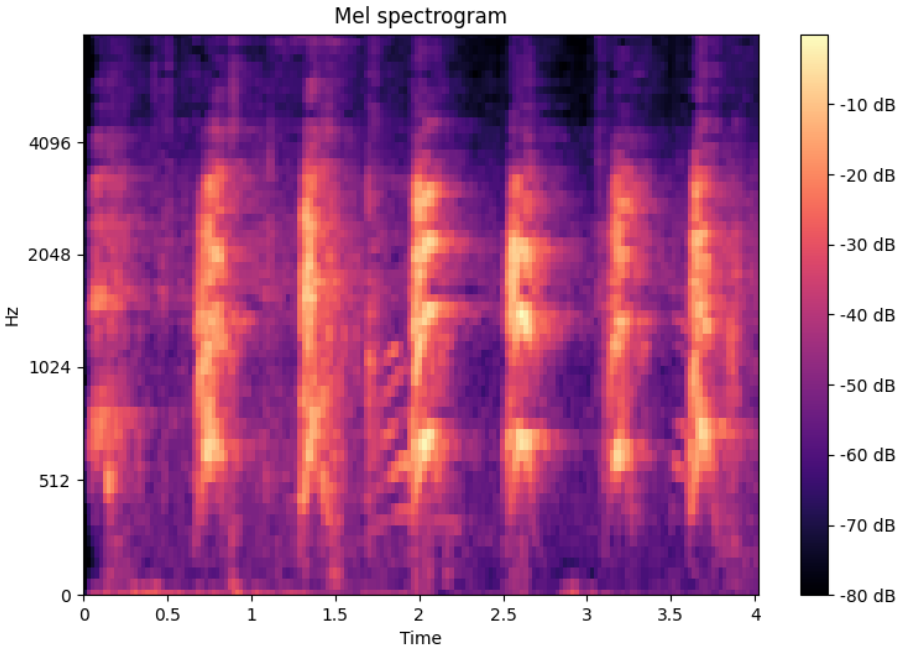


8 pav. Mel‘o skalės priklausomybė nuo Hertzo skalės

Suprasti kokie vaizdai susidaro garso signalui pritaikant Mel‘o spektrogramas parašome Python funkciją:

```
def show_mel_img(base_path, img_h):
  y, sr = librosa.load(base_path + "//audio//fold2//100652-3-0-0.wav")
  S = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=img_h, fmax=8000)
  S_dB = librosa.power_to_db(S, ref=np.max)
  img = librosa.display.specshow(S_dB, x_axis='time',y_axis='mel', sr=sr, fmax=8000)
  plt.colorbar(format='%+2.0f dB')
  plt.title('Mel spectrogram')
  plt.show()
```

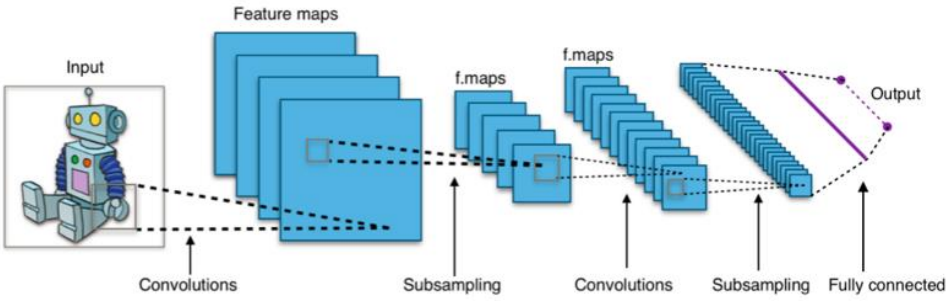
9 pav. Funkcija Mel‘o spektrogramos atvaizdavimui



10 pav. Garso signalas apvaizduotas Mel‘o spektrograma

2.3 Duomenų klasifikavimo metodas

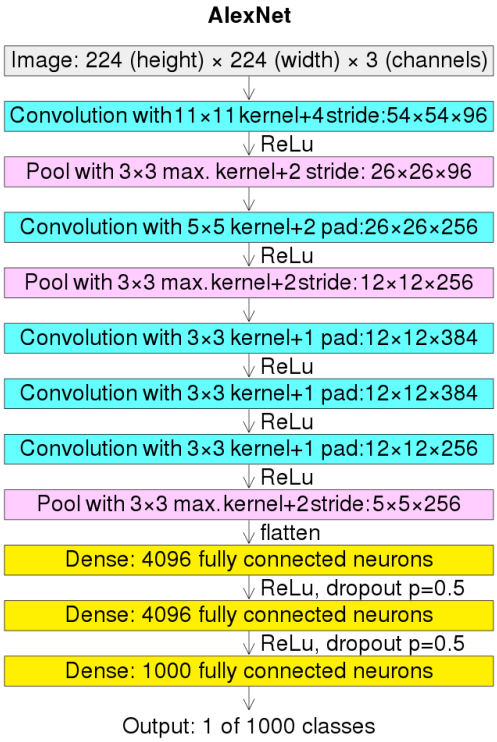
Kaip minėjome anksčiau duomenų klasifikavimui naudosime gilųjį konvoliucinį neuroninį tinklą. Šio tipo dirbtiniai neuroniniai tinklai, dažniausiai taikomi kompiuterinės regos srityje: vaizdų atpažinimui, objektų radimui ir identifikavimui nuotraukose, vaizdų segmentacijai ir natūralios kalbos atpažinimui. Pats pavadinimas „konvoliucinis neuroninis tinklas“ rodo, kad tinklui mokantis naudojama matematinė operacija, vadinama konvoliucija. Šie tinklai yra specializuotas neuroninių tinklų tipas, kuriame konvoliucija naudojama vietoj bendros matricos daugybos bent viename iš jų sluoksnių [17]. Tuo tarpu konvoliucinis neuroninis tinklas susideda iš įėjimo sluoksnio, paslėptų sluoksnių ir išėjimo sluoksnio.



11 pav. Tipinė konvoliucinio neuroninio tinklo struktūra

Šiame darbe naudosime tinklo struktūrą panašią į „AlexNet“ tinklą [4]. Pastarąją konvoliucinio neuroninio tinklo architektūrą, kurią sukūrė Alex Krizhevsky, bendradarbiaudamas su kitais mokslininkais. 2012 m. rugsėji „AlexNet“ dalyvavo „ImageNet Large Scale Visual Recognition Challenge“ varžybose[18]. Tinklas buvo tarp 5 geriausių

darbų pagal klaidų lygį - 15,3 %, o antroje vietoje likusį modelį aplenkė daugiau nei 10,8 procentinio punkto. Pagrindinis pirminio dokumento rezultatas buvo tai, kad modelio gylis buvo labai svarbus jo aukštam našumui. Modelio struktūra pateikta 12 pav. Vis dėlto didelis parametrų skaičius, kuris buvo brangus skaičiavimo požiūriu, buvo įmanomas dėl grafikos apdorojimo blokų (GPU) naudojimo mokymo metu. Eksperimentai buvo atlikti 2012m. todėl šis modelis buvo išskirstytas į dvi dalis, kurios paraleliai dirbo naudojant dvi vaizdo plokštes (GPU). Ši problema mums nebėra aktuali, nes dabartinės įrangos galimybės leidžia treniruoti sudėtingų struktūrų modelius su milijonais parametrų naudojantis paprastas žaidimams skirtas vaizdo plokštes.



12 pav. AlexNet konvoliucinio neuroninio tinklo struktūra

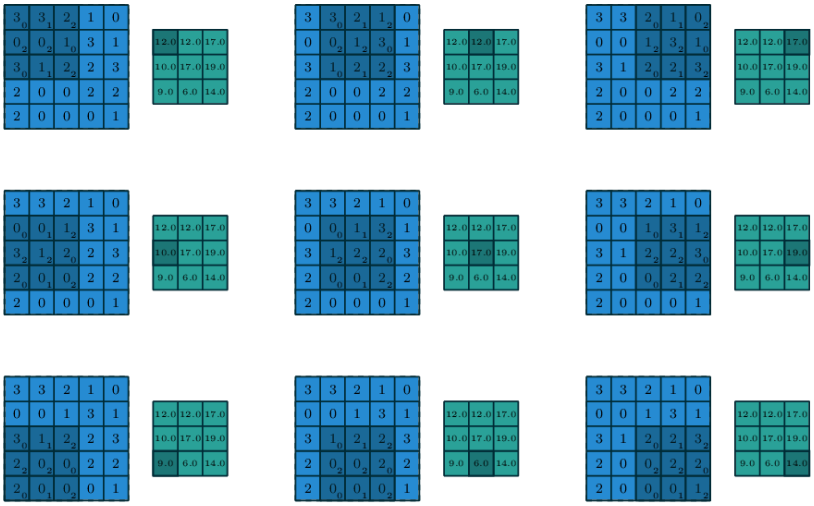
Naudodami konvoliucinius sluoksnius, MaxPooling2D ir Dropout funkcijas sudarome savo neuroninį tinklą kurio įėjimas būtų spektrogramos vaizdas (64x128 pikselių), o išėjimas klasės kuriai priskiriamas garsas numeris. Tinklo struktūrą pateikiame 13 pav.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 128, 96)       960
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 64, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 64, 128)       110720
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 32, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 16, 32, 128)       0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 18, 34, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 34, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 17, 256)        0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 11, 19, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 19, 512)       1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 9, 512)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 9, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 23040)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              23593984
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 26,240,842
Trainable params: 26,240,842
Non-trainable params: 0
_________________________________________________________________
```

13 pav. Pradinė tinklo struktūra

2.3.1 Dvimatis konvoliucinis sluoksnis

Dvimatė konvoliucija yra gana paprasta operacija: sukuriame branduolį (angl. kernel), kuris yra tiesiog maža svorių matrica. Šis branduolys „slenka“ per 2D įvesties duomenis, atlikdamas elementų dauginimą iš įvesties dalies, kurioje jis šiuo metu yra, tada rezultatai yra sumuojami į vieną išvesties pikselį. Branduolys pakartoja šį procesą kiekvienoje vietoje, per kurią slysta, konvertuodamas 2D funkcijų matricą į dar vieną 2D reikšmių matricą. Išėjimo reikšmės iš esmės yra įėjimo reikšmių svertinės sumos, esančios maždaug toje pačioje įvesties sluoksnio išėjimo pikselio vietoje.



14 pav. Dvimatės konloviucijos pavyzdys

Kaip matome iš 14 pav. dvimatė konvoliucija sumažina pradinį vaizdą. Paprasčiausiu atveju sluoksnio išėjimo reikšmę, kai įvesties dimensijos yra (N, C_{in}, H, W), o išvesties (N, C_{out}, H_{out}, W_{out}), galima tiksliai apibūdinti taip:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

Kur ★ išreiškiamas 2D kryžminės koreliacijos operatorius, N yra paketo dydis (angl. batch size), C reiškia kanalų skaičių, H yra įvesties vaizdo aukštis pikseliais, o W yra plotis pikseliais. Šį sluoksnį galime naudoti pasitelkiant „TensorFlow“ esančią funkciją: *tf.keras.layers.Conv2D()* .

Sluoksnio aktyvacijos funkcijai naudosime ReLU. Kompiuterinėje regoje, siekiant duomenų normalizacijos, populiariausia yra ReLU aktyvavimo funkcija, kuri yra įvardijama, kaip geriausiai tinkanti neuroninio tinklo mokymo efektyvumui. Ši funkcija aprašoma kaip:

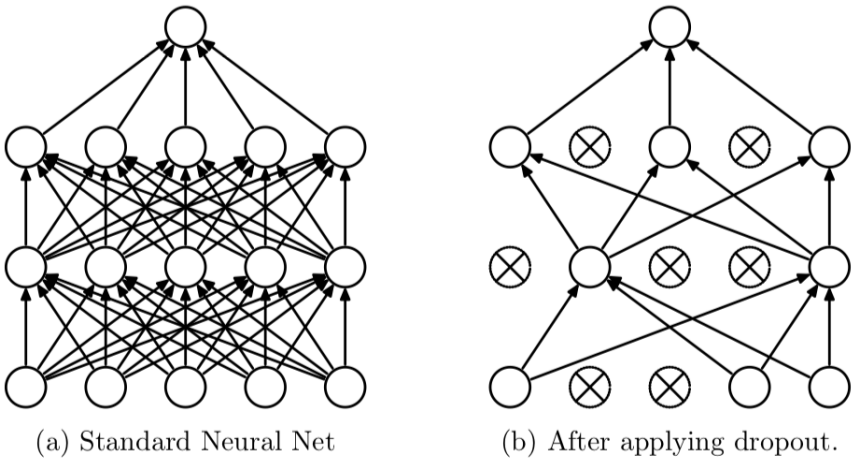
$$f(x) = x^+ = \max(0, x)$$

Akivaizdu kad ši funkcija gautas neigiamas įėjimo reikšmės, po konvoliucijos operacijos, pakeičia į 0.

2.3.2 Parametrų išmetimo technika

Išmetimas (angl. Dropout) yra reguliarumo metodas, skirtas sumažinti persimokymą ir pagerinti giliųjų neuronų tinklų testavimo rezultatus. Ši technika reiškia neuronų (paslėpto ar matomo sluoksnio) pašalinimą iš tinklo. Išmetimą galime interpretuoti kaip tikimybę, kad tam tikras sluoksnio neuronas bus paliktas mokymuisi. Kur 0 reiškia, kad nėra iškritimo, o 0,5 reiškia, kad 50% sluoksnio neuronų yra ignoruojami. Akivaizdu kad tada, kiekviena iteracija naudoja skirtingą modelio parametrų pavyzdį, o tai verčia kiekvieną neuroną turėti kuo svarbesnes savybes, kurias galima būtų naudoti su kitais atsitiktiniais neuronais. Nors

naudojant šią techniką yra mažinamas neuroninio tinklo sudėtingumas (sumažėja apmokamų parametrų skaičius), tačiau išmetimas taip pat padidina mokymo laiką, reikalingą modelio tikslumo konvergencijai. Originaliame darbe apie išmetimą pateikiami eksperimentiniai standartinių mašininio mokymosi problemų rinkinio rezultatai. Mokslininkai pateikė daugybę naudingų patarimų, į kuriuos reikia atsižvelgti, praktikoje naudojant šią techniką. Pagrindinis siūlymas yra paprastai naudoti nedidelę 20–50% neuronų iškritimo vertę. Pradedant modelio kalibravimą 20% – geras atskaitos taškas. Akivaizdu, kad per maža tikimybė turi minimalų poveikį, o per didelė vertė lemia nepakankamą tinklo mokymąsi.



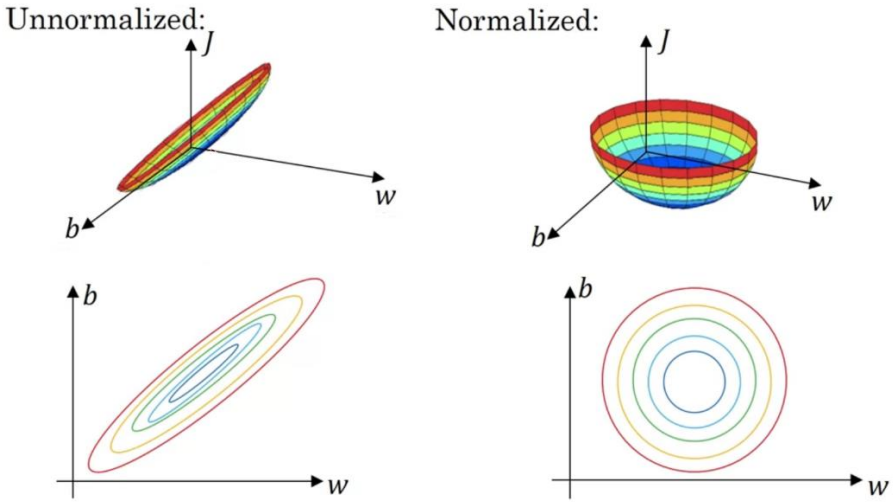
15 pav. Įprastas gilusis neuroninis tinklas (kairėje) ir tinklas naudojant išmetimo techniką (dešinėje)

Apibendrinus išmetimas yra puiki technika stengiantis sumažinti tinko prisitaikymą prie mokymo duomenų ir padidinti gebėjimą teisingai klasifikuoti nematytus duomenis. Todėl remdamiesi „AlexNet“ struktūra (kurioje naudoti du išmetimo sluoksniai su 0,5 tikimybę) į savo modelį įvesime analogišką techniką į keletą priešpaskutinių sluoksnių, o dar viename sluoksnyje arčiau įvesties naudosime išmetimą su 0,1 tikimybę.

2.3.3 Partijos normalizavimas

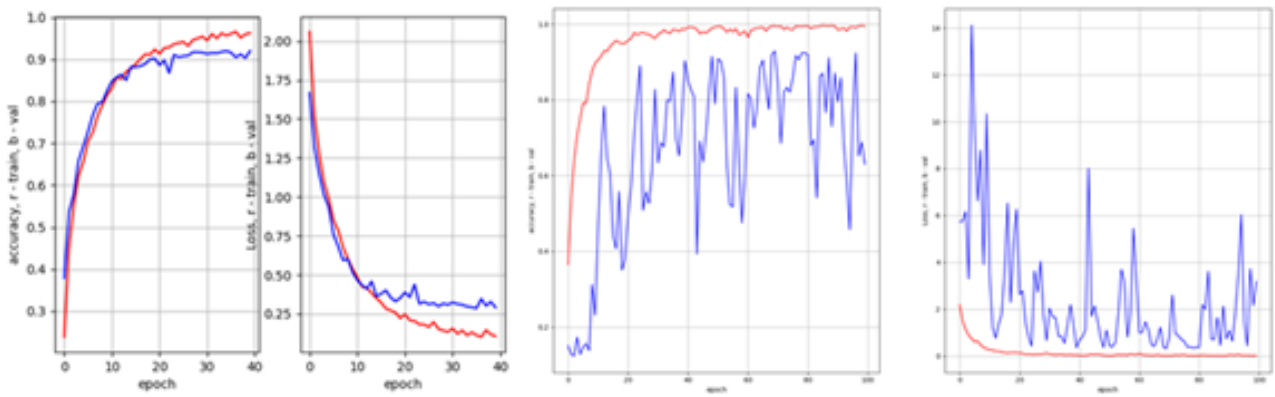
Partijos normalizavimas (angl. Batch normalization) yra metodas, naudojamas dirbtinius neuroninius tinklus padaryti greitesnius ir stabilesnius normalizuojant įvesties sluoksnį - percentruojant ir keičiant mastelį. Šią techniką 2015 m. pasiūlė Sergejus Ioffas ir Christianas Szegedy [19]. Partijos normalizavimas yra skirtas itin giliems neuroniniams tinklams. Šis mokymo metodas, standartizuoja kiekvienos mažos partijos sluoksnio įvestis. Tai stabilizuoja mokymosi procesą ir žymiai sumažina mokymosi laiką, reikalingą giliems tinklams apmokyti. Šio metodo vizualizacija pateikiama 16 pav.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$



16 pav. Partijos normalizavimo vizualizacija

Testuojant šią techniką pastebėjome, kad modelio tikslumas ir paklaida įvedus partijos normalizavimą tampa nepastovūs 17 pav. Kaip matome šios technikos naudojimas treniravimo mokymosi tikslumui įtakos neturėjo (abu modeliai pasiekė aukšta, apie 98%, treniravimo tikslumą), tačiau žiūrint į validavimo duomenis pastebime, kad įvedus normalizavimą atsiranda nepastovumas ir nematome konvergencijos (net ir padidinus mokymosi epochų skaičių nuo 40 iki 100). Vis dėlto verta paminėti, kad naudojant partijos normalizavimą buvo pastebėtas greitesnis apmokymo laikas, o klaida galimai slypi ne pačioje technikoje, o neteisinguose „Keras“ funkcijų parametruose. Taigi galutiniame modelyje šio metodo bus pilnai atsisakyta arba palikta tik keliuose viduriniuose tinklo sluoksniuose. Aišku mūsų tiriama duomenų bazės struktūra arba esantys įrašai galimai turėjo įtakos šiems rezultatams. Svarbu paminėti, kad partijos normalizavimas nerekomenduojamas kaip alternatyva tinkamam modelio duomenų paruošimui [20].



17 pav. Partijos normalizavimo technikos testavimas, kairėje nenaudojamas normalizavimas, dešinėje naudojamas

2.3.4 Modelio optimizatorius

Optimizatoriai yra algoritmai arba metodai, naudojami pakeisti neuroninio tinklo atributus, tokius kaip svoriai ir mokymosi greitis, siekiant sumažinti paklaidą. Šie algoritmai yra atsakingi už nuostolių mažinimą ir kuo didesnio tikslumo užtikrinimą. Vis dėlto, daugelis žmonių naudoja optimizatorius treniruodami neuroninį tinklą, nežinodami, koks metodas yra naudojamas, todėl svarbu apžvelgti šiame darbe naudojamą optimizatorių.

„Adam“ (Adaptive Moment Estimation) optimizatorius yra išplėsta stochastinio gradiento nusileidimo versija, kuri yra naudojama įvairiose giliojo mokymosi programose, tokiose kaip kompiuterinė rega ar natūralios kalbos apdorojimas. Ši technika pirmą kartą buvo pristatyta 2014 m [21]. „Adam“ algoritmo pagrindas yra ta, kad mes nenorime judėti labai greitai, nes tada išauga tikimybė peršokti minimumą, todėl stengiamės šiek tiek sumažinti greitį, kad būtų kruopšti paieška. Optimizatorius naudoja gradiento pirmojo ir antrojo momentų įvertinimus, kad pritaikytų mokymosi greitį kiekvienam neuroninio tinklo svoriui. „Adam„ siūlomas kaip efektyviausias stochastinis optimizatorius, kuriam reikia tik pirmos eilės gradientų, tuo pat metu nereikalaujant didelių atminties resursų. Prieš „Adam“ buvo pristatyta daug adaptyvaus optimizavimo metodų, kaip „AdaGrad“, „RMSP“, kurių našumas yra geras, palyginti su stochastinio gradiento nusileidimu, tačiau šie algoritmai turi tam tikrų trūkumų, pavyzdžiui, generalizavimo našumas (per didelis prisitaikymas prie mokymo duomenų), kuris kai kuriais atvejais yra prastesnis nei stochastinio gradiento nusileidimo. Taigi, „Adam“ buvo pasiūlytas, kaip algoritmas su geresniu generalizavimo našumu.

Norint matematiškai aprašyti „Adam“ pradedame nuo gradiento vidurkio ir dispersijos (pirmojo ir antrojo momento) skaičiavimo, naudodami į stochastinį gradiento nusileidimą panašias formules:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_1 v_{t-1} + (1-\beta_2) g_t^2$$

Kur v_t yra eksponentiškai mažėjantis praeities gradientų kvadratų vidurkis, o m_t yra eksponentiškai mažėjantis praeities gradientų vidurkis. β_1 ir β_2 yra nykimo laipsniai, kurie kontroliuoja santykinį praeities ir dabartinio gradiento indėlį. Pagal nutylėjimą naudojant „Keras‘‘ šios vertės yra: $\beta_1 = 0{,}9$ ir $\beta_2 = 0{,}999$. Bendrai šios reikšmės dažniausiai būna labai didelės, tai reiškia, kad optimizatoriui didesnę įtaką daro praeities vertės nei dabartinės. Problemos atsiranda naudojant šias išraiškas, kadangi dažniausiai v_t ir m_t konverguoja į 0. Taip atsitinka, nes pirmuoju algoritmo vykdymo metu v_t ir m_t inicijuojami kaip nuliniai vektoriai. Todėl „Adam“ įveda papildomą šališkumo korekciją savo formulėje:

$$\hat{m}_t = \frac{m_t}{1-\beta_1}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2}$$

Šias pataisytas, gradiento pirmąjį ir antrąjį momentus naudojančias, reikšmes sujungiame į tą pačią adaptyviąją mokymosi greičio formulę, kurią naudoja šaknies vidurkio kvadrato šeimos algoritmai:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

Taigi, „Adam“ šiuo metu yra vienas iš populiariausių optimizavimo algoritmų, pagrinde dėl to, kad jis užtikrina greitą konvergenciją, yra gebus mokytis esant didėliai dispersijai ir turi keičiamus β_1 ir β_2 parametrus. Vis dėlto svarbu paminėti, kad „Adam“ kompiuterinių skaičiavimų požiūriu yra brangus algoritmas.

3. Testavimas

Šio darbo testavimas buvo atliekamas kompiuteriu su Intel Core i5-4590 CPU 3.30GHz procesoriumi. Operatyviosios atminties kiekis – 16 GB, o konvoliucinių neuroninių tinklų mokymui svarbiausias elementas buvo naudota Nvidia GeForce GTX 1050Ti vaizdo plokštė.

Taigi galutiniame modelio variante pasirinkome nenaudoti partijos normalizavimo technikos bet palikome išmetimo techniką. Galutinė modelio architektūra pavaizduota 18 pav.

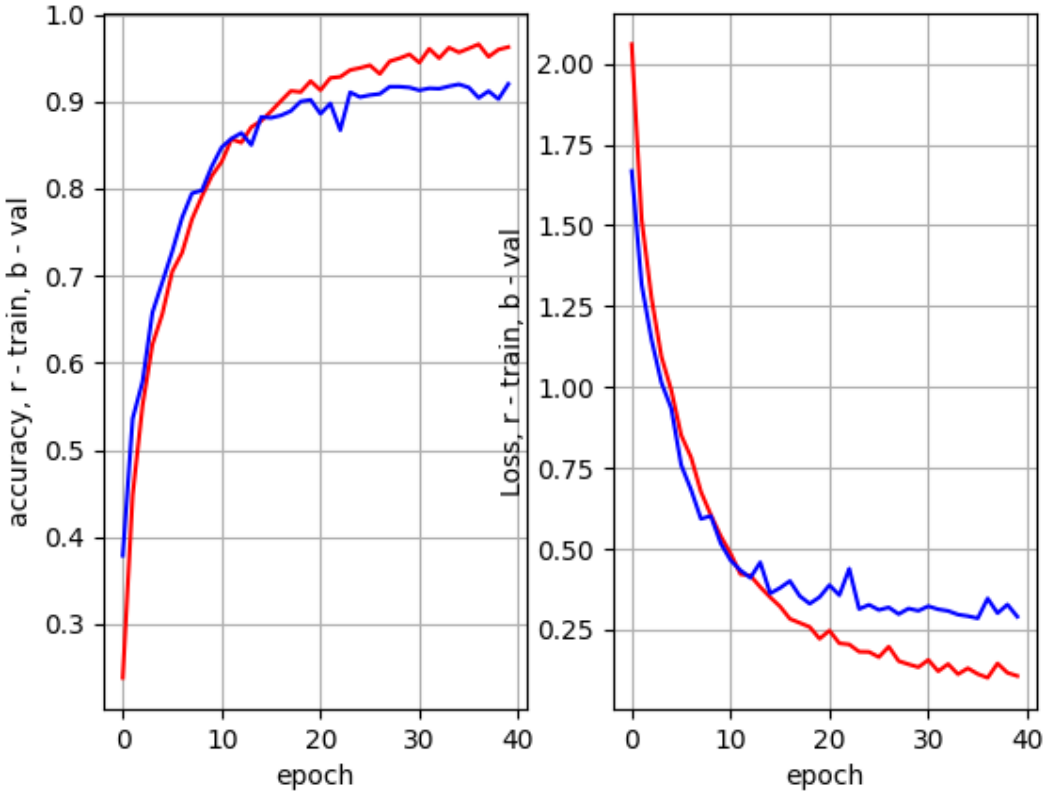
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 128, 96)       960
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 64, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 64, 128)       110720
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 32, 128)       0
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 18, 34, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 34, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 17, 256)        0
_________________________________________________________________
dropout (Dropout)            (None, 9, 17, 256)        0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 11, 19, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 19, 512)       1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 9, 512)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 9, 512)         0
_________________________________________________________________
dense (Dense)                (None, 5, 9, 1024)        525312
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 9, 1024)        0
_________________________________________________________________
flatten (Flatten)            (None, 46080)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              47186944
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 49,309,514
Trainable params: 49,309,514
Non-trainable params: 0
_________________________________________________________________
```

18 pav. Galutinė neuroninio tinklo struktūra

Testavimo metu buvo atliekamos dvejos modelio įvertinimo technikos, tai yra - naudojant testavimo ir treniravimo duomenis santykiu 75/25, ir K-karto kryžminis patvirtinimas naudojant 10 skirtingų duomenų kombinacijų pateikiamų UrbanSound8K duomenų rinkinyje. Toliau aptarsime ir palyginsime rezultatus gautus abejais būdais.

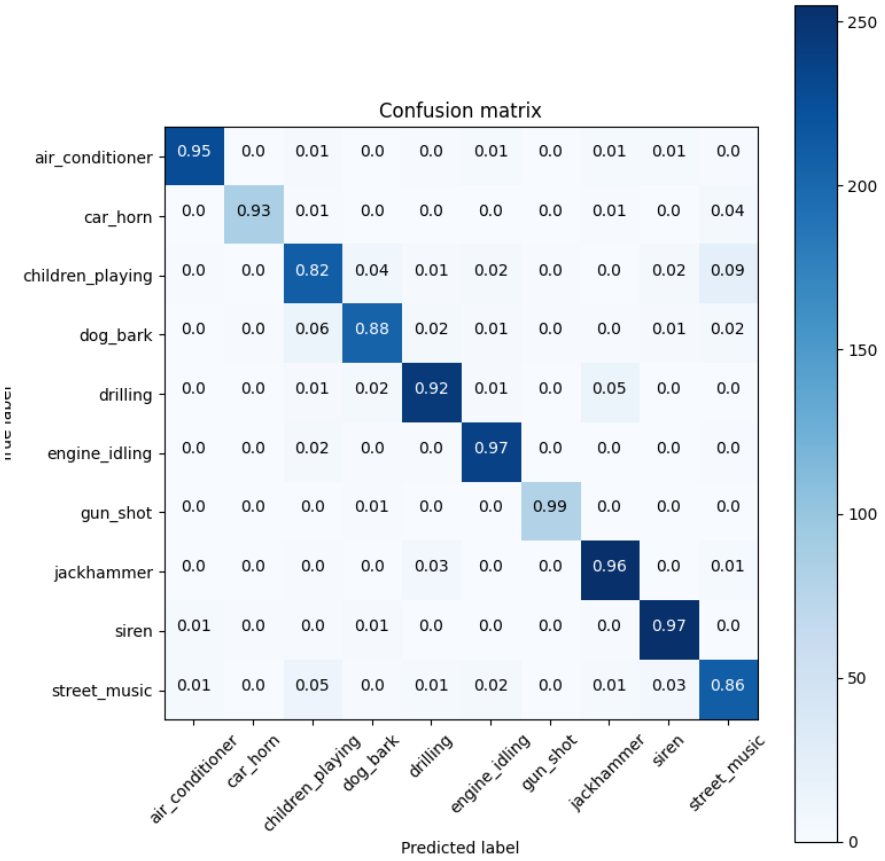
3.1 Testavimas naudojant patikrai

Kaip anksčiau minėjome treniravimui buvo nuspręsta naudoti 75% duomenų t.y. 6549 garso įrašų, o patikrai 25% - 2183. Kadangi buvo naudojama gan gili modelio struktūra ir įėjimai buvo paveikslėliai, (64x128 pikselių dydžio) treniravimas buvo atliktas tik 40 epochų. Galiausiai, kaip matome iš 19 pav., modelis sugeba pasiekti apie 92% tikslumą.



19 pav. Modelio klasifikavimo tikslumo ir klaidų grafikai. Mėlyna kreivė validavimo duomenys, raudona - treniravimo

Rezultatus vizualizuoti pasitelkiame klaidų matricą 20 pav. Kaip matome vienas iš geriausių rezultatų buvo pasiektas testuojant su sirenos garsais – 97%. Tai paaiškinti galima tuo, kadangi sirena yra pasikartojantis signalas kurį klasifikuoti neuroniniam tinklui tikriausia buvo lengva. Mažiausias tikslumas – 82% matomas ties vaikų žaidimų garsais. To priežastis yra ganėtinai neapibrėžta pati garso klasė, kadangi vaikai žaisdami skiria ganėtinai įvairius garsus, todėl ir turimų duomenų imtis neleido pasiekti aukštesnių rezultatų.



20 pav. Modelio klaidų matrica

3.2 Testavimas naudojant 10-kartų kryžminį patvirtinimą

Oficialioje UrbanSound8K duomenų rinkinio svetainėje autoriai skelbia, kad norint publikuoti straipsnį naudojant šią duomenų bazę reikia atlikti 10 kartų kryžminį patvirtinimą [22]. Svarbu naudoti šią techniką ir būtent 10 skirtingų rinkinių kryžminį patikriną, nes buvo pastebėta kad su kai kurias duomenų rinkiniais yra pasiekiamas per aukštas tikslumas. Duomenų rinkinio autoriai teigia, kad svarbu atlikti pilną dešimties kombinacijų patikrą (o ne penkerių ar dar mažesnę) ir įvertinti gautų modulių parametrų vidurkį. Modifikavus savo kodą šiems reikalavimams išpildyti gavome 72% modelių tikslumo vidurkių reikšmę kaip pateikta 21 pav.

```
Epoch 00023: val_accuracy did not improve from 0.77539
Epoch 00023: early stopping
Temp k-Folds Accuracy: 0.7210669485277077
Average 10 Folds Accuracy: 0.7210669485277077
```

21 pav. Validavimo tikslumas naudojant kryžminį patvirtinimą

Tikslumo nuosmukiui, lyginant su paprastu testavimo metodu, įtakos turėjo ir tai, kad treniravimo laikas išaugo bent 10 kartų. Taip atsitiko nes modelį reikia testuoti su dešimt skirtingų paketų, be to ir pats treniravimo duomenų skaičius išaugo iki maždaug 90%

(priklausomai nuo pasirinko aplankalo, nes juose esančių įrašų skaičius nėra vienodas). Todėl taupant laiką teko sumažinti epochų skaičių iki 30. Vis dėlto šie rezultatai nėra blogi, nes mokslininkai naudodami šią duomenų bazę moksliniuose darbuose pasiekia 75-85% tikslumą.

Išvados

1. Susipažinta su aplinkos garsų bei muzikos klasifikavimų metodikomis kituose moksliniuose darbuose.
2. Išanalizuotos ir praktiškai išbandytos technikos neuroninių tinklų technikos: dvimatė konvoliucija, išmetimas, partijos normalizavimas.
3. Praktiškai pavyko suklasifikuoti aplinkos garsus naudojant UrbanSound8K duomenų rinkinį.
4. Naudojant 75/25 treniravimo/tikrinimo santykį buvo pasiektas apie 92% modelio tikslumas.
5. Naudojant 10 imčių kryžminę validaciją pasiektas nemažesnis kaip 72% modelio tikslumas.

Naudotos literatūros sąrašas.

1. Music Genre Classification Derek A. Huang.
   Prieiga per: http://cs229.stanford.edu/proj2018/report/21.pdf
2. GTZAN muzikos žanrų duomenų rinkinys.
   Prieiga per: http://marsyas.info/downloads/datasets.html
3. Classifying environmental sounds using image recognition networks. Venkatesh Boddapati.
   Prieiga per: https://www.sciencedirect.com/science/article/pii/S1877050917316599
4. ImageNet Classification with Deep Convolutional Neural Networks Alex Krizhevsky.
   Prieiga per:
   https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
5. Going deeper with convolutions. Christian Szegedy.
   Prieiga per: https://arxiv.org/pdf/1409.4842.pdf
6. ESC duomenų bazės.
   Prieiga per:
   https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/YDEPUTf
7. DASEE buitinių garsų duomenų rinkinys.
   Prieiga per: https://www.kaggle.com/abigailcopiaco/daseedataset
8. Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks. Guangyong Chen.
   Prieiga per: https://arxiv.org/pdf/1905.05928.pdf
9. Nuotraukų CIFAR-10 duomenų rinkinys
   Prieiga per: https://www.cs.toronto.edu/~kriz/cifar.htmlf
10. Deep Residual Learning for Image Recognition. Kaiming He.
    Prieiga per: https://arxiv.org/pdf/1512.03385.pdf
11. A Dataset and Taxonomy for Urban Sound Research Justin Salamon
    Prieiga per:
    http://www.justinsalamon.com/uploads/4/3/9/4/4394963/salamon_urbansound_acmmm14.pdf
12. Layer Normalization. Jimmy Lei Ba.
    Prieiga per: https://arxiv.org/pdf/1607.06450.pdf
13. Instance Normalization: The Missing Ingredient for Fast Stylization. Dmitry Ulyanov
    Prieiga per: https://arxiv.org/pdf/1607.08022.pdf
14. Group Normalization. Yuxin Wu.
    Prieiga per: https://arxiv.org/pdf/1803.08494.pdf
15. Environmental sound classification using temporal-frequency attention based convolutional neural network (Wenjie Mu)
    Prieiga per:
    https://www.researchgate.net/publication/355903162_Environmental_sound_classification_using_temporal-frequency_attention_based_convolutional_neural_network
16. D. O'shaughnessy, Speech communication: human and machine. Universities press, 1987.

17. Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016).
    Prieiga per: https://www.deeplearningbook.org/
18. „ImageNet Large Scale Visual Recognition Challenge“ varžybs.
    Prieiga per: https://www.image-net.org/challenges/LSVRC/
19. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Sergey Ioffe
    Prieiga per: https://arxiv.org/pdf/1502.03167.pdf
20. Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions. Jason Brownlee 187 psl.
21. Adam: a method for stochastic optimization. Diederik P. Kingma
    Prieiga per: https://arxiv.org/pdf/1412.6980.pdf
22. UrbanSound8K duomenų rinkinys
    Prieiga per: https://urbansounddataset.weebly.com/urbansound8k.html