VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

# Spalvų vaizduose keitimas naudojant dirbtinius neuroninius tinklus

## Color change in pictures using artificial neural networks

Magistro baigiamasis darbas

Atliko: Deivis Maziukas (parašas)

Darbo vadovė: Prof. Dr. Olga Kurasova (parašas)

Recenzentas: Prof. Dr. Rimantas Vaicekauskas (parašas)

Vilnius – 2021

# Santrauka

Spalvų vaizduose keitimas naudojant dirbtinius neuroninius tinklus yra tema, tyrinėjanti dirbtinių neuroninių tinklų panaudojimą keičiant nespalvotus paveikslėlius į spalvotus. Tema yra dalis iš dirbtinių neuroninių tinklų panaudojimo vaizdų analizei dalių.

Dirbtiniai neuroniniai tinklai naudojami plačioje užduočių srityje. Kartu tobulėjant technologijoms, atsiranda vis geresnių būdų atlikti paskirtas užduotis panaudojant dirbtinius neuroninius tinklus.

Spalvų vaizduose keitimas apima kelis skirtingus būdus kaip panaudojant dirbtinius neuroninius tinklus galima nespalvotam paveikslėliui suteikti kuo tikroviškesnes spalvas. Metodai skiriasi pagal savo panaudojimo būdą, vaizdų kokybę, taip pat skiriasi ir problemos, su kuriomis susiduria skirtingi neuroniniai tinklai.

Šiame magistro baigiamajame darbe buvo išanalizuotos pasirinkto dirbtinio neuroninio tinklo galimybės, pakeisti parametrai, pakeista architektūra, siekiant gauti kuo geresnį rezultatą. Rezultatai buvo vertinami lyginant originalius paveikslėlius su nuspalvintais paveikslėliais, naudojant vaizdų kokybės palyginimo metrikas.

Darbo eigoje buvo atliktos analizės įvairiais parametrais, įvairiomis pakeistomis dirbtinio neuroninio tinklo architektūromis. Buvo įvertinti analizės rezultatai apskaičiuojant juos matematinėmis formulėmis. Geriausi rezultatai buvo palyginti su kitu neuroniniu tinklu.



Raktiniai žodžiai: Dirbtinis neuroninis tinklas, analizė, rezultatai, vaizdų spalvinimas, palyginimas

## Summary

Color change in pictures using artificial neural networks is about using artificial neural networks to change grayscale images to coloured images. This thesis is part of a larger part of computer imagery and imagery analysis.

Artificial neural networks are used in a very wide variety of tasks. As technology progresses, so does artificial neural networks and efficiency in which they are doing their tasks.

There are multiple ways how artificial neural networks can be used to colour images. They differ in their processes, how they work, results they show and others. There are also problems which exist by using neural networks to colour images.

In this thesis, an artificial neural network was chosen, to analyze its capabilities and limitations. Also, different parameters of this neural network were analyzed, as well as different architectures to get as good as possible results. Final coloured images were compared using image comparison algorithms. Best results were compared with other artificial neural networks.



Key words: artificial neural networks, analysis, results, colorization, comparison

Turinys

# 1. Įvadas

## 1.1.Temos aktualumas

Spalvų vaizduose keitimas naudojant dirbtinius neuroninius tinklus – tema, skirta grafinei vaizdų manipuliacijai. Tai apima spalvų, rezoliucijos, dydžio keitimą ir kitus vaizdo parametrų keitimus. Vaizdų keitimas turi platų panaudojimo ratą, pavyzdžiui, spalvos gali būti pridėtos į nespalvotus vaizdus kad pagerinti jų išraiškingumą [TMK02]. Taip pat vaizdų keitimas panaudojant stilius labai svarbus socialiniuose tinkluose kur nuotraukų dalinimasis ir pramogos yra pagrindinės dalys [JYL+17]. Jų panaudojimas vaizdams keisti išpopuliarėjo kartu su pačių dirbtinių neuroninių tinklų išpopuliarėjimu. Populiarios dirbtinių neuroninių tinklų bibliotekos kaip *TensorFlow* arba *PyTorch* leido lengviau panaudoti dirbtinius neuroninius tinklus įvairiose srityse. Išpopuliarėję neuroniniai tinklai leido programuotojams sukurti įvairios programinės įrangos, skirtos vaizdų keitimui – nuo paprastų spalvų keitimų ir objektų atpažinimų iki paveikslų stilių perkėlimo į kitus vaizdus.

## 1.2.Pagrindinis darbo tikslas

Pagrindinis darbo tikslas – modifikuoti pasirinktą dirbtinį neuroninį tinklą, skirtą nespalvotų vaizdų spalvinimui, siekiant sukurti tinklą, kuris gebėtų kuo tikroviškiau nuspalvinti vaizdus.

## 1.3. Keliami pagrindiniai darbo uždaviniai

Įgyvendinant magistro baigiamojo darbo tikslą, būtina įgyvendinti tam tikrus konkrečius uždavinius:

1. Ištirti kaip veikia išsirinktas dirbtinis neuroninis tinklas, atlikti pirminį galimybių vertinimą ir toliau pakeisti neuroninio tinklo nustatymus siekiant gauti geresnius rezultatus.
2. Apmokyti sukurtą dirbtinį neuroninį tinklą pagal nustatytas analizavimo taisykles iš vaizdų mokymo aibės. Šios aibės turi būti sudarytos iš nespalvoto vaizdo ir tikrų spalvų arba kitaip - originalaus vaizdo. Nespalvotas vaizdas turi būti naudojamas vaizdo spalvų keitimui, spalvotas, tikrų spalvų vaizdas bus naudojamas dirbtinio neuroninio tinklo tikslumui įvertinti.
3. Atlikti vaizdų spalvinimo procesą, sukurti dirbtinio neuroninio tinklo nuspalvintus paveikslėlius. Šie dirbtinio neuroninio tinklo sukurti spalvoti vaizdai bus naudojami neuroninio tinklo skaitiniame vertinime. Kuriant spalvotus vaizdus įvertinti ir tinklo architektūros parametrus, sukurti papildomus spalvotus vaizdus su kitais architektūros parametrais.

4. Atlikti dirbtinio neuroninio tinklo tikslumo vertinimą – atlikti matematinį tikslumo vertinimą. Gautus vertinimo rezultatus palyginti ir įvertinti pasikeitimus ir iš to padaryti išvadas.

### 1.4.Laukiami rezultatai

Sėkmingai įgyvendinto magistro baigiamojo darbo tikslai turi leisti gauti rezultatus, kuriuos galima padalinti į keletą dalių:

1. Surasta optimalesni dirbtinio neuroninio tinklo nustatymai, kurie pagerino galimybes, jei būtų taikomi pradiniai nustatymai pagal įvertintus vaizdų spalvinimo rezultatus.
2. Įvertinta, kurie dirbtinio neuroninio tinklo architektūros sprendimai padėjo geriausiai pagerinti rezultatą nuo pradinio tinklo nustatymų pagal gautus spalvinimo matavimo rezultatus.
3. Išaiškinta, kuris architektūros sprendimas arba pakeitimas ir kokia paklaida kuria spalvotus vaizdus, kurios labiausiai panašios į tikrąsias vaizdo spalvas.

## 2. Teorinė dalis apie dabar esančius neuroninių tinklų taikymus

Šiame skyriuje bus pateikiami teorinės magistro baigiamojo darbo idėjos, teorinė literatūros analizė, teoriškai pagrįsti problemos sprendimo būdai, taip pat bendroji teorinė keliamos problemos ir norimo pasiekti tikslo dalis. Šiame darbe, teorinė dalis dėl nuoseklumo, bendros tvarkos, pagalbos sau planuoti ir paprastumo yra padalinta į kelias teorinę sritį apimančias dalis:

1. Bendras įvadas į neuroninius tinklus ir kitas technologijas – šioje dalyje bus apibendrinta teorinė dalis apie technologijas ir sąvokas, kurios bus naudojamos šiame magistro baigiamajame darbe. Ši teorinė dalis yra reikalinga tam, kad būtų įvestas bendras supratimas apie ką bendrai yra šis magistro baigiamasis darbas.
2. Neuroninių tinklų ir kitų darbe minimų technologijų aktualumas ir praktinis taikymas – šioje dalyje bus apibendrinti bendri faktai ir žinios apie dirbtinių neuroninių tinklų ir kitų technologijų, naudojamų šiame magistro baigiamajame darbe, aktualumą visuomenėje ir technologijų srityje. Ši teorinė dalis yra reikalinga tam, kad būtų pagrįsta teorija, kad naudojamos technologijos yra aktualios ir turi praktinį taikymą, tokiu pačiu būdu įrodant, kad ir šis magistro baigiamasis darbas turi teorinę prasmę ir turi praktinį pritaikymą ir aktualumą.
3. Teorinė dalis apie magistro baigiamojo darbo uždavinį – šioje dalyje bus apibendrinta magistro baigiamojo darbo temos kilmė, reikšmė ir keliamos problemos teorija. Ši dalis yra reikalinga tam, kad būtų faktais pagrįsta teorija, taip pat parodyta, kad

siekiami rezultatai gali turėti teorinį ir praktinį taikymą. Šioje dalyje bus pasidalinta bendrai kokio tipo uždaviniai ir kokiems tikslams buvo skiriami moksliniai tyrimai, pasidalinta tų tyrimų rezultatais. Išsiaiškinta, kokios dažniausiosios problemos ištinka sprendžiant tokio tipo uždavinius, aprašyta keliamomis priemonėmis kaip tas problemas spręsti. Pabaigai, bus išanalizuota, kokiais būdais tokie uždaviniai yra sprendžiami, apibrėžti tokių sprendimo kelių pliusai, tai yra teigiamos dalys, ir minusai, tai yra kylančios problemos. Užbaigus šią dalį, sudaryta teorinių žinių dalis turėtų būti daugiau nei pakankama toliau vystant magistro baigiamojo darbo iškeltus tikslus ir problemos sprendimus.

4. Įvertinti gautus rezultatus – pasitelkiant objektyvias metrikas įvertinti gautų spalvotų vaizdų kokybę, gautus rezultatus palyginti tarpusavyje ir išsiaiškinti, kuris iš jų davė didžiausią spalvų pritaikymo kokybės pagerėjimą.

## 2.1.Bendroji dalis apie naudojamas technologijas

Dar iki pradedant teorinę dalį apie siekiama tirti sritį jau galima iš karto susiaurinti technologinę sritį, kuri bus naudojama toliau magistro baigiamajame darbe.

1. Dirbtinis intelektas – angliškai „*Artificial intelligence*“ (trumpinys AI) yra kompiuterijos mokslo viena iš dalių, kurioje specializuojamasi į protingų mašinų kūrimą, kad jos gebėtų spręsti problemas ir dirbti kaip žmonės. Tai nėra vienintelis iš apibrėžimų, nes bendro vieningo apibrėžimo nėra. Dirbtinis intelektas padeda mašinai orientuotis aplinkoje, atlikti sudėtingas užduotis, spręsti logines problemas, kitaip tariant, jis skirtas sukurti protingas mašinas.
2. Mašininis mokymasis – angliškai „*machine learning*“, irgi neturi bendro, vieningo apibrėžimo, bet jį galima apibūdinti iš esmės kaip vieną iš dirbtinio intelekto sričių, kurioje mašinos apmokamos spręsti problemas. Kitoks, alternatyvus apibūdinimas gali būti kaip algoritmai, gebantys atpažinti vienodus duomenų šablonus ir pagal juos galintys padaryti išvadas ir sprendimus. Dar dažnai sutinkamas apibrėžimas apibūdina mašininį mokymąsį kaip dirbtinio intelekto pritaikymą, kuriame sistema automatiškai mokosi ir tobulėja jos specifiškai ir konkrečiai neprogramuojant. Visi apibūdinimai iš esmės veda į vieną išvadą, kad mašininis mokymas skirtas apmokyti mašinas, kad jos atliktų darbą.
3. Gilusis mokymas – angliškai „*deep learning*“ – tai yra dalis iš mašininio mokymosi būdų, paremtas dirbtinių neuroninių tinklų darbu. Žodis „gilusis“ iš esmės atsiranda iš to, kad dirbtiniam neuroniniam tinkle egzistuoja keli sluoksniai neuronų, kurie gali priimti sprendimus iš gaunamų duomenų. Keli sluoksniai neuronų leidžia tokiam

tinklui dideliu patikimumu atlikti užduotis, susijusias su veido atpažinimu, kalbos atpažinimu, objektų atpažinimu ir daugybę kitų užduočių. Gilusis mokymas gali mokyti tiesiai iš duomenų šaltinių, nesvarbu kas tai būtų – paveikslėliai, žodžiai, tekstas. Dažnai pagerinti rezultatui tokiam tinklui galima tiesiog perduoti daugiau duomenų ir tokiu būdu pagerinti tinklo tikslumą.

4. Dirbtiniai neuroniniai tinklai – angliškai „*artificial neural network*“ (trumpinys ANN), kompiuterinės sistemos, kurių veikimas paremtas žmogaus arba kitų gyvūnų smegenų neuronų veikimu. Kaip ir visų gyvūnų smegenyse, taip ir dirbtiniuose neuroniniuose tinkluose keliais sluoksniais gali perdavinėti signalus į kitus sujungtus neuronus iki bus prieitas galutinis sluoksnis neuronų, kuriame bus gauta išvada iš dirbtinio neuroninio tinklo skaičiavimo.
5. Konvoliuciniai neuroniniai tinklai – angliškai „*convolutional neural network*“ – specializuota giliojo mokymo dirbtinių neuroninių tinklų dalis, skirta daugiausiai darbui su vaizdais. Šios užduotys apima, bet neapsiriboja vaizdų ir objektų klasifikavimu ir atpažinimu, spalvų atpažinimu, veidų atpažinimu. Verta paminėti, kad konvoliuciniai neuroniniai tinklai taip pat turi panaudojimo kryptį medicinoje, analizuojant medicininius vaizdus, natūralios kalbos atpažinime ir įvairiose kitose srityse.

Apibendrinant, šioje dalyje buvo įvardinti pagrindinės sąvokos, kurios bus toliau naudojamos magistro baigiamajame darbe. Nustačius pagrindines sritis, galima toliau parodyti, kiek svarbios ir aktualios yra paminėtos technologijos.

## 2.2. Neuroninių tinklų ir mašininio mokymosi aktualumas

Šioje dalyje bus siekiama parodyti, kad anksčiau nurodytos technologijos, kurios bus naudojamos magistro baigiamajame darbe yra apskritai aktualios pasaulyje ir ar jos gali turėti praktinį taikymą atliekant užduotis, kuriant sprendimus. Dėl paprastumo ir aiškumo, esamus pavyzdžius dirbtinių neuroninių tinklų ir kitų su jais susijusių technologijų galima padalinti į dvi dalis:

1. Bendroji, kur bus pateikti įvairių sričių dirbtinių neuroninių tinklų pavyzdžiai.
2. Neuroninių tinklų, kurie skirti darbui su vaizdų, spalvų atpažinimu. Ši sritis gali būti išskirta, nes ši sritis yra aktualiausia šiame magistro baigiamajame darbe

Apibendrinimo dalyje bus konkrečiais pavyzdžiais parodyta, kad nurodytos technologijos yra aktualios ir yra verta bei prasminga toliau gilinti bei tobulinti žinias šioje mokslo ir technologijų srityje.

## 2.3. Dirbtinių neuroninių tinklų pavyzdžiai

Atsakymo, kokią įtaką dirbtiniai neuroniniai tinklai turi, ilgai ieškoti nereikia. Dirbtinio intelekto technologijos – neuroniniai tinklai, gilus mokymasis ir kitos, daro vis didesnę ir pastebimą reikšmę. Lengviausia ir prasmingiausia į šį klausimą atsakyti pateikiant jau esamas dirbtinių neuroninių tinklų technologijas, esamas kaip praktiškai pritaikytas arba kaip technologinius patentus – kasdieniam gyvenime esantys prietaisai ir technologijos daro didžiausią ir labiausiai pastebimą įtaką žmogaus veikloje. Neuroninių tinklų technologijos šiandieniniame pasaulyje naudojamos arba tiriamos daugelyje sričių, tokiose kaip:

- Informacinės technologijos – tokių pavyzdžių galima ieškoti tarp didžiųjų informacinių technologijų korporacijų, kurios valdo milžiniškus duomenų kiekius ir turi pakankamai resursų kurti, tirti ir naudoti dirbtinius neuroninius tinklus. Viena tokių korporacijų – milžinių *Google* turi jau nemažą patirtį darbe su neuroniniais tinklais. Korporacija gali pasiūlyti naudotis *Tensorflow,* atviro kodo lengvai naudojamą programavimo biblioteką, kurią galima pritaikyti ir dirbtiniams neuroniniams tinklams. *TensorFlow* taip pat naudojamas *Google* vidiniams darbams ir tyrimams. *Google* taip pat prisidėjo ir prie *Deep Mind* vystymo. Viena iš *Deep Mind* sukurtų programų *AlphaGo* jau 2016 metų kovą žaidimo „*Go*“ metu rezultatu 4-1 įveikė tuometinį pasaulio čempioną Lee Sedol.

Padaugėjus duomenų ir ištobulėjus technologijoms, dirbtinių neuroninių tinklų naudojimas pasiekė niekada nematytas aukštumas ir leidžiantis tinklus panaudoti naujoms, anksčiau negalimoms užduotims.

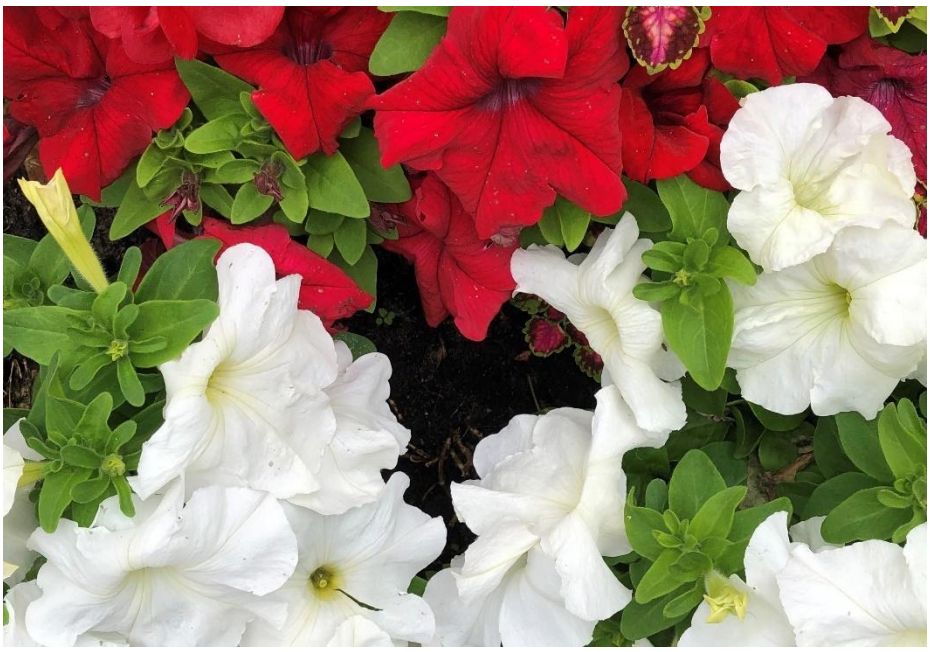
- Spalvų atpažinimas – ši sritis taip pat turi jau gerai išvystytas dirbtinio intelekto technologijas ir dirbtinių neuroninių tinklų naudojimą. Jų panaudojimas gali būti labai platus, pavyzdžiui, būti skirtas atpažinti automobilio spalvas [RP18]. Didelės korporacijos ir čia siekia panaudoti dirbtinius neuroninius tinklus. Vienas iš pavyzdžių – *Facebook* kuriamų naujų įrankių *Caffe2go*, kuris leidžia perkelti stilių (pavyzdžiui, iš žinomo paveikslo) ir perkurti vaizdą pagal tą stilių. Kaip rašoma, tai yra vienas iš įrankių, kuris leis vartotojams meniškai išreikšti save naujais būdais.

Aukščiau išvardintos sritys yra tik dalis tų, kuriose naudojami arba pradedami naudoti dirbtinių neuroninių tinklų technologijos, visas sritis išvardinti būtų neįmanoma. Iš jau išvardintų pavyzdžių galima tvirtinti, kad šiame magistro baigiamajame darbe siekiamos tirti technologijos turi didelę svarbą ir praktinį pritaikymą, kad jų tyrimas nebūtų beprasmis.

## 3. Spalvinimo dirbtiniais neuroniniais tinklais uždavinio analizė

Pagrindinė šio darbo analizuojama tema ir visos iš jos kylančios problemos – paveikslėlių spalvinimas, angliškai apibūdinimas kaip „*colorization*“. Tai yra užduotis, kaip paversti bespalvį, juodai-baltą paveikslėlį arba nuotrauką (1 pav.) į tokį, kuris turėtų kuo tikroviškesnes spalvas (2 pav.).

**1 pav. Gėlių nuotrauka be spalvų, nuotrauka kurta pačio magistro baigiamojo darbo autoriaus. Nuotraukai pritaikytas atvirkštinis procesas – nuo spalvotos nuotraukos (2 pav.) pašalintos spalvos.**



**2 pav. Originali, neretušuota, spalvota gėlių nuotrauka. Pilna ryškių spalvų, aiškių kontūrų tarp objektų. Nuotraukos autorius - šio magistro baigiamojo darbo autorius.**

Toks paveikslėlių spalvinimas iš pilkų, juodai-baltų atspalvių keitimas į spalvotą gali turėti įvairių praktinių taikymų, nuo pačių paprasčiausių tokių kaip tiesiog asmens hobis ir jo meninė išraiška, iki rimtesnių, tokių kaip istorinių fotografijų atkūrimas istorinio konteksto tikslui.

Verta paminėti, kad spalvų suteikimas neapsiriboja tik paveikslėliams. Vaizdų spalvinimas praktiškai gali apimti visą vaizdinę mediją ir turi taikymą įvairiose vaizdų srityse. Be paminėtų paveikslėlių spalvinimo, ši veiklą gali būti pritaikyta ir filmų spalvų atkūrimui (3 pav.).

**3 pav. Filmas „*Behind the screen*“ (1916), su Čiarli Čiaplinu, nuspalvotas neuroninio tinklo, originalas pilkų atspalvių (vaizdas iš [CQ19]).**

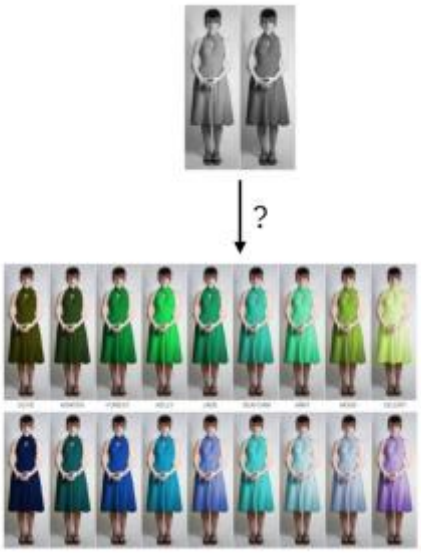
Tokių filmų spalvinimo tikslas pagal autorius jau pakankamai aiškus – suteikti žiūrovui didesnį pasitenkinimą matant spalvotą filmą [CQ19] . Be to, toks filmų spalvų atkūrimas gali padėti kitoms kompiuterių regos programoms, kurios apima vaizdų supratimą ir objektų sekimą [CQ19]. Žinoma, net paprasti filmai nėra apriboti spalvinimo galimybių. Pagal [HKM19], filmų spalvinimas gali būti išplėstas net į kitokio žanro filmus, tokius kaip japoniški „manga“ filmai (4 pav.).

**4 pav. Manga filmo kadras. Kairėje – sugeneruota objekto linija. Dešinėje – sukurtas, nuspalvintas kadras. Vaizdas iš [HKM19].**

Animacinių filmų spalvinimas automatiškai gali turėti teigiamų aspektų, tokių kaip darbo naštos sumažinimas spalvinant kiekvieną kadrą žmogui ir labiau techniškesni kaip vientisos spalvų schemos išlaikymas per visą filmą [HKM19]. Žinoma, nors užduotis spalvoti filmus atrodo ir skamba iš esmės taip pat kaip ir paprastus, vientisus kadrus, paveikslėlius, fotografijas, filmų spalvinimas atneša ir papildomas, naujas problemas. Viena tokių problemų gali būti vientisos spalvų schemos išlaikymas [HKM19]. Iš esmės tokią problemą galima apibūdinti paprastai – kaip užtikrinti spalvų vientisumą per visą filmą – pavyzdžiui, kad filmo

veikėjo švarkas, kuris buvo pilkas, vėliau filme arba net sekančiame kadre nebūtų, pavyzdžiui, žalias.

Svarbu atkreipti dėmesį, kad spalvų suteikimas paveikslėliui daug priklauso nuo interpretacijų – tai yra labai sunku pilnai užkrinti, kad bus atkurtos originalios spalvos, tokios, kokios buvo tuo metu kai paveikslėlis arba kitoks vaizdas buvo sukurtas. Paprastas pavyzdys yra istorinės, nespalvotos nuotraukos – kas gali užtikrinti, kad spalvos, panaudotos tai fotografijai nuspalvinti dirbtiniais neuroniniais tinklais arba kitomis technologijomis, tikrai buvo tos, kokios buvo tikrovėje? Pavyzdys spalvų interpretacijos pateiktas darbe [VK16], pagal kurio pateiktą pavyzdį su suknelėmis galima užduoti klausimą – jeigu istorinėje fotografijoje yra moteris su suknele, kaip galima nustatyti, kokia buvo jos tikra spalva, jeigu visos spalvos atrodo vienodai įtikinamai tikros? (5 pav.). Kaip aprašyta [VK16], autorius pamini, kad problema iš karto yra neturinti aiškių ribų, nes yra daugybė spalvų kurios gali būti interpretuojamas, tuo pasitelkdamas medžio lapų pavyzdį – kad lapai gali būti žali, geltoni arba rudi. Tikrą spalvą galima nustatyti tik tokiu atveju, jeigu toje aplinkoje buvo asmenys, kurie matė tikrąją objekto spalvą arba yra išlikusi tikra spalvota fotografija. Abi galimybės yra vienodai mažos, dėl to telieka tik interpretuoti spalvas.




**5 pav. Suknelė gali atrodo tikroviškai įvairiomis spalvomis – dėl to nežinant istorinio konteksto nustatyti tikrąją spalvą yra neįmanoma. Vaizdas iš [VK16].**

Verta paminėti, kad skirtingą spalvų interpretavimą galima įsitikinti praktiškai ir pačiam – užtenka panaudoti jau turimą nespalvotą paveikslėlį (1 pav.) ir jį nuspalvinti perduoti skirtingiems įrankiams. Pirmojo įrankio rezultatas duoda daugiau spalvotą rezultatą, bet labiau chaotišką (6 pav.), o sekantis įrankis duoda mažiau skirtingų spalvų, bet tikslesnį pagal objektų rėmus rezultatą (7 pav.)

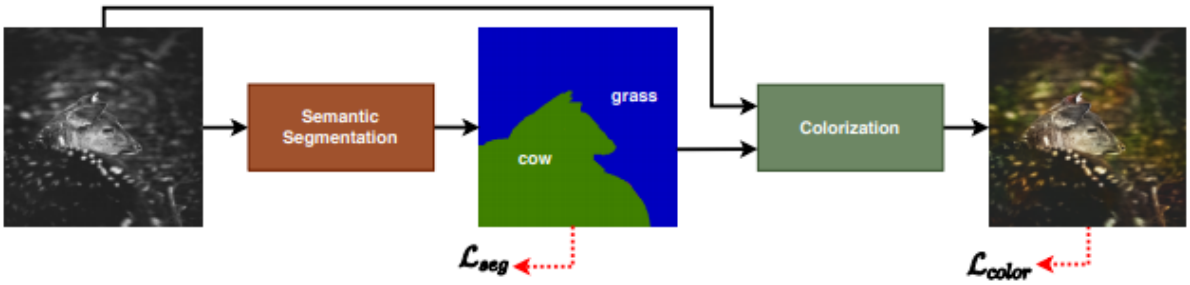

**6 pav. Nauja spalvų interpretacija. Daug mažiau dominuoja raudona spalva, atsirado didelės žalios spalvos zonos, taip pat žiedai, anksčiau buvę pilnai balti, dabar interpretuoti kaip turintys geltoną atspalvį.**

**7 pav.  Nauja, gan monotoniška interpretacija. Dominuoja balta ir žalia spalvos, trūksta ryškesnių spalvų.**

## 3.1. Dažniausiai kylančios problemos sprendžiant spalvinimo uždavinį

Jau iš karto yra matyti, kad spalvinimas naudojant dirbtinius neuroninius tinklus nėra paprasta užduotis, o sudaryta iš kelių sudėtingų dalių, norint išgauti kuo tikslesnį vaizdą. Kylančios problemos, apimančios spalvinimo iš juodai-balto paveikslėlio į spalvotą apima:

1. Objektų nustatymas paveikslėlyje – tai yra nustatymas kokie objektai yra atvaizduoti ir tinkamai surasti to objekto ribas ir pritaikyti pagal klasifikavimą atitinkamą, tinkamą spalvą. Pavyzdys gali būti ir pirmas paveikslėlis (1 pav.) – svarbu nustatyti, ties kuria vieta pasibaigia žiedlapis ir prasideda lapas tam, kad juos būtų galima nuspalvinti atskirai kaip atskirus objektus, kurių vienas – ryškiai raudonas arba baltas, kitas – žalias. Tai yra visiškai kitokio tipo problema negu tiesiog objekto spalvinimas. Ir jos sprendimo būdai gali priklausyti ir nuo būdo, kaip vėliau bus apdorojamas paveikslėlis. Pavyzdžiui, vienas iš būdų, naudojamas tokio tipo uždaviniams – segmentavimas, tai yra paveikslėlio dalinimas yra konkrečias dalis. Kaip pateikiamas pavyzdys [MLA+20], pirma paveikslėlis padalinamas į kelias dalis, ir tada, pagal konkrečias dalis spalvinamas ( 8 pav.)




**8 pav. Kaip pateikiamas [MLA+20], pirma paveikslėlis išdalinamas į reikšmines dalis („*semantic segmentation*“), ir toliau spalvinamas. Vaizdas iš [MLA+20].**

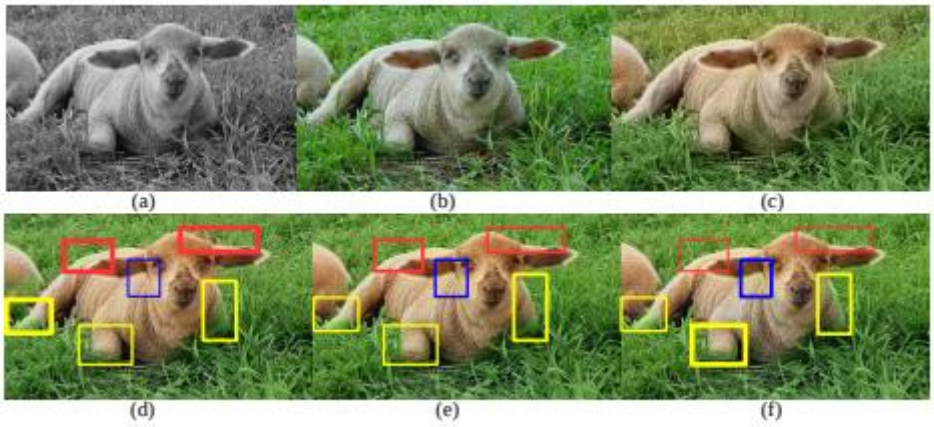
Aišku, toks vien tuo galima neapsiriboti ir tokį dalinimą toliau automatizuoti, pasirenkant kitokį kelią, kaip aprašoma [SRY+11], išskirtą objektą panaudoti kaip paiešką internete, pagal jį susirasti panašiausią į norimą arba netgi panaudoti kelis variantus, ir tokiu būdu sukurti naują, spalvotą vaizdą (9 pav.). Kaip rašoma [SRY+11], tokiu būdu pats vartotojas galėtų išsirinkti tinkamiausią vaizdą iš gautų.


**9 pav. Pagal [SRY+11] atpažįstamas objektas, tuomet pagal jo pavadinimą internete randami atitikmenys dešinėje ir gaunami keli variantai spalvoto objekto. Vaizdas iš [SRY+11].**

2. Atitinkamos spalvos parinkimas – skirtumas tarp aiškių ir interpretuojamų spalvų. Pavyzdys gali būti tie patys žiedai ir lapai – absoliučioje daugumoje atvejų, lapai tikrai negali būti raudoni arba oranžiniai, arba kiti objektai, antai dangus – tikrai negali būti žalias, tai iš kartų atrodytų neteisingai, todėl šiuose objektuose susiaurėja spalvų interpretacijos kelias. Kiti objektai yra gerokai tolerantiškesni interpretacijai, kaip pavyzdys 3 pav. – suknelė gali puikiai atrodyti ir bet kokios spalvos, arba labiau kasdieniškas pavyzdys – automobilio spalva, kuri gali gerai atrodo irgi iš gerokai daugiau atspalvių. Į šią problemą galima pažiūrėti ir iš kitos, meninės, pusės. Pavyzdžiui, didžioji dalis objektų spalvinama viena spalva, tačiau kas jeigu menininkas, naudojantis sukurtą programą nori ją panaudoti savo meniniais tikslais, tai yra pavyzdžiui nespalvotoje nuotraukoje esantį automobilį nuspalvinti ne viena, o keliomis spalvomis? Dar viena klaida, kuri aprašoma [RPA16], yra iš esmės atvirkštinė spalvų prieš tai buvusiai. Tai yra, jeigu dirbtinis neuroninis tinklas išmoksta pagal mokyklinį autobusą, kad autobusai yra geltoni, tai ir sekantys automobiliai, kurie bus panašūs į autobusus, bus geltoni, tuo gali išsiduoti kaip nerealistiškai nuspalvinti.
3. Spalvų taikymo problemos. Net ir sėkmingai radus objektus ir priskyrus atitinkamas spalvas ne visada pavyks išvengti problemų. Viena dažniausiai sutinkamų problemų – spalvos išsiliejimas (angl. „*colour bleeding*“). Ši problema trumpai apibūdinama kaip vieno objekto spalvos susiliejimas su kito objekto spalvomis. Net ir šiame darbe naudotuose pavyzdžiuose galima susidurti su šia problema. Tokios problematiškos vietos pažymėtos raudonu apskritimu (10 pav.)


**10 pav. Probleminės vietos paveikslėlyje, kuriame pažymėtose vietose aiškiai matyti kitų objektų „užlipimas“ ant kitų, kuris iš karto išduoda, kad paveikslėlis yra netikras.**

Problema yra sudėtinga išspręsti ir pati iš savęs yra verta tyrimų objektas, nes ją išsprendus galima gerokai pagerinti spalvotų paveikslėlių tikroviškumą. Šiai problemai spręsti sukurti sprendimai kaip tą galima pritaikyti gerinant galutinę kokybę. Vienas tokių – [JLC+18] (11 pav.)




**11 pav. Vienas iš spalvų išsiliejimo sprendimo būdų. Vaizdas iš [JLC+18].**

Toks sprendimo aprašytas [JLC+18], leidžia vaizdo pikselių tikslumu sukurti daug tikroviškesnius vaizdus, pavyzdžiui, kaip nurodyta nuotraukoje (11 pav.) geltonuose stačiakampiuose sumažinti susiliejimą su aplinkos objektais, raudonuose, rasti teisingai objektų kraštus, o mėlynuose teisingai nustatyti objekto kontekstą – tai šiuo atveju kad mėlyname stačiakampyje yra ne žolė, o tas pats objektas avis. Tokiu būdu, pikselių tikslumu, galima pilnai automatiškai ir labai kokybiškai sutvarkyti spalvas paveikslėlyje, ir tokiu būdu sukurti realistinį vaizdą.

Apibendrinant, bendrą spalvinimo procesą – nuo pačio paveikslėlio apdorojimo iki spalvų pritaikymo galima padalinti į kelias ar dar daugiau procesų, kurie veikia visumoje, siekiant bendro rezultato. Kiekvienas iš tų procesų, arba proceso dalių, turi savas kylančias problemas ir savus sprendimo būdus. Nors šioje dalyje buvo išskirtos trys problemos, tai tikrai nėra baigtinis sąrašas. Galima teigti, kad išsprendus vienas problemas ir toliau gerinant technologijas, neišvengiamai kils ir reikės spręsti naujas problemas.

## 3.2.Spalvų pritaikymo paveikslėlyje metodai

Net pats spalvinimo procesas panaudojant dirbtinius neuroninius tinklus gali būti įvairus. Kiekvienas iš kelių turi savų pliusų ir minusų, taip pat skirtingus žmogaus įsikišimo lygius – nuo pilnai nuo žmogaus įsikišimo priklausomus spalvinimo būdus iki pilnai automatinių, tai yra pilno spalvų apdorojimo be žmogaus įsikišimo. Iš esmės spalvinimo problemą, kaip aprašoma pavyzdžiui [DT17] ir kuriuo remiantis, galima padalinti į keletą skirtingų sprendimo kelių:

1. Užuominomis paremtas.
2. Pavyzdžiu paremtas.
3. Mokymu paremtas.

Kiekvienas iš šių būdų skiriasi savo architektūra ir logika, kaip pasiekti tikslą – iš nespalvoto paveikslėlio arba nuotraukos sukurti kuo tikroviškesnį spalvotą paveikslėlį arba nuotrauką. Visi nurodyti metodai turi savo teigiamas ir neigiamas dalis, kurios aprašytos 3.2.1 – 3.2.3 skyreliuose.

### 3.2.1. Užuominomis paremtas spalvų pritaikymo būdas

Spalvinimas panaudojant žmogaus brūkštelėjimu (angl. *scrible*) arba kitus spalvos pavyzdžius arba užuominomis paremtas – kaip aprašoma [ADY], pagal nedideles žmogaus nurodytas spalvos užuominas ant paveikslėlio, toliau naudojant dirbtinius neuroninius tinklus galima sukurti aukštos kokybės, tikroviškai atrodančią nuotrauką su vidutiniu žmogaus įsikišimu.

Tokiam būde nespalvota nuotrauka yra papildoma su spalvos brūkšneliais, kuriuos dirbtinis neuroninis tinklas panaudos kaip užuominą, kaip turi atrodyti objektas, ant kurio jis yra nupieštas. To pasekmė – spalvota nuotrauka ar paveikslėlis (12 pav.).

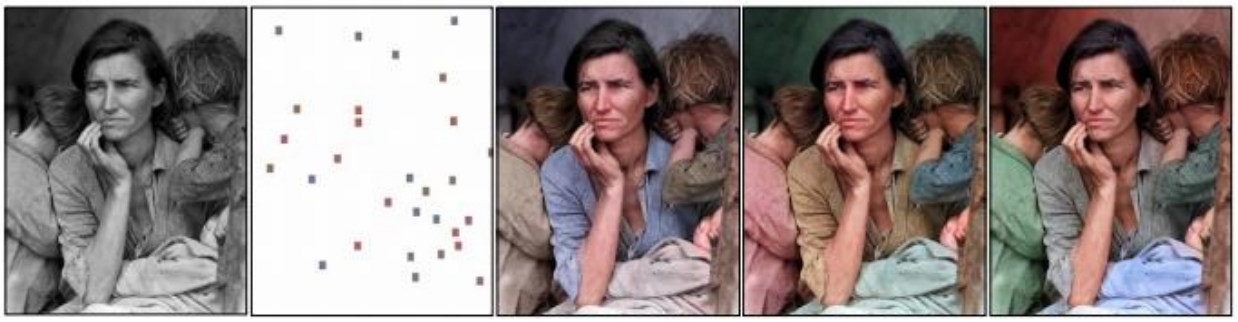
**12 pav. Kairėje – nuotrauka su užuominomis. Dešinėje – galutinis tokio tinklo rezultatas. Vaizdas iš [ADY].**

Pagrindiniai pliusai tokio spalvinimo kelio – galimybė sumažinti neteisingų interpretacijų nurodant konkrečias spalvas ir objektus, kaip jie turi atrodyti. Kitaip tariant, pagrindinis darbas tiksliai parinkti objektui spalvą lieka žmogui ir jis gali, ypač jeigu žmogus patyręs, labai tiksliai pateikti norimą spalvą, todėl programai lieka atitinkamai pritaikyti spalvą.

Minusas – šis kelias nėra pilnai automatinis, gali reikalauti pakankamai didelio žmogaus įsikišimo į spalvinimo procesą, ypač jeigu asmuo nepatyręs. Be to, nors ir žmogaus įsikišimas yra vidutiniškas, jis vis tiek negali apsaugoti nuo spalvinimo klaidų, tokių kaip spalvų išsiliejimas, blogo objektų ribų nustatymo ir panašių klaidų, kurios kyla naudojant dirbtinius neuroninius tinklus. Be to, kaip aprašoma [RJP+17] net tokiems objektams, kurių spalvų interpretacijų yra nedaug, tokių kaip žolė (žalia spalva ir atspalviai) arba dangus (mėlyna spalva

ir atspalviai) vartotojas vis tiek turi nustatyti spalvą. Galiausiai, jeigu žmogus nėra patyręs, jam gali nepavykti duoti tinkamas arba nuosekliai tinkamas spalvas ir tokiu atveju net teisingai atlikus spalvinimą paveikslėlis gali atrodyti neteisingai.

Aišku, tam tikrus šio būdo minusus galima mažinti ir patį spalvinimo būdą derinti su kitai, pavyzdžiui pilnai automatiniai, apmokytais algoritmais, tai yra panaudoti vartotojo įsitraukimą ir mokytus tinklus. Kaip aprašoma [RJP+17], galbūt galima panaudoti tai kas geriausia iš abiejų pasaulių – ir vartotojo įsitraukimą ir dideliais kiekiais duomenų apmokytus tinklus. Toks būdas naudoja žmogaus padėtas spalvos užuominas (angl. „*hint*“) ir apmokytą tinklą, kad gauti labiau tikrovišką paveikslėlį (13 pav.)



**13 pav. Kaip aprašoma [RJP+17] kairėje – nespalvota, toliau – vartotojo pateiktos užuominos ir trys siūlymai pagal apmokytus tinklus ir pasiremiant vartotojo užuomina. Vaizdas iš [RJP+17].**

Apibendrinant, tai yra vienas paveikslėlių spalvinimo būdų, kuriuo asmuo gali konkrečiai parinkti spalvas objektams, paliekant užbaigiamąjį darbą programinei įrangai. Tačiau, bendrai sudėjus, kadangi būdas reikalauja gan nemažo žmogaus įsitraukimo ir toks būdas negali pilnai užtikrinti nuo paprastų dirbtinių neuroninių tinklų klaidų, toks būdas gali neatrodyti tinkamas daliai spalvinimo užduočių.

### 3.2.2. Pavyzdinio paveikslėlio paremtas spalvų pritaikymo būdas

Spalvų stiliaus perkėlimas ant nespalvoto paveikslėlio arba pavyzdžiu paremtas – tai yra neuroninis tinklas, kuris paima stilių, tai yra objektų spalvas iš įvesties nuotraukos arba paveikslėlio ir gali  pagal tuos pačius objektus pritaikyti tą patį stilių (14 pav.). Toks spalvinimo būdas aprašytas darbe [TMK02], taip pat [BFZ+14] ir [RAD+12], [JYL+17].

**14 pav. Stiliaus perkėlimas iš pirmo paveikslėlio į nespalvotą antrą paveikslėlį. Galutinis rezultatas – trečiasis paveikslėlis. Vaizdas iš [TMK02].**

Toks būdas apima dvi įvestis, kaip aprašoma [TMK02], tai yra nespalvota nuotrauka arba paveikslėlis, kurią reikia spalvinti ir  pavyzdinis paveikslėlis, tai yra žmogaus duodama spalvota nuotrauka arba paveikslėlis. Antrojo pagrindu bus spalvinamas pirmas, juodai-baltas paveikslėlis. Tokiu būdu, spalvos perkeliamos nuo objekto prie nespalvoto, bet pagal supratimą, tokio pačio objekto, šiuo pateiktu pavyzdžiu, nuo medžio prie medžio, nuo žolės prie žolės, nuo dangaus iki dangaus ir panašiai. Galutinis rezultatas – spalvota nuotrauka arba paveikslėlis (7 pav.). Kaip aprašoma [TMK02] – tokiu būdu stengiamasi sumažinti žmogaus įsikišimo į spalvinimo procesą, ir tas dažniausiai sumažinama iki pavyzdinio paveikslėlio pateikimo. Dar toks būdas perkelti spalvas gali būti vadinamas „*Example-based colorization*“ [BFZ+14].

Toks būdas irgi turi savus pliusus ir minusus. Pagrindinis pliusas – beveik pilnai automatinis spalvinimas – reikia palyginti mažesnio žmogaus įsikišimo pateikiant pavyzdinę spalvų schemą iš spalvoto paveikslėlio, pagrindinius objektus galima sėkmingai nuspalvinti be didesnių pastangų, įvairiais rakursais ir kaip rašoma, [BFZ+14], leidžia vartotojui tam tikrą lankstų būdą išvengti neaiškių spalvinių vietų, jeigu jis suteikia pakankamai gerą paveikslėlį  (15 pav.)

**15 pav.  Skirtingų gėlių spalvų perkėlimas pagal pavyzdį. Vaizdas iš [BFZ+14].**

Minusas – padidėja klaidų tikimybė, jeigu trūksta objektų spalvinių nuorodų. Pavyzdys, jeigu nespalvotame paveikslėlyje yra nenumatytas objektas, pavyzdžiui kaminas, tai pagal duotą spalvų schemą jis gali būti neteisingai suinterpretuotas ir nuspalvintas, tarkim, žaliai arba blogiausiu atveju likti nespalvintas. Tokiu būdu jis atrodo labai nenatūraliai ir paveikslėlis gali atrodyti neteisingas.

Tokiu atveju toks spalvinimo būdas, pagal [BFZ+14], tokios klaidos gali būti pašalintos papildomu žmogaus darbu – tai yra surasti arba nufotografuoti paveikslėlį toje pačioje vietoje arbe rasti paveikslėlį, kuriame esantys objektai kuo labiau atitiktų esančius nespalvotame paveikslėlyje.

Apibendrinant, toks spalvinimo būdas jau reikalauja mažiau žmogaus įsikišimo,

padidėja automatizavimas, tačiau jo efektyvumas priklauso nuo žmogaus darbo pritaikant pavyzdį spalvų – kitaip tariant, koks geras bus pavyzdys spalvų schemos, kuri turi būti pritaikyta kitam paveikslėliui.

### 3.2.3. Apmokymu paremtas spalvų pritaikymo būdas

Pilnai automatinis spalvinimas arba apmokymu paremtas – pagrindinis būdas, tiriamas šiame magistro baigiamajame darbe. Kaip aprašoma [RPA16], tokio būdo tikslas yra pilnai automatinis spalvinimo būdas, kuris gali atkurti kuo realistiškesnes ir gyvybingesnes spalvas. Iš [RPA16] aprašyto, sukurto ir išbandyto dirbtinio neuroninio tinklo esmės toks būdas veikia tuo kad jam pateikiamas paveikslėlis be spalvų ir neuroninis tinklas grąžina paveikslėlį su spalvomis – be jokio žmogaus įsikišimo. Dirbtinis neuroninis tinklas, aprašytas [RPA16], apmokomas daugiau nei 1 milijonu paveikslėlių ir tuo geba atkurti realistinį spalvų vaizdą (16 pav.).

**16 pav. Duotas pavyzdys – kairėje – atkurtos spalvos, dešinėje – tikroji nuotrauka[1].**

Toks algoritmas iš apmokytų duomenų mėgina spėti kiekvieno pikselio galimą spalvų paletę – pavyzdžiui, obuolys gali būti raudonas, žalias, bet greičiausiai nebus mėlynas [RPA16].

Tačiau net ir ganėtinai lengvai suprantamas tokio uždavinio sprendimo būdas tobulėja kartu su mokslu ir technologijomis, kartu su jomis atsiranda ir naujos metodikos sprendimui pasiekti. Pavyzdžiui, tokiam uždaviniui spręsti gali būti pritaikyti nauji dirbtiniai neuroniniai tinklai, tokie kaip generuojantys priešiški neuroniniai tinklai (angl. *Generative Adversiarial Network*, GAN). Pirmą kartą pasiūlyti 2014 metai, jie jau turi praktinio pritaikymo galimybę vaizdinėje medijoje, pavyzdžiui pagal apmokytą duomenų aibę sugeneruoti gyvai neegzistuojančio žmogaus veidą.[2]. Toks tinklas iš esmės paremtas dviejų konkuruojančių neuroninių tinklų idėja – vienas yra generatoriau neuroninis tinklas, kitas yra diskriminatoriaus neuroninis tinklas. Jo veikimo idėja yra ta, kad generatoriaus tinklas apmokomas kurti kuo

[1] Vaizdas iš https://richzhang.github.io/colorization/resources/imagenet_comparison.html
[2] https://thispersondoesnotexist.com/

tikresnius duomenis ir artimesnius realybei duomenis, o diskriminatoriaus tinklas apmokomas ir veikia tuo, kad jis tikrina tuos duomenis, tokiu būdu abu tinklai veikia iki tol kol generuojantis tinklas kuria duomenis panašius į realybėje esančius duomenis. Todėl verta paminėti, kad tokia idėja – panaudoti duomenis generuojančius tinklus, rado savo panaudojimo būdą ir vaizdų spalvų generavime. Tokį sprendimo būdą pasiūlė [KEM18], kuriame dviejų konkuruojančių neuroninių tinklų – generuojančio ir diskriminatoriaus, atlieka konvoliuciniai neuroniniai tinklai, tiesiog vienas apmokomas ir bando generuoti kuo tikroviškesnes spalvas nespalvotoje nuotraukoje, kitas apmokomas kuo ir bando atspėti ar sugeneruotas vaizdas netikras. Iš esmės darbas turi būti vykdomas iki tol, kol nebebus įmanoma atspėti ar vaizdas nėra netikras, ir tokio metodo rezultatas gali būti pilnai spalvotas paveikslėlis, kaip ir visų kitų metodų (17 pav.).

**17 pav. Generuojančio priešinančio tinklo rezultatas – kairėje nespalvota nuotrauka, viduryje originalas ir dešinėje yra generuojančio tinklo sukurtų spalvų nuotrauka. Vaizdas iš [KEM18].**

Tokio būdo – pilnai automatinio spalvų generavimo, pagrindinis pliusas – nėra jokio žmogaus įsikišimo. Galima pilnas spalvų atkūrimo automatizavimas, pateikiant algoritmui tik nespalvotą, apdirbimui paruoštą paveikslėlį ir rezultate gaunant realistinį vaizdą. Iš esmės tokio būdo pagrandinis limitas yra jo apmokymo duomenų gausa. Kitaip tariant, kuo gausiau apmokomas dirbtinis neuroninis tinklas, tai tuo geresnį vaizdą galima gauti. Sutaupoma daug žmogaus darbo laiko, o panaudojimo būdai – platūs ir sunkiai aprašomi, tokie kaip nespalvotų nuotraukų spalvinimas arba žmogaus meninė išraiška yra tik vieni iš jų.

Pagrindiniai minusai – sudėtingumas, nes toks tinklas turi būti apmokamas dideliais duomenų kiekiais kad būtų kuo tikslesnis ir vis tiek lieka gan didelė klaidų tikimybė. Tokios klaidos kaip spalvų išsiliejimas ir panašios gali būti taisomos keliais skirtingais būdais, pavyzdžiui didinant duomenų kiekį dirbtinių neuroninių tinklų mokyme, kad tinklas pasidarytų dar tikslesnis, arba pritaikyti kitokius sprendimo būdus, kaip [JLC+18], kuriame aprašoma, kaip pikselių tikslumu išspręsti šią problemą.

Apibendrinus, toks spalvinimo būdas yra sudėtingesnis, tačiau perima iš žmogaus daug darbo kurį žmogui gali būti per sunku atlikti, ir pakankamai gerai išplėtotas tinklas ir tinkamai pasirinktas sprendimo būdas gali padėti sukurti labai tikroviškus, sunkiai nuo realistiškų atskiriamus vaizdus.

### 3.3. Teorinės dalies apie spalvinimo būdus apibendrinimas

Apibendrinant įvairius metodus, spalvų taikymą ant nespalvotų nuotraukų arba paveikslėlių galima padalinti į kategorijas, kurios naudoja skirtingus metodus tikslui pasiekti. Tuos skirtingus būdus skiria įranga, kurią jie naudoja, žmogaus intervencijos lygis į bendrą spalvinimo procesą, sudėtingumo lygis kaip jį galima įgyvendinti, taip pat problemos, kurios lydi skirtingą spalvinimo būdą. Galima teigti, kad netgi galutinis rezultatas skiriasi tarp visų skirtingų būdų. Verta paminėti, kad nors čia buvo išskirti trys skirtingi būdai, kartais skirstymas gali skirtis, kaip antai tarp 2 skirtingų būdų arba 3 skirtingų būdų. Taip pat įmanomas ir mišrus būdas, kuriame gali būti panaudoti keli metodai pasiekti tikslui.

## 4. Analizuojamas dirbtinis neuroninis tinklas

Prieš atliekant pagrindinį analizės darbą, reikia pasirinkta tam skirtą įrankį - dirbtinį neuroninį tinklą, kuris gali nuspalvinti paveikslėlius. Kadangi spalvinimo uždavinys yra labai kompleksiškas ir sudėtingas, tai praktiškesnis pasirinkimas būtų pasinaudoti jau sukurtu, atviro kodo paremtu neuroniniu tinklu, nes sukurti tokį tinklą nuo pradžių užimtų nepagrįstai daug laiko. Taigi, pasirenkant atviro kodo tinklą, kad būtų galima atlikti uždavinius, jis turi prieš tai atitikti tam tikrus kelis kriterijus:

- Tinklas turi būti atviro kodo, kad būti galima atlikti pakeitimus skirtingoms analizėms.
- Tinklas turi turėti savo apmokymo dalį, kad tinklą būtų galima išmokyti iš naujo su kodo pakeitimais ir tokiu būdu išanalizuoti pakeitimus, ar jie turėjo kokį nors poveikį kokybei.
- Tinklas turi turėti savo jau apmokytą modelį, su kuriuo būtų galima palyginti rezultatus.

Taigi, turint pradinius kriterijus, galima išrinkti dirbtinį neuroninį tinklą. Nors dirbtinių neuroninių tinklų, kurie, pavyzdžiui, atlieka klasifikavimo užduotis, yra gana nemažai, dirbtinių neuroninių tinklų, kurie atlieka spalvinimo uždavinį, nėra tiek daug ir pasirinkimas nėra didelis. Galiausiai, atmetus keletą variantų, pasirinkimas buvo vykdomas iš:

- Interactive Deep Colorization In PyTorch[3].

[3] https://github.com/richzhang/colorization-pytorch

- Colorful Image Colorization[4].
- Automatic Image Colorization[5].

Analizėms atlikti buvo pasirinktas trečiasis variantas, nes be to, kad atitinka prieš tai nurodytus kriterijus, jis taip pat gali pasiūlyti ir kitus naudingus darbui atlikti privalumus:

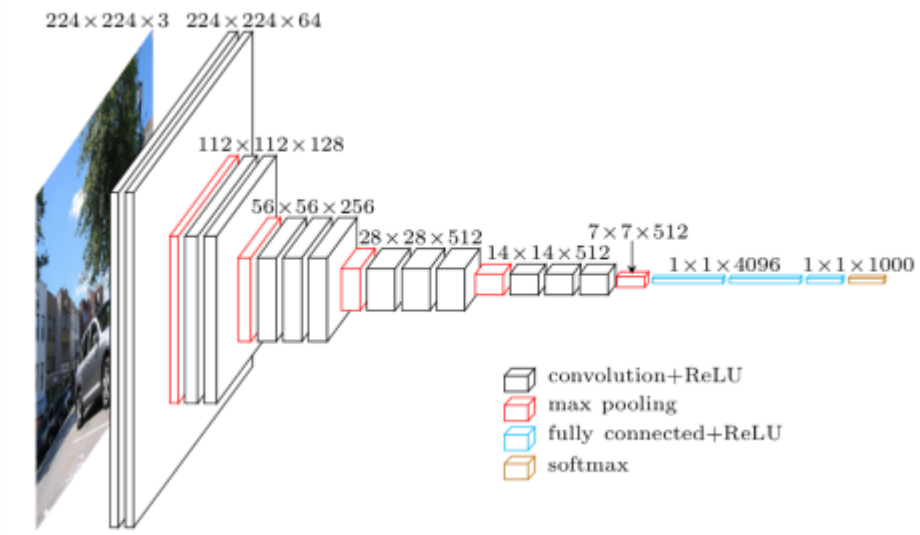
- Gerai struktūrizuotas, sukurtas iš kelių, skirtingus darbus atliekančių dalių — atsiveria galimybė išbandyti, ar pakeitus vieną programinę dalį su kita, galima pagerinti rezultatus, neperdarant per daug pačio tinklo.
- Viskas vykdoma automatiškai — žmogaus įsikišimas minimalus, tai yra apmokymas vykdomas, kol yra sustabdomas automatiškai ir nuspalvinimas vykdomas irgi be žmogaus instrukcijų, tik iš to, ką tinklas yra išmokęs.
- Mokymo objektas gali būti tiesiog paprastas spalvotas paveikslėlis.
- Neuroninio tinklo rezultatai yra neblogi jau pasinaudojant tinklu paprastai, „be keitimų".
- Naudoja pakankamai naujas, gausiai dokumentacija aprašytas programines bibliotekas ir struktūras, tokias kaip *Tensorflow*.
- Labai paprastas paleidimas - nereikalinga sudėtinga ar brangi techninė ir programinė įranga, pavyzdžiui, veikia *Windows 10* sistemoje, nereikia diegti kitos sistemos.

Taigi, apžvelgus pliusus ir minusus pasirinktų dirbtinių neuroninių tinklų, toliau analizės bus vykdomos su „*Automatic Image Colorization*" dirbtiniu neuroniniu tinklu. Tai yra dirbtinis neuroninis tinklas, sukurtas *GitHub* naudotojo *Chong Guo* pavadinimu „*Automatic Image Colorization*"[6], ir kuris paremtas kito naudotojo – *Ryan Dahl* aprašyta dirbtinio neuroninio tinklo architektūra[7] (20 pav.). Dirbtinis neuroninis tinklas yra atviro kodo – jį gali išbandyti visi kurie turi galimybę. Pagal aprašytą ir sukurtą architektūrą – pats tinklas naudoja ir yra paremtas dviejų mokslinių darbų apie dirbtinius neuroninius tinklus– *VGG* klasifikatoriumi [SZ15] ir hiperstulpais [HAG15].

*VGG* klasifikatorius – tai yra kartais klasikine vadinama konvoliucinio dirbtinio neuroninio tinklo architektūra, kuri pasižymi dideliu tikslumu norint atpažinti objektus vaizduose. Pati tokio konvoliucinio neuroninio tinklo architektūra pirmą kartą pristatyta autorių *Karen Simonyan* ir *Andrew Zisserman* ir aprašyta „*Very deep convolutional networks for large-scale image recognition*" [SZ15]. Kaip minėjo patys autoriai, jų architektūra darbo pristatymo

[4] https://github.com/richzhang/colorization
[5] https://github.com/Armour/Automatic-Image-Colorization/tree/tf-1.12
[6] https://github.com/Armour/Automatic-Image-Colorization/tree/tf-1.12
[7] https://tinyclouds.org/colorize/

metu gerokai skyrėsi nuo prieš tai buvusių – tinklas pasižymėjo smulkiais filtrais ir mažėjančia sluoksnių architektūra – kaip toks tinklas atrodo galima pamatyti pagal 18 pav.

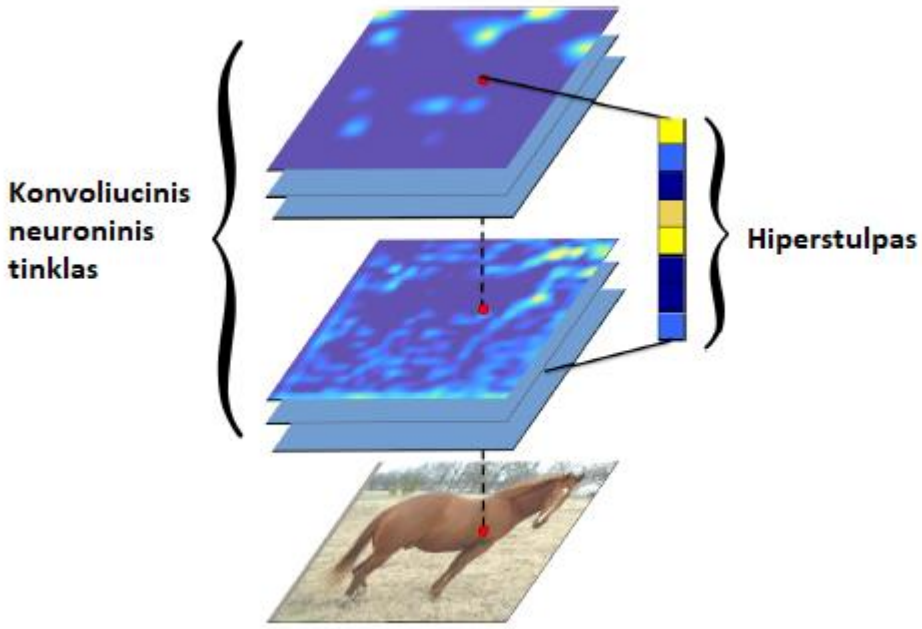



**18 pav. VGG tinklo architektūra atvaizduota grafiškai[8].**

Pats tinklas buvo pritaikytas dirbti su 224 pikselių dydžio paveikslėliais. Kuomet buvo apmokomas *ImageNet*[9] paveikslėlių duomenų bazės, autoriams pavyko pasiekti 92,7 % klasifikavimo tikslumą. Šis klasifikavimo neuroninis tinklas yra viena iš sudedamųjų analizuojamo darbo dalių ir skirta kuo tiksliau atpažinti vaizdus, esančius paveikslėliuose.

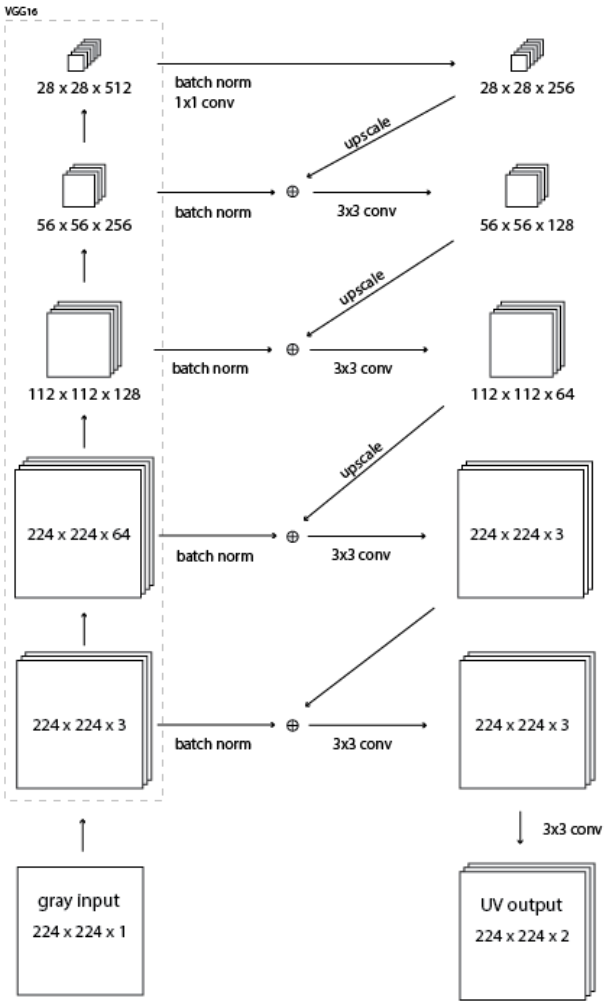
Antroji svarbi analizuojamo tinklo dalis – hiperstulpas (angl. „*hypercolumn*“). Tai yra dirbtinio neuroninio tinklo architektūra, kada informacija išsaugoma keliuose sluoksniuose. Apie hiperstulpus dirbtiniuose neuroniniuose tinkluose pirmą kartą parašyta *Bharat Hariharan, Pablo Arbelaez* ir *Ross Girshick* darbe „*Hypercolumns for Object Segmentation and Fine-grained Localization*“ [HAG15]. Hiperstulpų veikimas paremtas tuo, kad didžioji dauguma dirbtinių neuroninių tinklų informacijos pateikimas naudoja paskutinį sluoksnį, tačiau informacija tame sluoksnyje gali būti vaizdiškai per daug stambi kad atlikti tikslumo reikalaujančius veiksmus. Tam tikslui pasitelkiami pirmieji sluoksniai, kurie yra daug detalesni tačiau juose trūksta bendro vaizdo klasifikavimo. Hiperstulpai padeda spręsti šią problemą suderinant geriausius sprendimus iš viso neuroninio tinklo. Tokiame tinkle pikselis veikia kaip vektorius, kuris aktyvuoja visus virš jo esančius sluoksnius, kuriuose būna išsaugota reikalinga informacija – tokiu atveju tai leidžia pagerinti pačios dirbtinio neuroninio tinklo išvesties tikslumą.

[8] Vaizdas iš https://neurohive.io/en/popular-networks/vgg16/
[9] https://image-net.org/




**19 pav. Hiperstulpo schema – pikselis paveikslėlyje aktyvuoja visus virš jo esančius sluoksnius. Vaizdas iš [HAG15].**

Sujungus *VGG* ir hiperstulpo tinklus, buvo sukurta dirbtinio neuroninio tinklo architektūra, kurioje vietoje klasifikavimo užduočių vykdomas paveikslėlių spalvinimas, pasitelkus objektų atpažinimą. Panaudotas *VGG* tinklas kiekviename sluoksnyje iš apmokyto hiperstulpo gauna spalvinę informaciją iki gaunamas galutinis spalvotas paveikslėlis.




**20 pav. Dirbtinio neuroninio tinklo architektūra[10].**

Pats tinklas įgyvendintas pasitelkiant *Tensorflow* dirbtinių neuroninių tinklų programavimo biblioteką ir *Python* programavimo kalbą. Pačiame tinkle naudojamas *VGG16* apmokytas klasifikavimo modelis, skirtas klasifikuoti objektus. Pavadinime skaičius 16 reiškia, kad tinklą sudaro 16 sluoksnių. Tinklą apmokyti yra paprasta – bet koks paveikslėlis gali tapti mokymų aibės dalimi. Tinklas vykdo spėjimą dvejoms YUV spalvų formato dalims - mėlynumo (U dalis) ir raudonumo (V dalis). Iš nespalvoto paveikslėlio perimama spalvos intensyvumo (Y) dalis. Paveikslėlio spalvų informacija yra sujungiama ir prieš išsaugant, ji yra konvertuojama į RGB (angl *.red, green, blue* - raudona, žalia, mėlyna) formatą ir paveikslėlis gali būti išsaugomas.

Analizuojamas dirbtinis neuroninis tinklas, toks kokia jo versija yra įkelta atviro kodo svetainėje *GitHub,* neatlikus jokių programinio kodo pakeitimų, kartu su prieš tai išmokytu dirbtinio neuroninio tinklo modeliu, nuo šiol bus vadinamas originaliu tinklu ir jis tampa pagrindu toliau atliekant analizes ir atliekant palyginimus su kitomis analizėmis.

[10] Vaizdas iš https://tinyclouds.org/colorize/

Apibendrinant, analizės darbui atlikti pasirinktas tinklas paremtas dvejais moksliniais darbais [SZ15][HAG15] apie dirbtinius neuroninius ir tinklus ir jų galimybėmis. Pasirinktame dirbtiniame neuroniniame tinkle įgyvendintos aprašytos architektūros ir sprendimai, kurie buvo kartu suintegruoti sukuriant naują neuroninį tinklą kurio paskirtis – išmokti objektų spalvas ir jas atkurti nespalvotiems paveikslėliams.

# 5. Vaizdų vertinimo kriterijai ir metodai

Viena svarbiausių mokslo tiriamojo darbo dalių yra gautų rezultatų vertinimas. Vertinant rezultatus, galima vertinti ar pavyko pasiekti tikslus ir padaryti išvadas. Šiame darbe pagrindinis darbas vyksta su vaizdais – tai yra su spalvotais, dirbtinio neuroninio tinklo apdorotais vaizdais. Vertinimas yra kaip gerai pavyko dirbtiniam neuroniniam tinklui atlikti savo darbą. Vaizdų vertinimą galima padalinti į dvi dalis – subjektyvų ir objektyvų vertinimą.

Subjektyvų vertinimą sudaro žmonių nuomonė į matomus vaizdus. Tokiu atveju žmonėms yra duodamas vaizdas ir žmonės vertina kiek tikroviškas toks vaizdas yra. Toks vertinimas turi savo pliusų ir minusų. Tokio vertinimo teigiamos pusės yra:

- Lankstesnis vertinimas – nuspalvinti objektai gali būti įvairių spalvų (pavyzdžiui, automobilis gali būti raudonas arba mėlynas) ir tai neįtakos žmonių nuomonės – kelios spalvos jam gali būti vienodai įtikinamos.
- Tikroviškas vertinimas – pats žmogus geriausiai žino, koks vaizdas jam atrodo tikroviškiausias.
- Žmonėms galima duoti atlikti įvairius testus su vaizdais – leisti palyginti vaizdus, trumpam parodyti kelis vaizdus ir leisti pasirinkti tą kuris jam atrodo netikroviškas ir panašiai.

Nors žmonių subjektyvus vertinimas atrodo paprastas ir suprantamas, jis taip pat turi ir tam tikras neigiamas puses, tokias kaip:

- Neaiškūs vertinimo kriterijai – neaišku, kokiais kriterijais remiantis skirtyti rezultatus į tikroviškus ir netikroviškus ir kuo jie skiriasi.
- Nepastovūs rezultatai – žmonės nevienodai vertina rezultatus – kas vienam žmogui gali atrodyti tikroviška, kitam gali atrodyti netikroviška.
- Neaišku, kaip parodyti tikroviškumą – tai yra kuo skiriasi labai tikroviška nuo netikroviškos, ar tai yra skalės nuo 1 iki 5 vertinimas ar kitoks.

Šiame magistro baigiamajame darbe labiau tinkamas yra objektyvus vertinimas. Kadangi pagrindinis tikslas yra sukurti kuo tikslesnes spalvas nespalvotiems paveikslėliams, tai objektyvus vertinimas tam tinkamas dėl šių pagrindinių teigiamų kriterijų:

- Turint originalių spalvų nuotrauką ir nuspalvintą paveikslėlį, galima pasinaudojus metrikomis greitai palyginti kuo panašūs abu paveikslėliai ir tą parodyti skaitine išraiška.
- Analizę ir skaičiavimą galima atlikti iš karto po nuspalvinimo.
- Galima pasinaudoti jau suprogramuotomis galimybėmis panaudoti metrikas.
- Galima atlikti daug daugiau testų ir skaičiavimų negu naudojant žmonių nuomonės tikrinimą.

Nors siūlantis daug pliusų, šis būdas taip pat turi ir savų neigiamų pusių. Pagrindinė pusė – spalvų įvairovė. Tai yra tokia bėda, kad tam tikri objektai turi būti atkurti konkrečiomis spalvomis, nors teoriškai gali tikroviškai atrodyti keliomis spalvomis. Pavyzdys – automobilis, kuris gali būti bet kurios spalvos. Šios matematinės metrikos rodys labai didelę paklaidą, jeigu automobilis bus kitos spalvos, negu originalioje nuotraukoje. Ši klaida iš dalies gali būti suvaldoma, jeigu nuotraukose bus mažai objektų kurie gali būti kelių teisingų spalvų, pavyzdžiui dangus, kuris didžiąją dalį laiko bus mėlynas arba augalai, kurie beveik išskirtinai visi žali. Tokiu atveju galima daug tiksliau įvertinti kaip dirbtinis neuroninis tinklas atkuria spalvas.

## 6. Objektyvus vertinimas pasitelkiant matematines metrikas

Rezultatus galima įvertinti ir objektyviai, pasitelkus matematines formules. Kadangi dirbtinio neuroninio tinklo galutinis rezultatas yra spalvotas paveikslėlis, tai vienas iš gerų variantų yra palyginti galutinį spalvotą paveikslėlį su originaliu spalvotu paveikslėliu. Tokio palyginimo tikslas – pamatyti ir įvertinti, kaip tiksliai dirbtinis neuroninis tinklas atkūrė tikras paveikslėlio spalvas. Nors pilnai tiksliai atkurti spalvų nėra įmanoma dėl to kad nespalvotame paveikslėlyje nėra išsaugota buvusių spalvų informacija, pasitelkus matematines formules galima įvertinti kiek arti tiesos buvo dirbtinio neuroninio tinklo spėjimas. Vaizdų vertinimo srityje jau yra paruoštos ir naudojamos matematinės formulės būtent vaizdų ir kitos medijos kokybės vertinimui. Šiame magistro baigiamajame darbe bus naudojamos 5 skirtingos matavimo metrikos ir taip pat bus įvertinta kuri iš jų įvertina spalvinimo paklaidą geriausiai.

## 6.1. Vidutinė kvadratinė paklaida – MSE

Vidutinė kvadratinė paklaida (angl. *Mean square error*) arba trumpinys *MSE*, yra metrika, skirta įvertinti paklaidų kvadratų vidurkius tarp tikrosios reikšmės ir spėjamos reikšmės. *MSE* dažnai naudojama dirbtinių neuroninių tinklų mokyme kaip praradimų funkcija, taip pat dažnai naudojama statistikoje paklaidos skaičiavimui. Tokia formulė išreiškiama paprastai:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$$

Čia *n* – spėjimų skaičius, Y – tikroji reikšmė, Y – spėjama reikšmė. Tokiu atveju, turint vienodo dydžio ir vienodos struktūros paveikslėlius, bet su skirtingomis spalvomis, paklaidą galima skaičiuoti tarp tos pačios koordinatės pikselių, panaudojant RGB reikšmes ir paklaidą tarp jų, galiausiai pabaigoje jas sudedant ir išvedant bendrą vidurkį. Kaip atrodo *MSE* skaičiavimo rezultatai palyginus su originaliu vaizdu galima pamatyti 1 lentelėje.

1 lentelė – MSE skaičiavimo pavyzdžiai

| | | |
|---|---|---|
|  MSE = 0 (vienodi vaizdai) |  MSE = 123,5677 |  MSE = 442,6087 |

Šios metrikos programinį įgyvendinimą galima įdiegti prieš atliekant analizę, pasinaudojant jau esančiomis programavimo bibliotekomis. Viena tokių – *Python* programavimo biblioteka *skimage*. Joje esanti funkcija *measure.compare_mse* iš karto leidžia kaip parametrus duoti du vienodi dydžių paveikslėlius ir kaip atsakymą grąžina MSE rezultatą.

## 6.2. Signalo ir triukšmo santykis - PSNR

Signalo ir triukšmo santykis (angl. *peak-to-noise ratio*), trumpinys *PSNR* – matematinė metrika, dažnai naudojama vaizdų atkūrimo kokybės vertinimui, pavyzdžiui, atkūrus vaizdą iš jo kompresijos arba suspaudimo. Galutinis rezultatas nurodomas decibelais (*db*) ir kuo didesnė gaunama reikšmė – tuo geresnė paveikslėlio kokybė. Tokią metriką galima išreikšti taip:

$$PSNR = 10\left(\frac{MAX^2}{MSE}\right)$$

Čia MAX yra didžiausia pikselio reikšmė (pavyzdžiui, RGB atveju tai būtų 255) ir *MSE* – *MSE* formulės rezultatas tarp vienodų pikselių. Tokia metrika irgi skaičiuojama kiekvienam pikseliui iki gaunamas galutinis rezultatas. Kaip atrodo šios metrikos rezultatai galima pamatyti 2 lentelėje.

2 lentelė – PSNR skaičiavimo pavyzdžiai

| | | |
|---|---|---|
|  PSNR = ∞ [11](vienodi vaizdai) |  PSNR = 22,01879 |  PSNR = 21,6743 |

Kaip ir anksčiau minėtą metriką, norint įdiegti jos programinį įgyvendinimą užtenka pasinaudoti *Python* programavimo biblioteka *skimage*. Joje esanti funkcija *measure.compare_psnr* kuri iš karto leidžia pasinaudoti formule. Šioje funkcijoje kaip parametrus reikia duoti du vienodus dydžių paveikslėlius ir kaip atsakymą funkcija grąžina *PSNR* rezultatą.

## 6.3. Šaknis iš vidutinės kvadratinės paklaidos- RMSE

Šaknis iš vidutinės kvadratinės paklaidos (angl. *Root mean square error*), trumpinys *RMSE* – matematinė metrika, dažniausiai naudojama statistikoje ir matuojant spėjimų kokybę. Metrika taip pat naudojama apmokant neuroninius tinklus kaip praradimų funkcija. Tokią funkciją galima išreikšti kaip:

$$RMSE = \sqrt{MSE} = \sqrt{\sum_{i=1}^{n} \frac{(Y_i - \hat{Y}_i)^2}{n}}$$

Čia *MSE* – formulės galutinis rezultatas, *n* – spėjimų skaičius, Y – tikroji reikšmė, Y – spėjama reikšmė. Toks rezultatas dar toliau gali būti normalizuojamas ir pavadintas normalizuota šaknimi iš vidutinės kvadratinės paklaidos (arba NRMSE). Tokiu atveju normalizacija atliekama pagal formulę:

$$NRMSE = \frac{RMSE}{Y_{max} - Y_{min}}$$

Čia *RMSE* – metrikos galutinis rezultatas, $Y_{max}$ – didžiausia tikroji reikšmė, $Y_{min}$ - mažiausia tikroji reikšmė. Normalizacija gali būti atliekama jeigu norima palyginti skaičius iš skirtingų matavimo vienetų. Tokios metrikos geriausias rezultatas laikomas kuo mažesnis skaičius. Kaip atrodo tokios metrikos palyginimas su paveikslėlių kokybe galima pamatyti 3 lentelėje.

[11] Kadangi vienodų paveikslėlių MSE formulės rezultatas = 0, tai pagal PSNR, formulėje tokiu atveju gaunama dalyba iš nulio.

3 lentelė – RSME skaičiavimo pavyzdžiai

| | | |
|---|---|---|
|  |  |  |
| RMSE = 0 (vienodi vaizdai) | RMSE = 0,12723 | RMSE = 0,16701 |

Norint panaudoti šią metriką analizės metu, reikia pasinaudoti *skimage Python* bibliotekos funkcija *measure.compare_nrmse*, kurios parametrai – du paveikslėliai, tai yra originalių spalvų ir spėjamų spalvų paveikslėlis.

## 6.4. Struktūrinio panašumo indekso skaičiavimas – SSIM

Struktūrinio panašumo indekso skaičiavimas (angl. *struktural similarity index measure*) – trumpinys *SSIM* – matematinė metrika, naudojama būtent dviejų vaizdų palyginimui. Ji gali būti naudojama atrasti klaidos koeficientą po suspaudimo, po vaizdo pakeitimų ir kitokių struktūrinių vaizdo pakeitimų. Ši metrika skaičiuojama prieš tai paskaičiuojant kitas tris metrikas – šviesumo, kontrasto ir struktūros. Šviesumo matas yra vertinamas išvedant vidurkį iš visų pikselių ir yra skaičiuojamas:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Čia $X_i$ yra paveikslėlio X pikselis, $N$ yra pikselių skaičius. Toliau turi būti skaičiuojamas kontrastas, kuris yra standartinis nuokrypis visų paveikslėlio pikselių. Jis yra skaičiuojamas pagal formulę:

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N} (X_i - \mu_x)^2\right)^{\frac{1}{2}}$$

Čia $x$ yra paveikslėlis ir $(\mu)$ yra visų paveikslėlio pikselių vidurkis. Galiausiai, paskutinis matas, struktūros, gaunamas prieš tai suskaičiuotų reikšmių dėka:

$$(X - \mu_x)/\sigma_x$$

Apskaičiavus visus matas, toliau siekiama palyginti paveikslėlius su apskaičiuotomis reikšmėmis. Tokiu atveju, pradedant nuo šviesumo, palyginimas vyksta pagal formulę:

$$l(x,y) = \frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1}$$

Čia $x$ ir $y$ yra paveikslėliai, kurie yra palyginami, o $C_1$ yra konstanta, kuris skirta palaikyti reikšmes tokiu atveju jeigu atsakymas gautas būtų 0. $C_1$ reikšmė gaunama pagal formulę:

$$C_1 = (K_1 L)^2$$

Čia $L$ yra skaičius, kuris pikselio skaitinė išraiška (pavyzdžiui, tai gali būti 255 jeigu paveikslėlis – 8 bitų), ir $K_1$ (kartu ir $K_2$) yra konstantos, duotos formulės pradžioje – dažniausiai tai yra $K_1 = 0{,}01$ ir $K_2 = 0{,}03$. Toliau, kontrasto palyginimo formulėje sekama, kad:

$$c(x,y) = \frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2}$$

Šioje formulėje $C_2$ gaunamas panašiai kaip ir $C_1$, pagal formulę:

$$C_2 = (K_2 L)^2$$

Galiausiai, galutinė *SSIM* rezultato formulė gaunama pagal:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_x\sigma_y + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)}$$

Visas *SSIM* rezultatas yra tarp 0 ir 1, su 1 kaip geriausia reikšme (panašiausi paveikslėliai) ir 0 kaip pačia blogiausia. Kaip atrodo apskaičiuotas *SSIM* palyginant paveikslėlius, galima pamatyti 4 lentelėje:

4 lentelė – SSIM skaičiavimo pavyzdžiai

| | | |
|---|---|---|
|  SSIM = 1 (vienodi vaizdai) |  SSIM = 0,96729 |  SSIM = 0,97272 |

Kadangi pati metrika yra gana sudėtinga, tai paprastoje prieš tai naudotoje *skimage* bibliotekoje programinio įgyvendinimo nėra. Pasitelkus kitą biblioteką, *sewar*, galima atrasti ir šią metriką – pagrindiniai jos parametrai yra spalvintas ir originalus paveikslėlis, taip pat K1 ir K2 matai (pagal nutylėjimą, pateikti kaip 0,01 ir 0,03), dėl to pasinaudojant šia biblioteka, galima nesunkiai integruoti šią metriką į analizę.

## 6.5. Daugialypio dydžio struktūrinio panašumo indekso skaičiavimas – MS-SSIM

Daugialypio dydžio struktūrinio panašumo indekso skaičiavimas (angl. *Multi-scale struktural similarity index measure*), trumpinys *MS-SSIM*, patobulintas *SSIM* skaičiavimo būdas, pirmą kartą pristatytas 2003 ir pasiūlęs į esančią metriką įtraukti daugiau kintamųjų, tokių kaip skirtingi masteliai [WSB03]. Nauja pasiūlyta metrika, kuri naudoja skirtingų mastelių pasinaudojant filtrus ir paveikslėlių pavyzdžių sumažinimą. Skirtingai nuo paprasto *SSIM*, MS-

SSIM kontrastas ir struktūra apskaičiuojama įvairiuose dydžiuose, o šviesumas apskaičiuojamas tik pradiniam palyginime. Apibendrintai, nauja formulė pristatyta kaip:

$$MS - SSIM(x, y) = [l_M(x, y)]^{\alpha M} \prod_{j=1}^{M} [c_j(x, y)]^{\beta j} [s_j(x, y)]^{\gamma j}$$

Čia $x, y$ – lyginami paveikslėliai, o $\beta, \alpha, \gamma$ – skirtingų dydžių komponentės. Pagal kūrėjus, optimalūs komponenčių dydžiai yra $\beta_1 = \gamma_1 = 0{,}0448$, $\beta_2 = \gamma_2 = 0{,}2856$, $\beta_3 = \gamma_3 = 0{,}3001$, $\beta_4 = \gamma_4 = 0{,}2363$, $\alpha_5 = \beta_5 = \gamma_5 = 0{,}1333$ [WSB03]. Geriausias tokios metrikos rezultatas – 1, kada gaunamas lyginant vienodus vaizdus. Kaip atrodo apskaičiuota palygintų paveikslėlių *MS-SSIM* metrika galima pamatyti 5 lentelėje.

5 lentelė – MS-SSIM skaičiavimo pavyzdžiai

| | | |
|---|---|---|
|  MS-SSIM = 1 (vienodi vaizdai) |  MS-SSIM = 0,95501 |  SSIM = 0,91270 |

Kaip ir su paprasta *SSIM* metrika, į analizės bandymus metriką galima įdiegti jau suprogramuotais įrankiais. Tokį būdą pasiūlo jau prieš tai paminėta programavimo biblioteka *sewar*, kuri kartu pasiūlo ir skaičiavimą *MS-SSIM*. Jos parametrai, kaip ir visų prieš tai metrikų, apima abu paveikslėlius – originalų ir nuspalvintą, taip pat galima pasirinkti suvesti skirtingus kintamųjų β, γ, α dydžius (pagal nutylėjimą, siūlomi pradiniai dydžiai atitinka tai kas aprašoma [WSB03]), taigi pasinaudojant šią biblioteka galima nesunkiai įdiegti šį skaičiavimo būdą į bendrą sistemą.

Apibendrinant, vaizdų analizėje galima pritaikyti įvairias matematines metrikas, skirtas vaizdų kokybei nustatyti. Kiekviena iš tų metrikų turi skirtingą savo skaičiavimo būdą ir siekiamą rezultatą, dėl to panaudojant skirtingas metrikas galima tikrai detaliai nustatyti paveikslėlio kokybę. Taip pat, apibendrinant galima nustatyti, kuri iš tų metrikų tiksliausiai atspindi paveikslėlio kokybę.

# 7. Metodologija ir analizavimo strategija

Norint surasti geriausią rezultatą atitinkančius parametrus, reikalinga strategija, su kuria galima efektyviai ir organizuotai išbandyti kiek įmanoma daugiau variantų parametrų įvairovės. Visi bandymai turi būti atliekami vienodomis sąlygomis ir vienodais kriterijais kad gauti kuo tikslesnius rezultatus. Todėl iš pradžių, prieš atliekant analizę, reikalinga atlikti papildomos užduotys:

- Išrinkti paveikslėlių bandymų tematiką ir konkrečią sritį – nors teoriškai dirbtinis neuroninis tinklas gali išmokti neribotą kiekį objektų ir jų spalvų, reikalinga apsiriboti ties tam tikra sritimi, kad analizė turėtų tam aiškias ribas. Tas taip pat reikalinga ir dėl kompiuterinių skaičiavimo resursų ribotumo ir techninės įrangos galimybių. Todėl šiai analizei atlikti išrinkta tematika yra gamtos vaizdai – tai yra pakankamai siaura, bet ir pakankamai sudėtinga ir įvairi sritis, kurioje galima išbandyti dirbtinio neuroninio tinklo tikslumą. Į šią tematiką įeina miškų vaizdai, žolės, miškų skliautai, medžiai, miško takeliai ir panašūs vaizdai.
- Išrinkti apmokymo vaizdų aibę – kad būtų galima atlikti bandymus, dirbtinis neuroninis tinklas turi būti apmokytas. Reikalinga pakankamai didelė paveikslėlių aibė pagal išrinktą tematiką, tai yra gamtos vaizdų tematika. Šiai analizei atlikti pasitelkta apmokymo paveikslėlių aibė iš tinklapio *Kaggle*, paveikslėlių duomenų bazė „*Places365*“[12]. Iš šios aibės išrinktos kategorijos - *forest_path, forest_road, forest-broadleaf, field_wild, lawn.* Iš viso apmokymo paveikslėlių aibę sudaro 25 000 paveikslėlių.
- Išrinkti analizės paveikslėlių aibė – kad vienodomis sąlygomis būtų galima palyginti rezultatus, reikalinga paveikslėlių aibė, ant kurios bus apmokytas tinklas. Ji turi būti spalvotų paveikslėlių su tematika, parinkta bandymams, tai yra gamtos vaizdai. Tam, kad rezultatai būtų kuo tikslesnis, reikalingi originalūs, niekur nenaudoti vaizdai, nes panaudojant vaizdus, rastus internete, kyla rizika, kad jie gali būti pakliuvę tarp apmokymo aibės ir taip potencialiai neuroninis tinklas gali parodyti geresnius rezultatus. Dėl to buvo nufotografuota 10 originalių paveikslėlių, kurios toliau bus pasitelktos analizei. Originalias nuotraukas galima pamatyti skyrelyje „*Priedai*“. 10 paveiksėlių taip pat buvo pasirinkta ir dėl kelių kitų priežaščių - tai yra pakankamas kiekis norint išvesti rezultato vidurkį, taip pat atlikus spalvinimo bandymą, taip pat

[12] https://www.kaggle.com/benjaminkz/places365?select=train

peržiūrėti, ar nuspalvinimo rezultatai kokybiškai atitinka tuos, kuriuos parodo apskaičiuotos metrikos.

Visos prieš tai reikalingos užduotys buvo reikalingos dar iki analizės pradžios ir skirtos padaryti toliau daromas analizes kuo tikslesnes. Atlikus šias užduotis, toliau galima organizuoti analizės strategijas.

Analizės strategijos apima pakeitimus dirbtinio neuroninio tinklo parametrams ir pačiam neuroniniam tinklui. Galima daryti prielaidą, kad skirtingos parametrų ir dirbtinio neuroninio tinklo architektūros pakeitimai padės gauti geresnius rezultatus negu prieš tai kurto autoriaus naudoti. Siekiant gauti tokį rezultatą, šiame mokslo tiriamajame darbe numatoma atlikti tokius bandymus ir atsakyti į šiuos klausimus:

- Mokymo iteracijų skaičiaus keitimas – ar didesnis iteracijų skaičius tikrai padeda gauti geresnį rezultatą, ar galima nutraukti neuroninio tinklo mokymą anksčiau ir gauti panašų arba geresnį rezultatą, tokiu atveju sutaupyti ir laiko apmokymams?
- Mokymo paveikslėlių grupės (angl. „*batch*“) dydžio keitimas – ar su didesnėmis grupėmis galima gauti tikslesnį ir įvairiapusiškesnį rezultatą? Ar šis pakeitimas turi kokį nors poveikį tinklo apmokymo tikslumui?
- Naujo klasifikavimo tinklo pakeitimas – kadangi dabartinė neuroninio tinklo architektūra naudoja *VGG16* klasifikatorių, tai ar pakeitus jį naujesniu *VGG19* arba *ResNet50* galima gauti geresnius rezultatus?
- Geriausių rezultatų palyginimas su kitais neuroniniais tinklais – ar gauti rezultatai vidutiniškai atrodo taip pat gerai palyginus su kitais neuroniniais tinklais ir jų spalvintais paveikslėliai?

Visos neuroninio tinklo analizės bus atliktos vienodomis sąlygomis. Visi spalvinimai bus atliekami tiems patiems 10 paveikslėlių - originalias nuotraukas galima pamatyti skyrelyje “Priedai”. Šios nuotraukos bus paverstos į nespalvotas, tuomet, pasinaudojant neuroniniu tinklu, bus nuspalvintos atgal į spalvotas. Tuomet bus atliktas palyginimas su originaliomis spalvomis ir pagal turimas metrikas tarp dviejų vaizdų bus išvestas rezultatas. Tokių rezultatų bus 10. Galiausiai, iš 10 gautų rezultatų bus išvestas bendras metrikos vidurkis, kuris ir bus laikomas galutiniu metrikos ir analizės rezultatu.

Apibendrinant, pagal sukurtą strategiją ir turimas atlikti užduotis, galima toliau pasiruošti atlikti dirbtinio neuroninio tinklo analizę ir gauti rezultatus, iš kurių toliau galima daryti apibendrintas išvadas.

# 8. Atliktų analizės bandymų rezultatai

## 8.1. Pasiruošimas atlikti analizes

Prieš atliekant analizę, reikalinga tam tikra tinklo modifikacija. Šios analizės apima daug skaičiavimo, dėl to reikalinga modifikacija šiems rezultatams gauti. Į dirbtinį neuroninį tinklą reikalingos įdiegti matematinio skaičiavimo metrikos metodai, aprašyti praeitame skyriuje ir papildomas rezultatų išsaugojimas. Gauti matematinės išraiškos rezultatus padės *Python* programavimo biblioteka *sewar* ir *skimage*. Jose jau yra sukurtos skaičiavimo funkcijos pagal prieš tai aprašytas metrikas. Toliau padaryti pakeitimai failui ***test.py*** , kurie atvaizduoti 21 pav. ir 22 pav.

```python
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt

from config import batch_size, display_step, saving_step, summary_path, testing_summary
from common import init_model
from image_helper import concat_images
from sewar import full_ref
from skimage import measure
import cv2
```

**21 pav. Pridėtos bibliotekos prie failo.**

```python
while not sess.should_stop():
    step += 1

    l, pred, color, gray = sess.run([loss, predict_rgb, color_image_rgb, gray_image], feed_dict={is_training: False})

    print(" Testing iter %d, Minibatch Loss = %f" % (step, l))
    avg_loss += l

    for i in range(len(color)):
        summary_image = concat_images(gray[i], pred[i])
        summary_image = concat_images(summary_image, color[i])
        plt.imsave("%s/images/orig/%d_%d.png" % (testing_summary, step, i), color[i])
        plt.imsave("%s/images/gray/%d_%d.png" % (testing_summary, step, i), gray[i])
        plt.imsave("%s/images/colo/%d_%d.png" % (testing_summary, step, i), pred[i])

        plt.imsave("%s/images/%d_%d.png" % (testing_summary, step, i), summary_image)

        file = open("summary/test/images/testai/%d_%d.txt" % (step, i), "w+")
        ref_img = cv2.imread("summary/test/images/orig/%d_%d.png" % (step, i), 1)
        img = cv2.imread("summary/test/images/colo/%d_%d.png" % (step, i), 1)

        mse_skimg = measure.compare_mse(ref_img, img)
        file.write("MSE: based on scikit-image = %s" % mse_skimg)
        file.write("\n")

        psnr_skimg = measure.compare_psnr(ref_img, img, data_range=None)
        file.write("PSNR: based on scikit-image = %s" % psnr_skimg)
        file.write("\n")

        rmse_skimg = measure.compare_nrmse(ref_img, img)
        file.write("RMSE: based on scikit-image = %s" % rmse_skimg)
        file.write("\n")

        ssim_img = full_ref.ssim(ref_img, img, ws=11, K1=0.01, K2=0.03, MAX=None, fltr_specs=None, mode='valid')
        file.write("SSIM: structural similarity index = " + str(ssim_img[0]))
        file.write("\n")

        msssim_img=full_ref.msssim(ref_img, img, weights=[0.0448, 0.2856, 0.3001, 0.2363, 0.1333], ws=11, K1=0.01, K2=0.03, MAX=None)
        file.write("MSSSIM: multi-scale structural similarity index = %s" % msssim_img)
        file.write("\n")

        file.close()
    if step >= len(file_paths) / batch_size:
        break

print("Testing finished!")
print("Total average loss: %f" % (avg_loss / len(file_paths)))
```

**22 pav. Pridėtas papildomas kodas rezultatams išsaugoti.**

Svarbu žinoti, kad neuroniniame tinkle taip pat naudojama suprogramuota praradimų funkciją (angl. *loss function*). Ji veikia paprastai - praradimo dydžio skaičiavimas atliekamas panaudojant originalų ir nuspalvintą paveikslėlius, tuomet skaičiuojamas atstumo skaičiavimas atitinkamoms pikselių reikšmėms. Tuomet tas pats vykdomas abiems paveikslėliams prieš tai juos išblukinant (angl. *blur*) panaudojant Gauso matricų filtrus. Filtrų dydžiai yra 3x3 ir 5x5. Tuomet apskaičiuojamas atitinkamas atstumas tarp išblukintų paveikslėlių pikselių reikšmių. Pabaigai, turint tris dydžius, iš jų išvedamas vidurkis.

Prie failo pradžios pridėtos *skimage* ir *sewar* toliau leidžia pasinaudoti matematinėmis funkcijomis. Toliau yra sukurtas tekstinio failo objektas ir į jį įrašomi funkcijų rezultatai. Funkcijoms visada perduodami 2 paveikslėliai – originalus ir dirbtinio neuroninio tinklo sukurtas. Iš jų yra apskaičiuojami numatytų matematinių funkcijų rezultatai. Be to, papildomai sukurtas naujas paveikslėlių išsaugojimas, kur paveikslėliai yra išsaugomi į atskirus aplankalus, jeigu jų prireiktų kaip atskirų objektų. Susikūrus šį vertinimą, galima toliau atlikti dirbtinio neuroninio tinklo analizę.

## 8.2. Originalaus tinklo rezultatai

Kaip prieš tai buvo minėta, originalus tinklus yra nepakeista, nemodifikuota analizuojamo neuroninio tinklo versija, kartu su prieš tai išmokytu neuroninio tinklo modeliu. Ši versija bus naudojama kaip pagrindas toliau atliekant visas analizes.

Toliau, kad būtų su kuo palyginti, reikia atlikti bandomus spalvinimus su originaliu, nekeistu dirbtiniu neuroniniu tinklu. Kadangi dirbtinio neuroninio tinklo kodo autorius pateikė atsisiuntimui jau paruoštą ir išmokytą modelį[13], kuris buvo apmokytas atliekant 3 000 000 iteracijų, tai spalvinimo procesą galima pradėti iš karto be jokio papildomo apmokymo.

Analizės paveikslėlių aibė buvo autoriaus išrinktos nuotraukos, kurias galima pamatyti skiltyje „*Priedai*“. Gautos analizės iš visų 10 paveikslėlių, apskaičiuotas bendras jų vidurkis ir gauti rezultatai yra surašyti 6 lentelėje.

6 lentelė – Pradiniai originalaus dirbtinio neuroninio tinklo rezultatai

| *Formulė* | *Vidurkis* |
|---|---|
| MSE | 207,3758 |
| PSNR | 25,80668 |
| RMSE | 0,133882 |
| SSIM | 0,948347 |
| MSSSIM | 0,952993 |

[13] https://github.com/Armour/Automatic-Image-Colorization/releases/tag/2.0

Tokio dirbtinio neuroninio tinklo pradinius rezultatus galima pamatyti paveikslėliuose 23 pav. Dirbtinis neuroninis tinklas sėkmingai atliko savo darbą ir pateikė savo rezultatą po bandymo.

**23 pav. Kairėje – nespalvotas paveikslėlis, viduryje – dirbtinio neuroninio tinklo atkurtos spalvos, dešinėje – originalas.**

Gavus rezultatus, toliau galima atlikti pagrindinius mokymo testus, nes turint pradinius ir nekeisto tinklo rezultatus, galima tolimesnius rezultatus su juo palyginti.

## 8.3. Iteracijų dydžio analizė

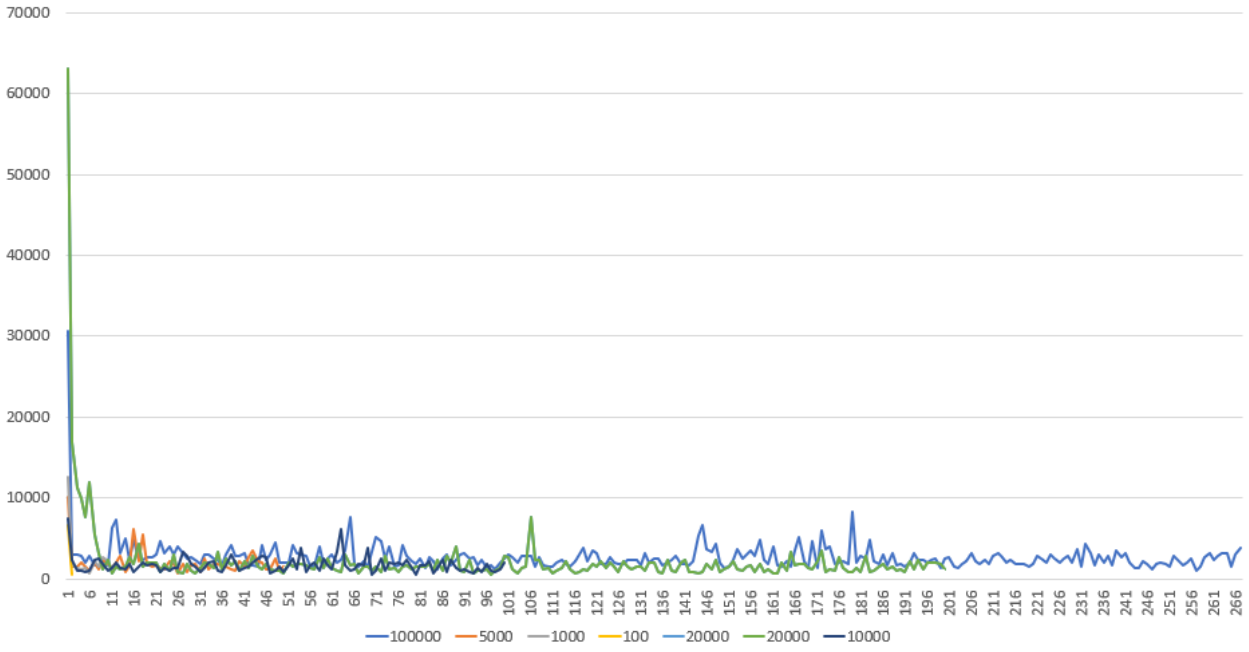
Pirmoji analizė yra su mokymo iteracijų skaičiumi. Pirmas spėjimas gali būti, kad su didesniu iteracijų skaičiumi galima gauti geresnius rezultatus, arba kad po kurio laiko tinklas persimokys ir pradės daryti klaidas. Kad tai patikrinti ir išmatuoti kaip iteracijų skaičius daro įtaką spalvinimo kokybei, toliau atliktos analizės tokiais parametrais ir gauti rezultatai atvaizduoti 24 pav. ir 25–29 pav.

- Grupės dydis (angl. *batch size*) – 6.
- Validacijos dažnis (angl. *validation*) – 2000 iteracijų - kas kiek laiko atliekamas spalvinimo bandymas ir skaičiuojama paklaida.
- Maišymo dažnis (angl. *shuffle*) – 2000 iteracijų - kas kiek laiko paimama nauja grupė atsitiktinių paveikslėlių ir jų naudojimas mokymo metu.
- Iteracijų skaičius (angl. *iterations*)– priklauso nuo bandymo.

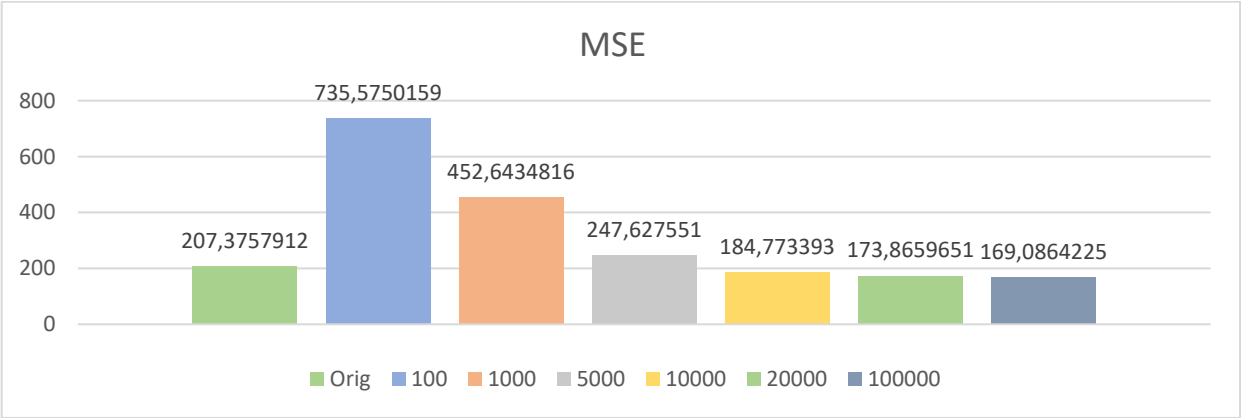
- Neuroninio tinklo architektūra – nekeista.

7 lentelė – Iteracijų analizės tinklo apmokymo laikai

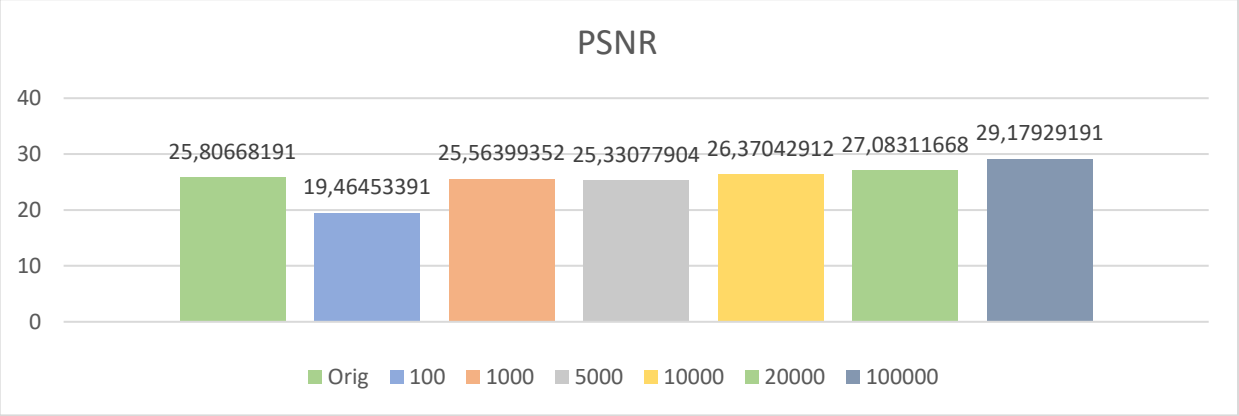
| | |
|---|---|
| 100 iteracijų | 3 min. 45s |
| 1000 iteracijų | 12 min. 2s |
| 5000 iteracijų | 31 min. 36s |
| 10000 iteracijų | 1 val. 7 min. 14s |
| 20000 iteracijų | 1 val. 48 min. 5s |
| 100000 iteracijų | 6 val. 14 min. 26s |



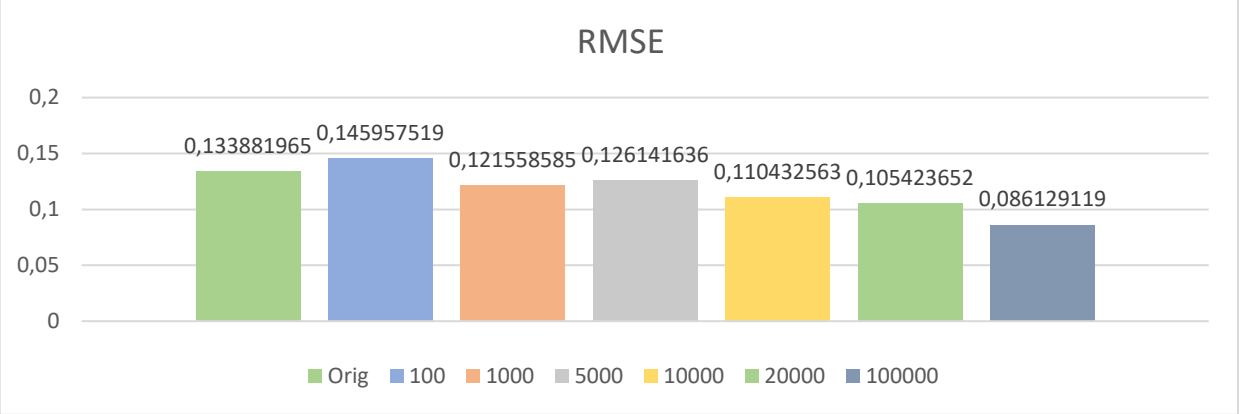

**24 pav. Iteracijų skaičiaus įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



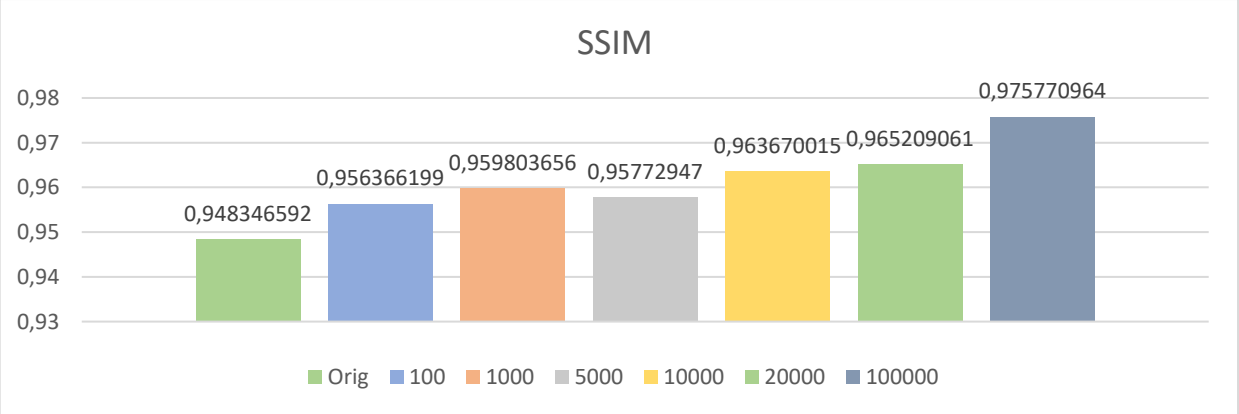

**25 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



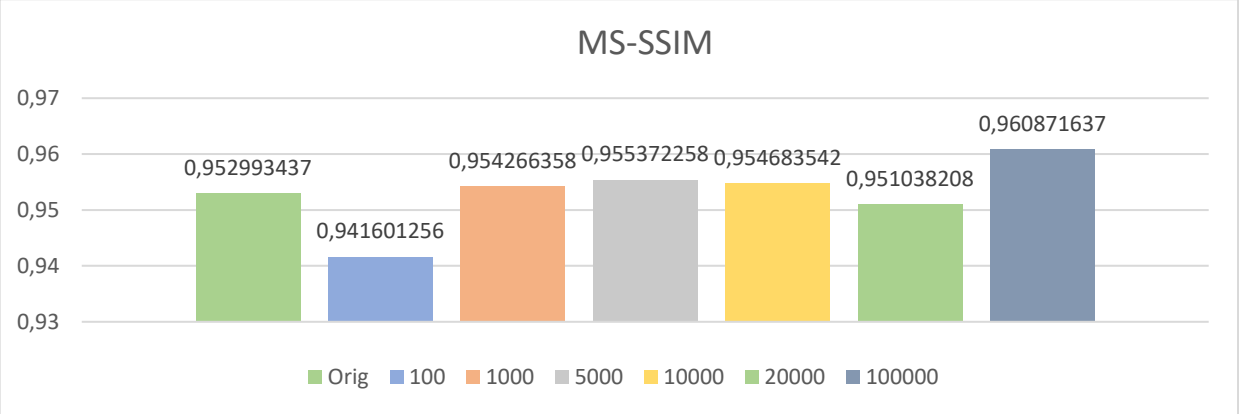

**26 pav. PSNR metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**27 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**28 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**29 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

Iteracijų analizės rezultatas – pavyko gauti geresnius rezultatus nuo ~20000 iteracijų, kada grafikai jau pradėjo rodyti geresnius rezultatus negu originalus tinklas. Nors skyrėsi tik iteracijų skaičius, tai yra kiek ilgai vykdyti skaičiavimus, grafikai skiriasi vienas nuo kito, nors atrodo, kad jie turi sutapti. Paklaidos mokymo metu visada atsiranda, nes ne visada atsitiktinai parenkami tie patys paveikslėliai, taip pat mokymo metu naudojama daug kitų atsitiktinių duomenų - dėl to grafikai negali būti identiški. Geriausiai tai parodė *SSIM, PNSR* ir ypač *MSE* grafikai, kuris gerai sutapo su apmokymo iteracijų grafiku – iš pat pradžių spalvinimas gerokai klaidingas nuo originalaus vaizdo, tačiau vėliau nuosekliai žemėjo ir pasiekė geresnius rezultatus. Tačiau verta atkreipti dėmesį, kad skirtumas tarp 10 000, 20 000 ir 100 000 iteracijų nėra toks žymus, koks buvo tarp mažesnių iteracijų, tai iš to galima daryti išvadą, kad galbūt verta nutraukti mokymą tuojau po 20 000 iteracijų ir taip sutaupyti laiko (pagal mokymo laiką – apie ~4 valandas). Kaip stipriai skiriasi paveikslėlių spalvinimo kokybė tarp skirtingų mokymo laikų, galima pamatyti ir vizualiai. Skirtumą tarp geriausio ir blogiausio rezultatą galima pamatyti 30 pav. ir 31 pav.

**30 pav. Blogiausias rezultatas – MSE = 978.6742, 100 iteracijų bandymas.**

**31 pav. Geriausias rezultatas – MSE = 82.0900, 100 000 iteracijų bandymas.**

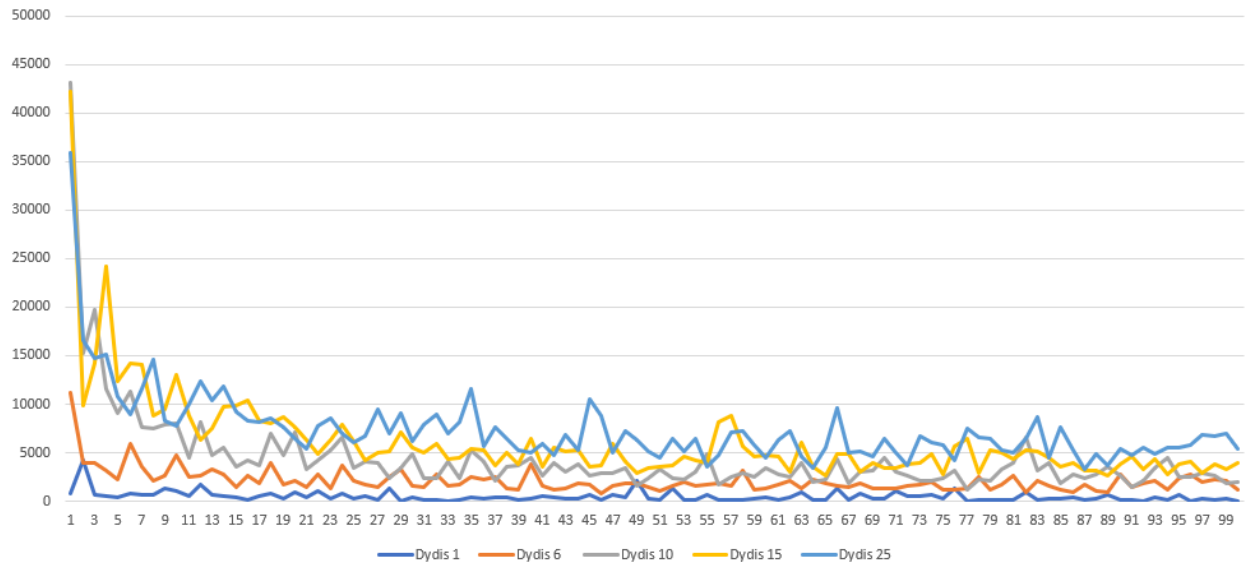
## 8.4. Grupės dydžio analizė

Antroji dalis analizių susideda iš grupės dydžio bandymų. Grupės dydis – tai paveikslėlių kiekis kurį vienu iteracijos metu išmoksta neuroninis tinklas iki tą grupę pakeičiant. Kitaip tariant, kuo didesnė grupė, tuo didesnis mokymosi iteracijos metu naudojamų paveikslėlių kiekis. Kadangi priklausomai nuo grupės dydžio, neuroniniam tinklui gali būti sunkiau išmokti, reikalinga atlikti bandymus ir su iteracijomis, galbūt ilgesnis mokymas didesniu iteracijų kiekiu padės taip pat gauti geresnius rezultatus. Toliau atliktos analizės tokiais parametrais ir gauti rezultatai atvaizduoti 32–37 pav.

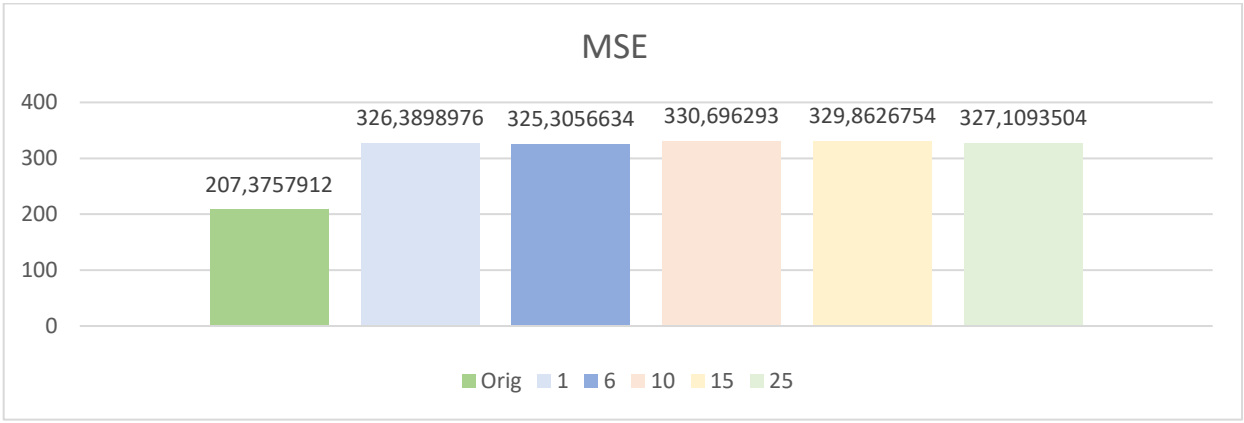
- Grupės dydis – (1, 6, 10, 15, 25) – 25 pasirinktas kaip aukščiausia reikšmė dėl techninių galimybių limito.
- Validacijos dažnis – 2000 iteracijų.
- Maišymo dažnis – 2000 iteracijų.
- Iteracijų skaičius – (100, 1000, 5000, 10000, 20000).
- Neuroninio tinklo architektūra – nekeista.

8 lentelė –Grupės dydžio analizės tinklo apmokymo laikai. Vertikaliai– iteracijų skaičius, horizontaliai – grupės dydis

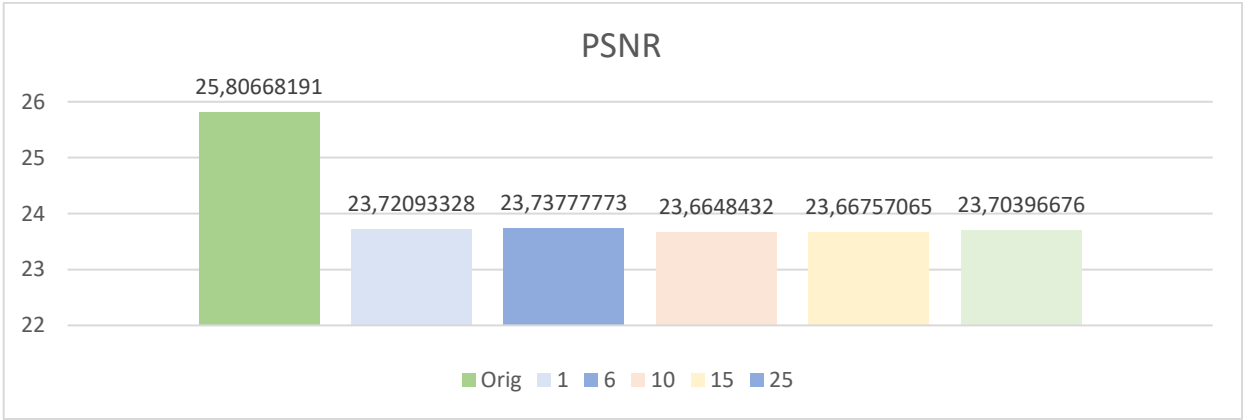
| | *1* | *6* | *10* | *15* | *25* |
|---|---|---|---|---|---|
| *100* | 2 min. 7s | 3 min. 45s | 4 min. 1s | 4 min. 22s | 5 min. 17s |
| *1000* | 5 min. 56s | 12 min. 2s | 14 min. 3s | 15 min. 44s | 17 min. 32s |
| *5000* | 10 min. 11s | 31 min. 36s | 48 min. 29s | 1 val. 3 min. 5s | 1 val. 32 min. 21s |
| *10000* | 21 min. 36s | 1 val. 7 min. 14s | 1 val. 29 min. 56s | 2 val. 14 min. 7s | 3 val. 52 min. 44s |
| *20000* | 37 min. 29s | 1 val. 48 min. 5s | 2 val. 15 min. 35s | 4 val. 23 min. 36s | 7 val. 39 min. 56s |



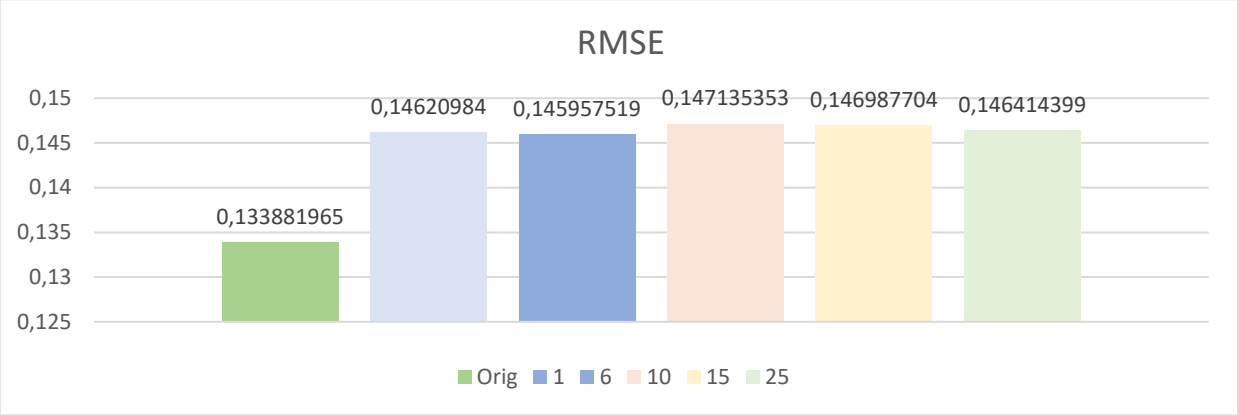

**32 pav. Grupės dydžio 100 įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 1 iteracija).**



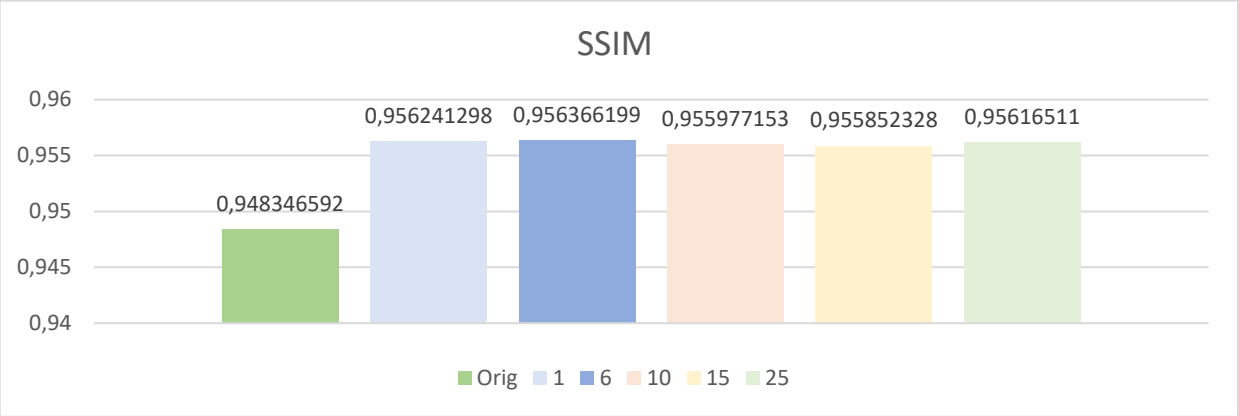

**33 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



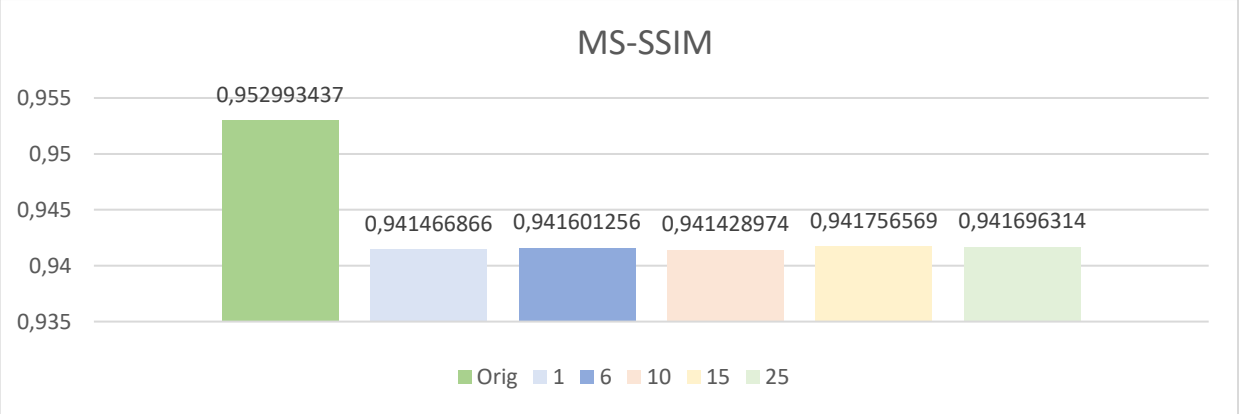

**34 pav. PSNR metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**35 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

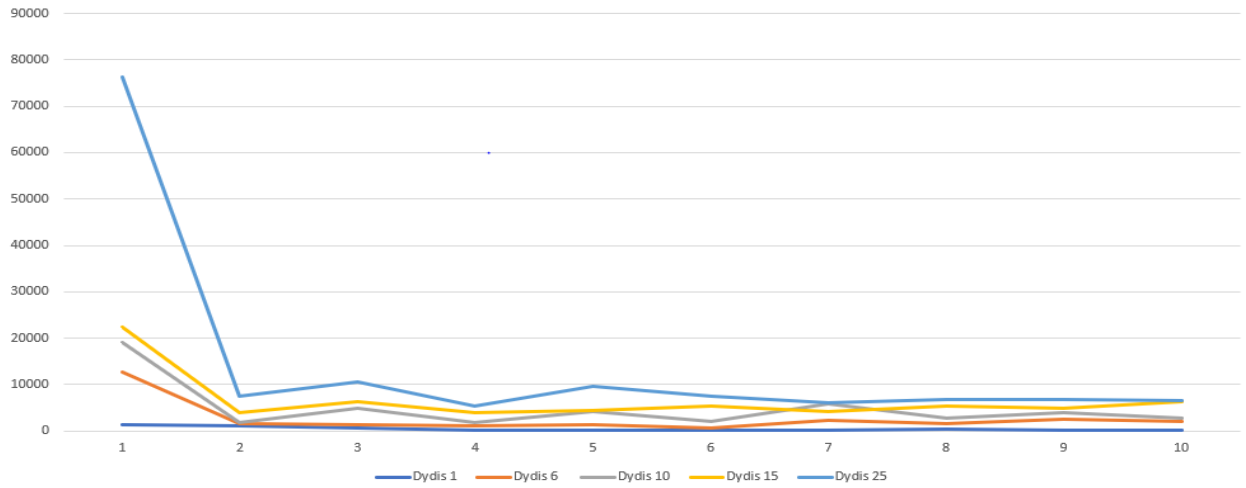



**36 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

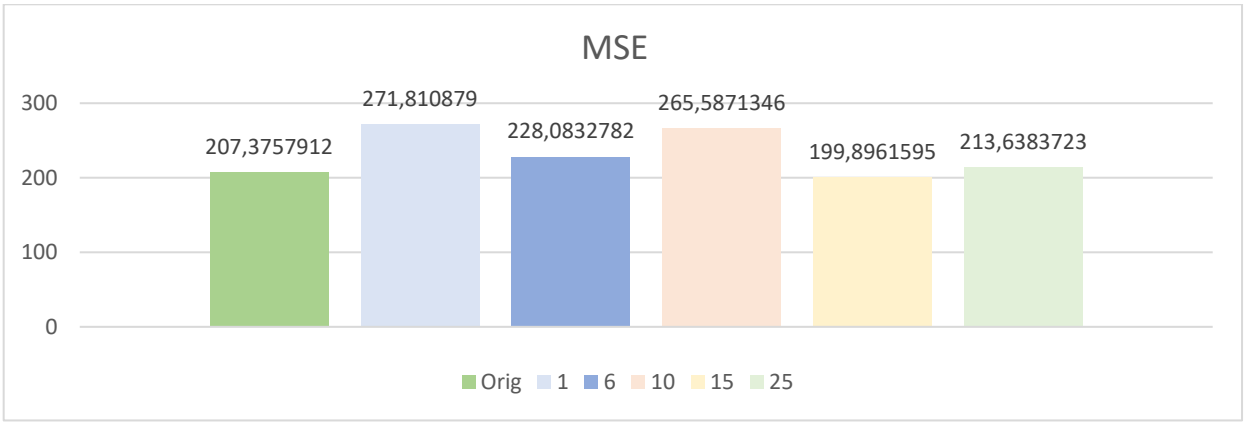



**37 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

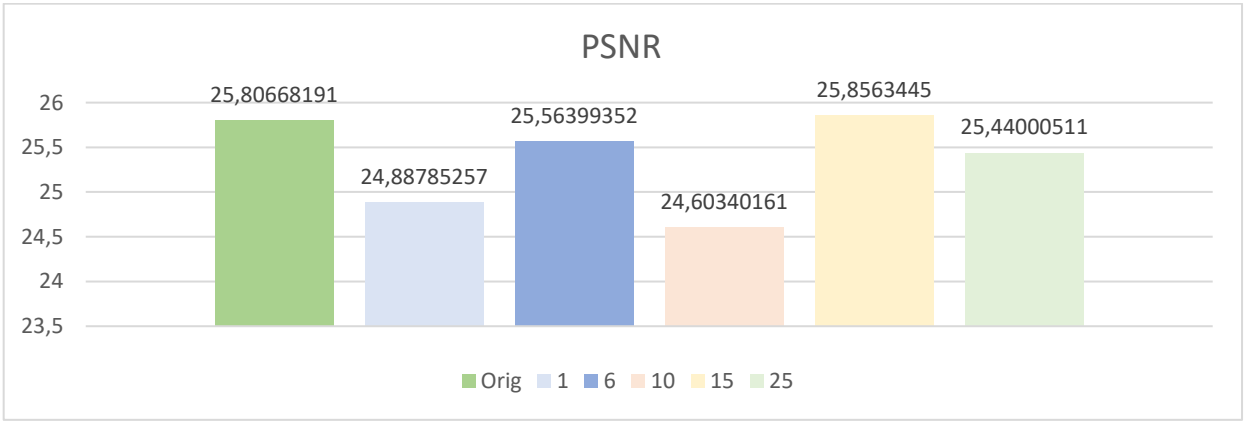
Pirmasis grupės bandymas buvo nesėkmingas ir beveik visomis metrikomis nusileidžia originaliam tinklui ir jo modeliui. Nors mokymo laikas buvo itin trumpas, tinklui nepavyko pademonstruoti greito rezultato. Sekantys bandymai apims grupės dydžio didinimą ir galbūt iš to ateinantį didesnį tikslumą. Analizės rezultatai atvaizduoti 38–43 pav.



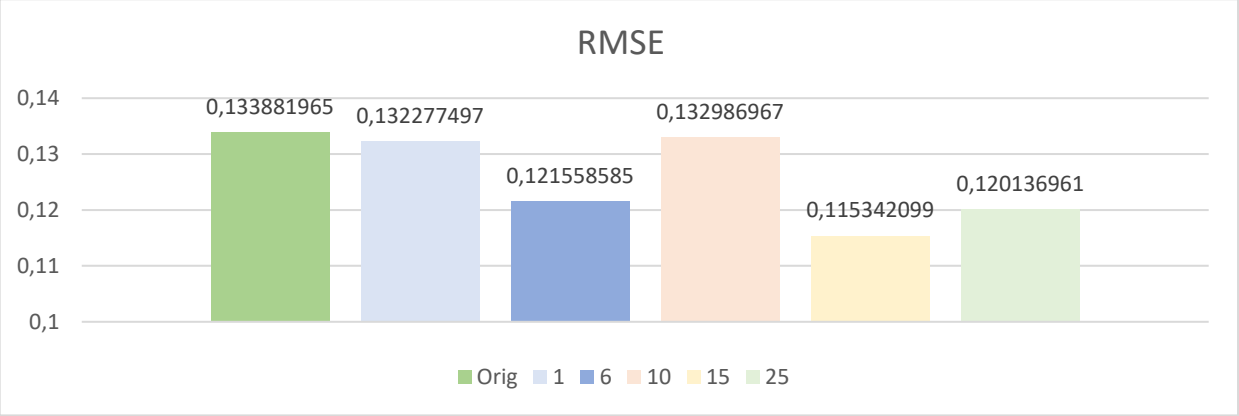

**38 pav. Grupės dydžio 1000 įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



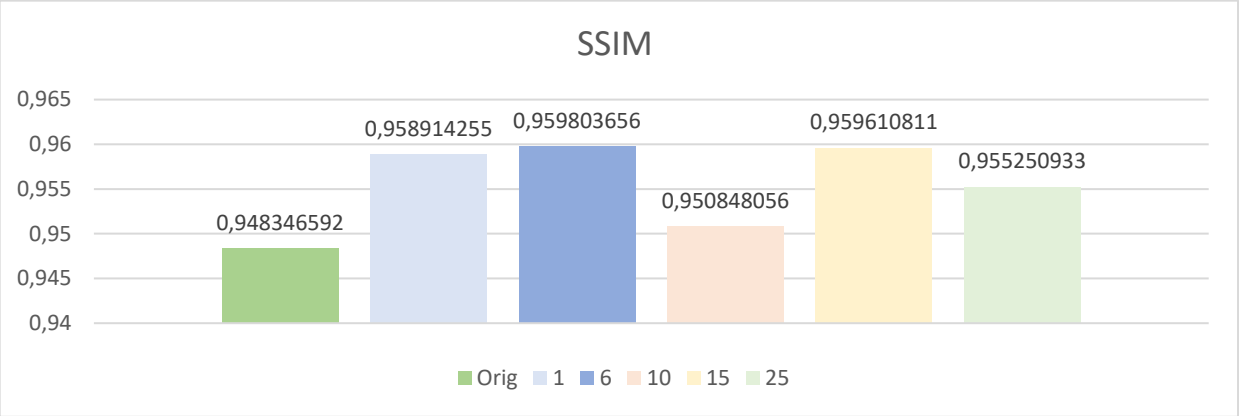

**39 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



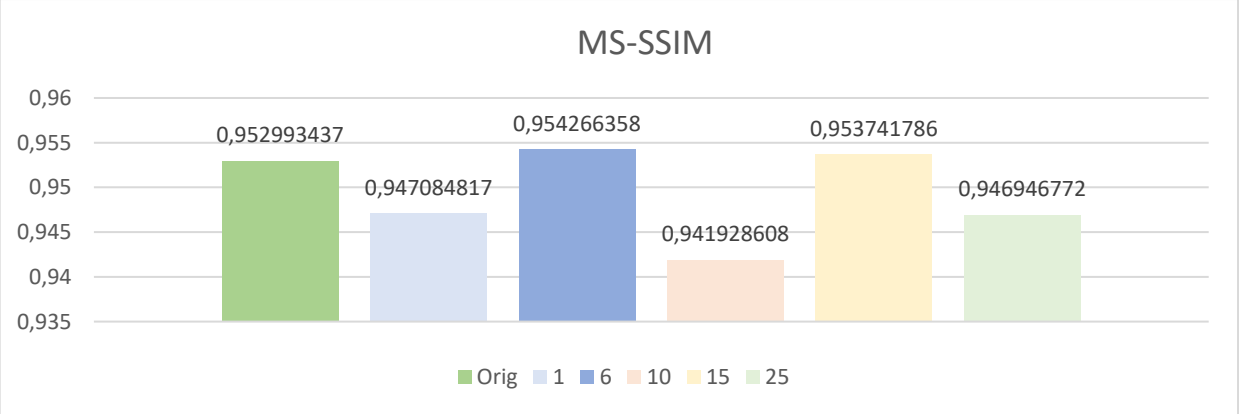

**40 pav. PSNR metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**41 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

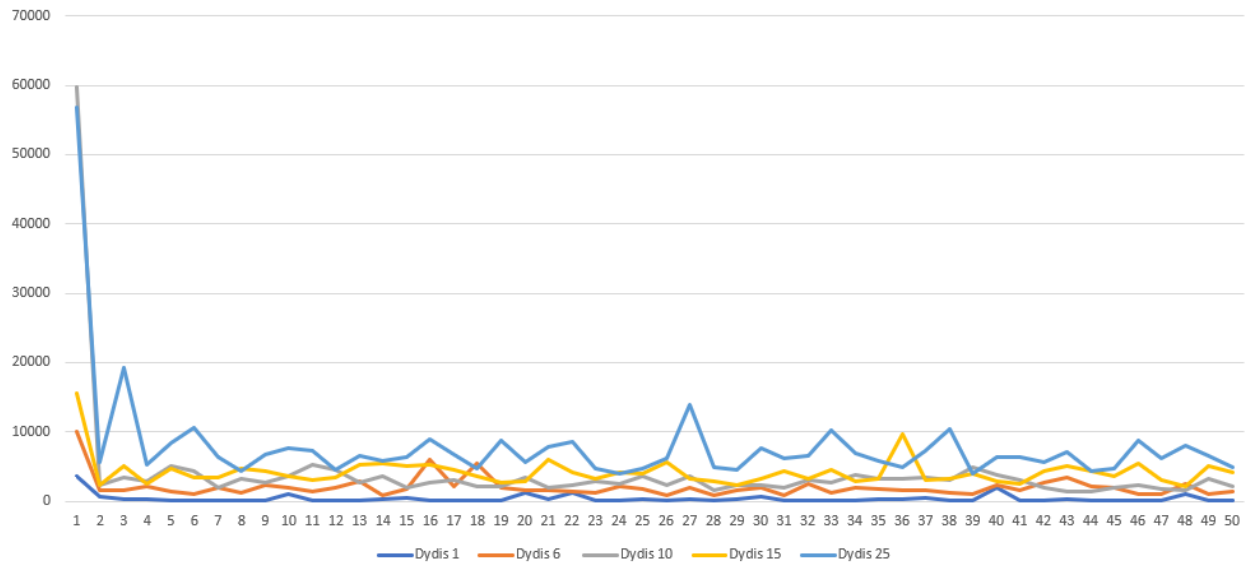



**42 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

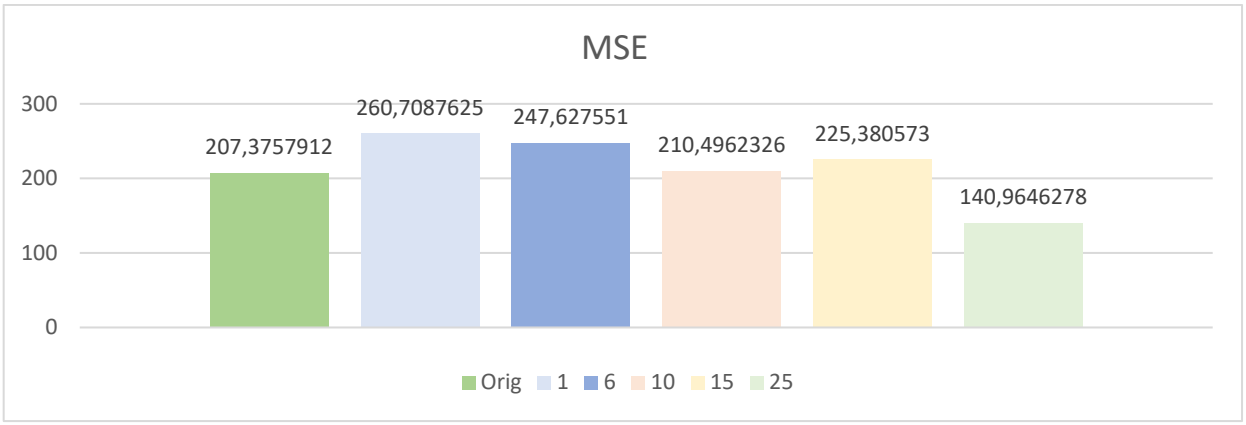



**43 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

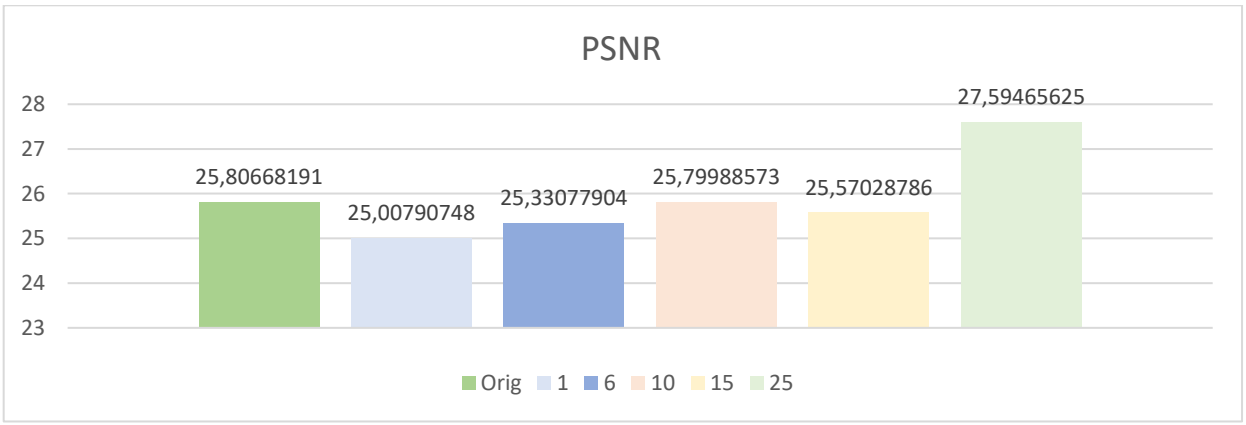
1000 iteracijų grupės analizės rezultatai jau yra panašūs į originalaus, nors didžioji dauguma metrikų vis dar rodo kad originalus tinklas vis dar geresnis spalvinti paveikslėlius. Taip pat verta atkreipti dėmesį, kad pradeda skirti įvairių metrikų rezultatai ir kai kuriais atvejais jos gali neatspindėti kokybės tiksliai. Rezultatai, gauti dar padidinus iteracijų skaičių atvaizduoti 44–49 pav.



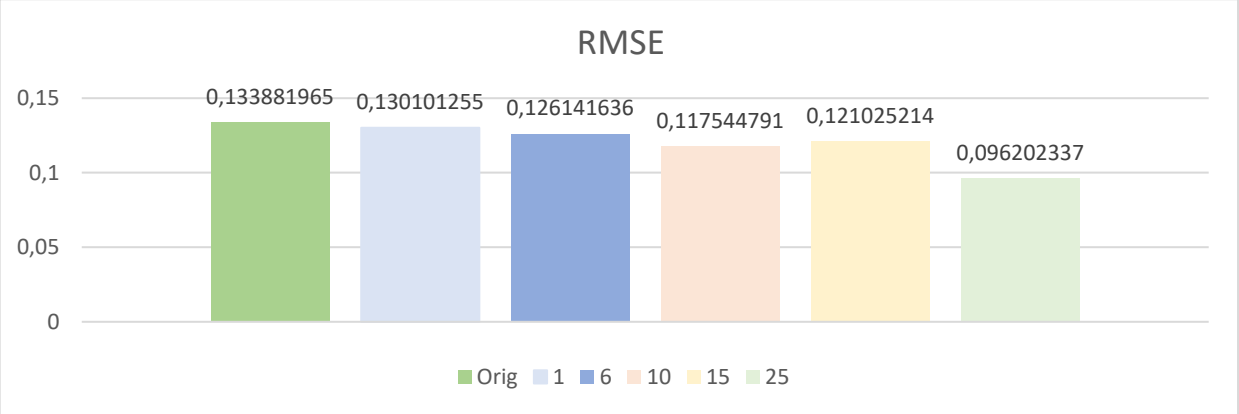

**44 pav. Grupės dydžio 5000 įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



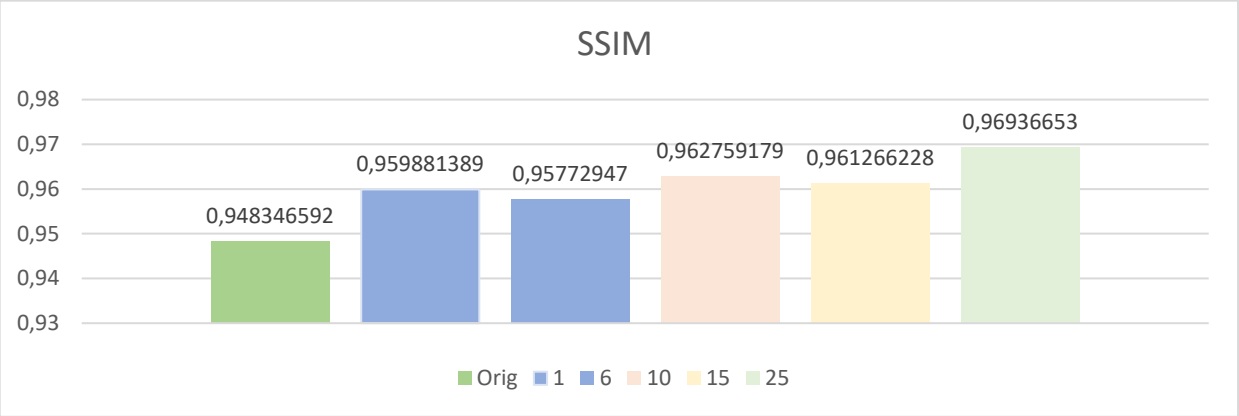

**45 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



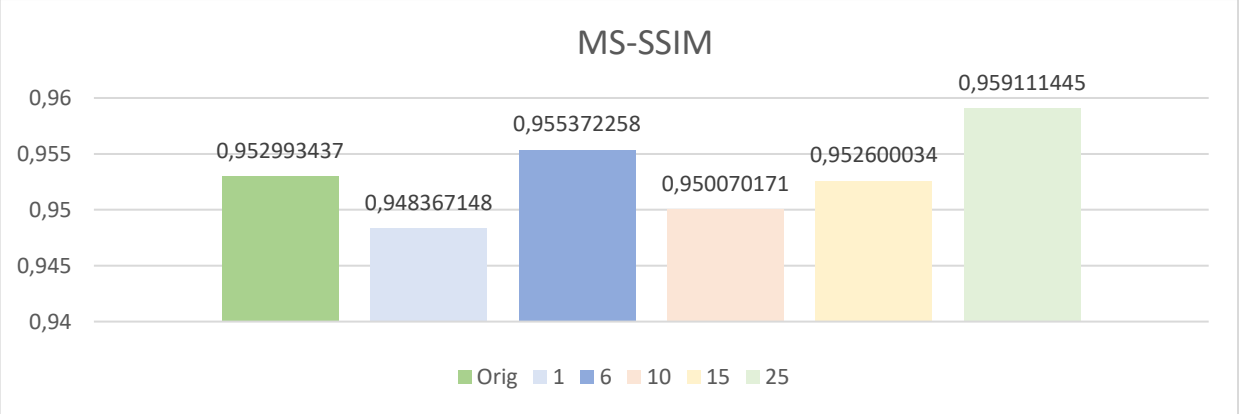

**46 pav. PSNR smetrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**47 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

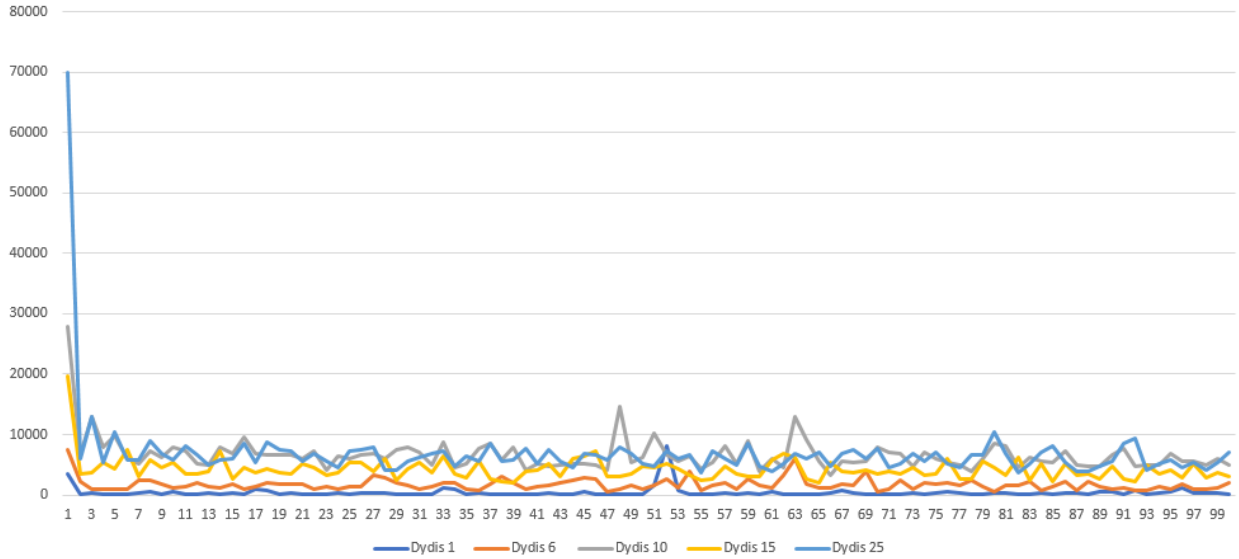



**48 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

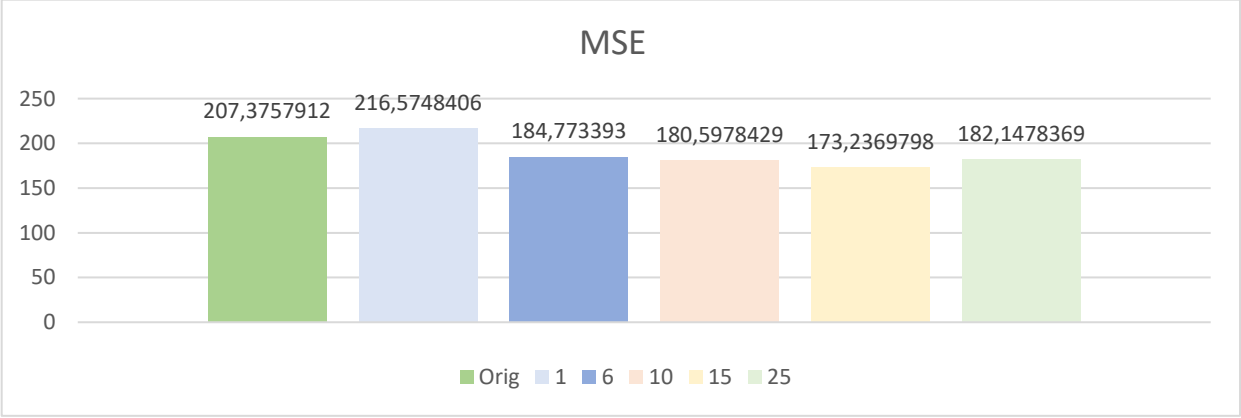



**49 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

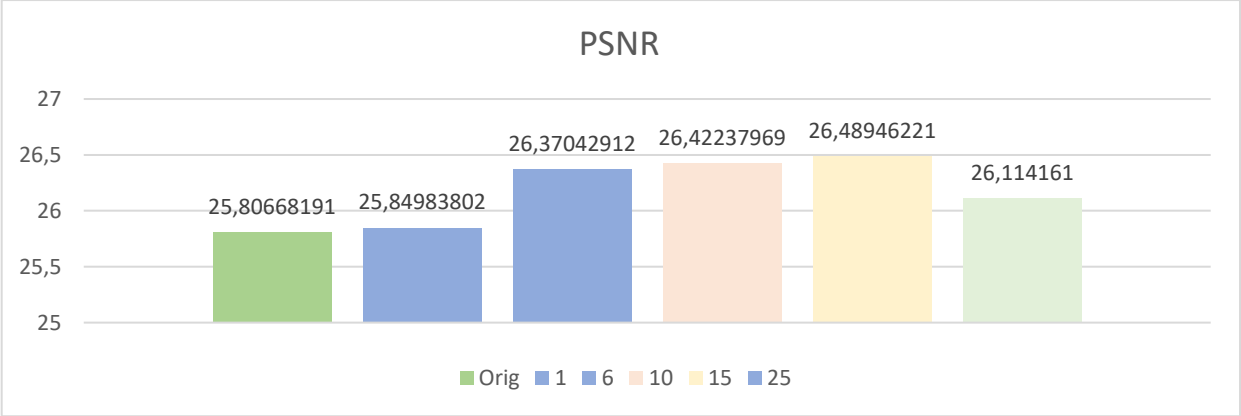
Pagal 5000 iteracijų analizę 10 paveikslėlių grupei jau matyti tendencija kad padidinus paveikslėlių kiekį kartu su iteracijų skaičiumi bendra kokybė jau pradeda kilti. Nors praradimų funkcijos grafikas kol rodo didesnę mokymosi paklaidą, tolygiai su didesne grupe, galutinis spalvinimo rezultatas tampa kokybiškesnis. Sekanti analizė padidinus iteracijų skaičių yra atvaizduota 51–55 pav.



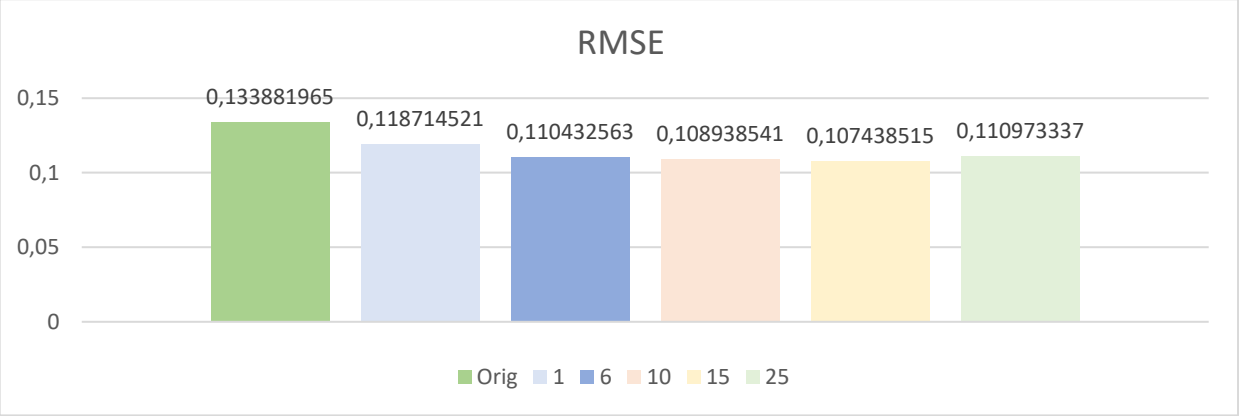

**50 pav. Grupės dydžio 10000 įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



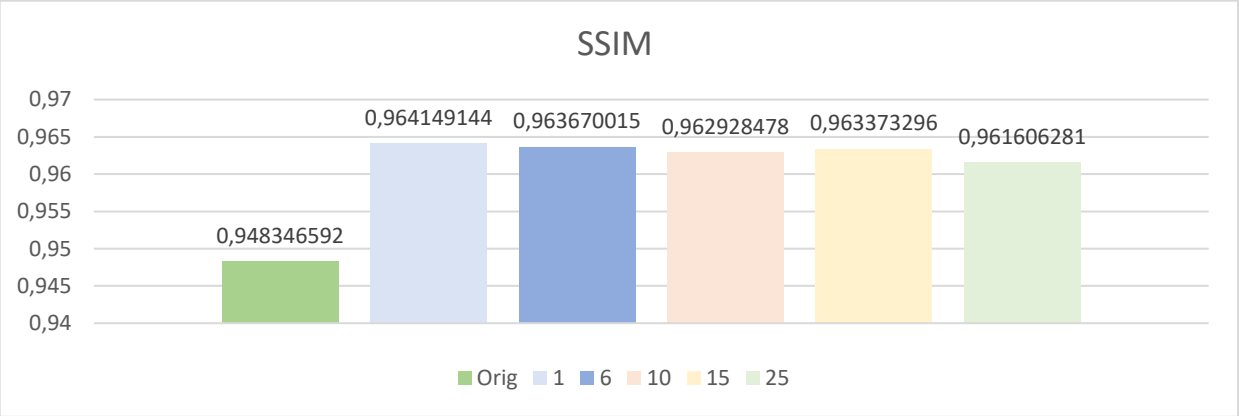

**51 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



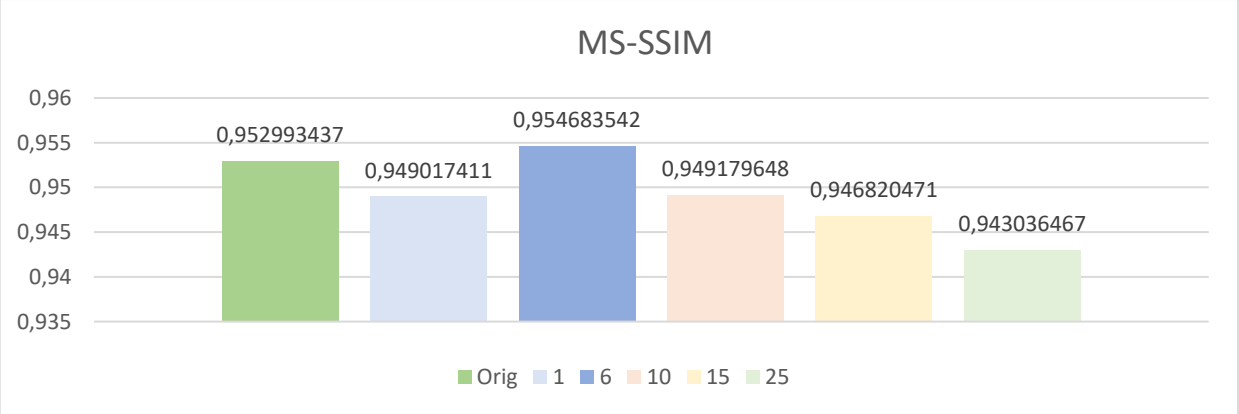

**52 pav. PSNR metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**53 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

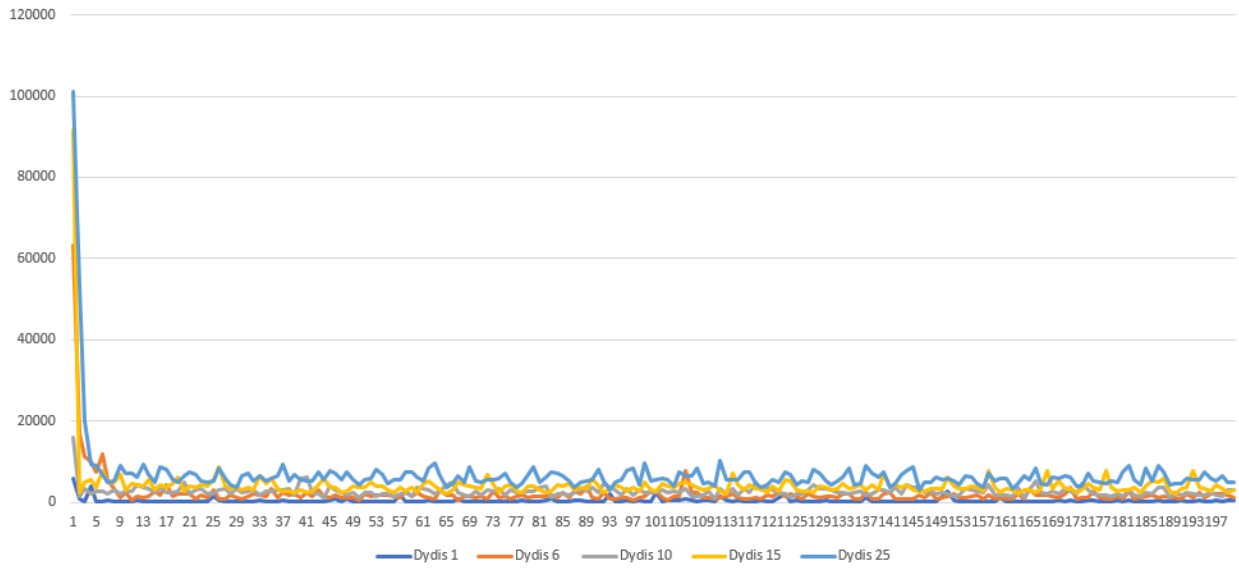



**54 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

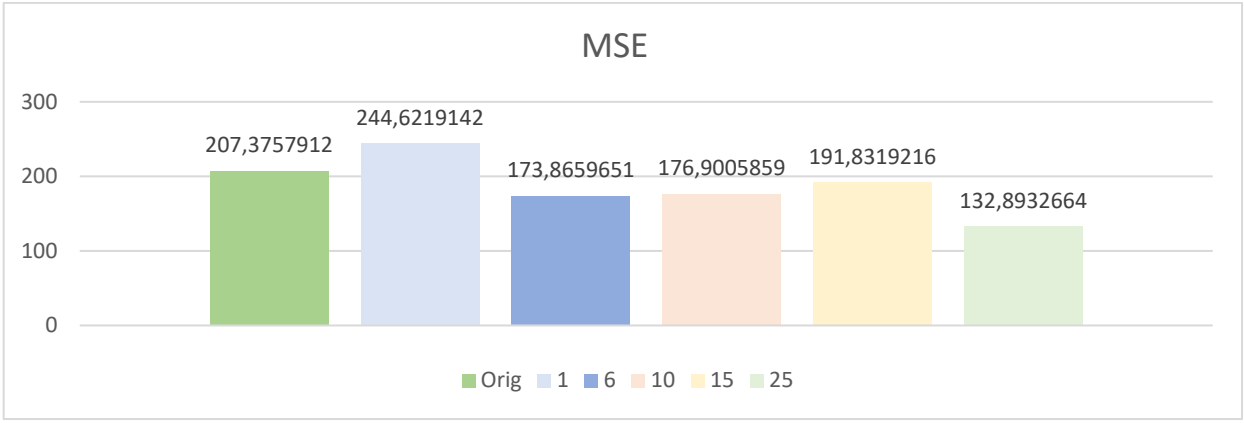



**55 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

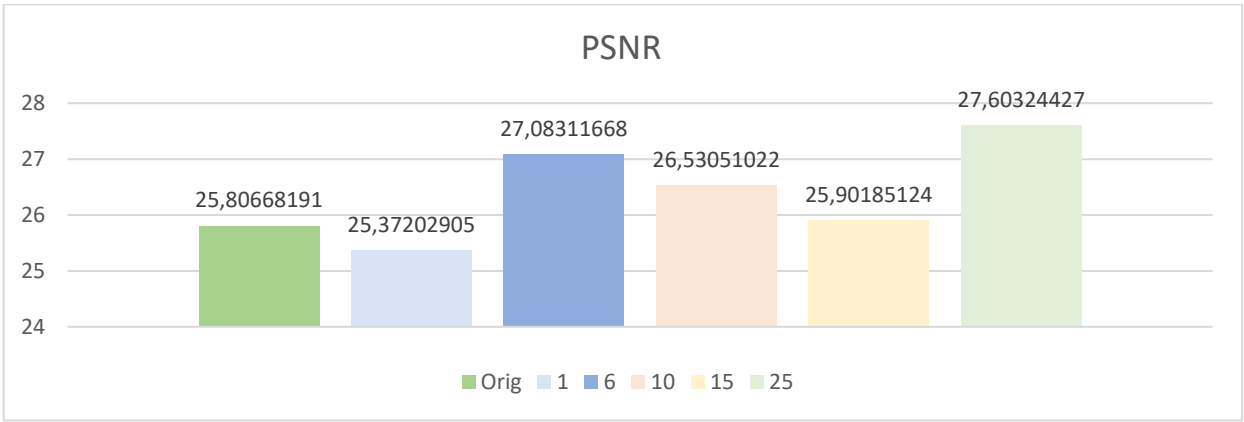
Pagal 10000 iteracijų ir grupių dydžių analizę galima pamatyti panašią tendenciją, kad padidėjus grupėms ir iteracijoms padidėja kokybė pagal keletą metrikų. Tačiau atsiranda kita tendencija kad nuo tam tikros ribos paklaida spalvinime didėja nepaisant to, kad grupės kiekis padidėjo. Ar tokia tendencija išlieka ir toliau, parodys paskutinė analizė, kurios rezultatai atvaizduoti 56–61 pav.



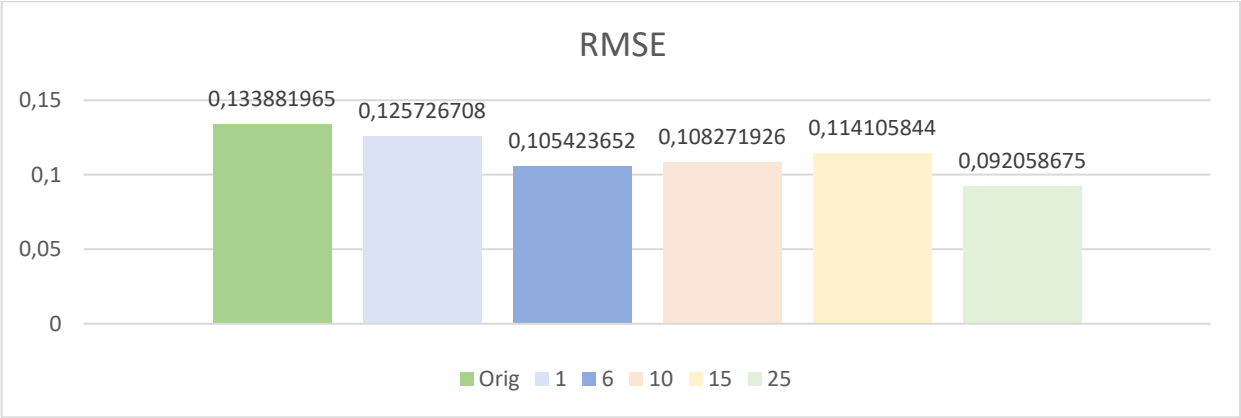

**56 pav. Grupės dydžio 20000 įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



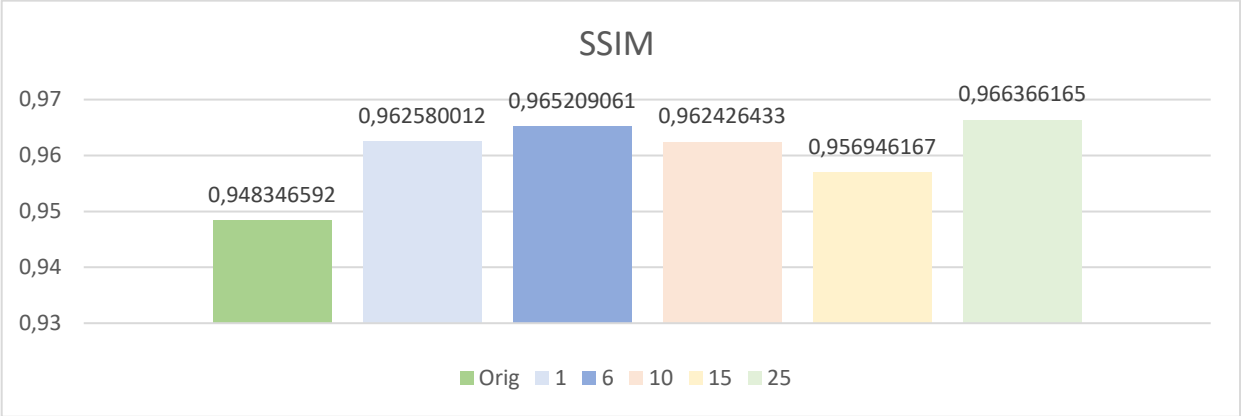

**57 pav. MSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



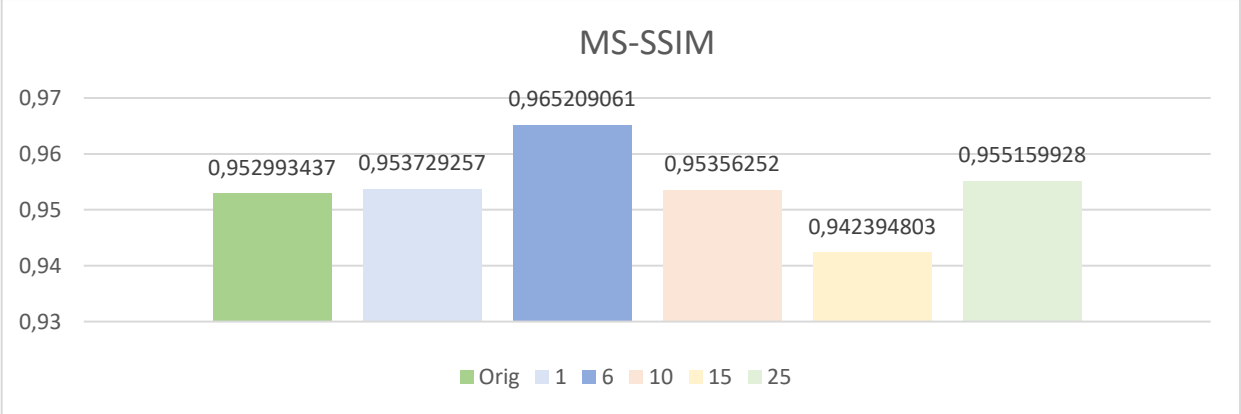

**58 pav. PSNR metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**59 pav. RMSE metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**60 pav. SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**61 pav. MS-SSIM metrikos vidutinis rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

Paskutinėje analizėje išliko panaši tendencija pagal keletą metrikų (*PSNR*, *SSIM*, *MS-SSIM*), kad didesnis grupės skaičius nebūtinai reiškia geresnius rezultatus. Verta atkreipti dėmesį kad padidėjus grupės skaičiui, praradimų funkcija taip pat rodo vis blogesnį rezultatą didėjant grupei, nors grafikas greitai nusileidžia (pav. 56) ir lieka tolygus kaip ir kiti grafikai viso mokymo metu. Ir nors padidinant grupės skaičius iš tiesų padidėja neuroninio tinklo tikslumas, verta atkreipti dėmesį kad kartu su šiuo nustatymu stipriai išauga ir mokymo laikas. Kitu atveju, nepadidinus mokymo laiko neuroninis tinklas daro dideles spalvinimo klaidas. Kaip analizė

parodė, kad gauti gan kokybišką rezultatą, užtenka ir prieš tai išbandytų ~20 000 iteracijų. Koks skirtumas yra tarp geriausio ir blogiausio šios analizės dalies rezultatų galima pamatyti 62 pav. ir 63 pav.

**62 pav. Geriausias rezultatas – MSE = 48,08002, grupės dydis 15, 20000 iteracijų.**

**63 pav. Blogiausias rezultatas – MSE = 1744,2633, grupės dydis 10, 100 iteracijų.**

## 8.5. Naujų klasifikatorių bandymas

Originaliame neuroniniame tinkle naudojamas klasifikatoriaus tinklas *VGG16* – šiame tinkle jis padeda klasifikuoti objektus, esančius vaizduose. Tačiau galima panaudoti ir kitokius klasifikatorius, tokiu atveju iš dalies pakeičiant ir pačio tinklo struktūrą. Didelę dalį naujų klasifikatorių galima rasti atviru kodu, kartu su iš anksto mokytais modeliais. Du nauji klasifikatoriai, su kuriai buvo atlikta analizė, yra *VGG19* ir *ResNet50*.

*VGG19* klasifikatorius yra naujesnis *VGG16* klasifikatorius, pagrindinis skirtumas kad naują tinklą sudaro 19 vietoje 16 sluoksnių. Panaudotas analizei naujas *VGG19* tinklas yra atviro kodo ir pasiekiamas internete[14] kartu su apmokytu modeliu, ir jo įdiegimas į tinklą vyko keliose neuroninio tinklo vietose - hiperstulpo įgyvendinimo dalyje (failas *residual_encoder.py*) ir

[14] https://github.com/machrisaa/tensorflow-vgg/blob/master/vgg19.py

mokymo, spalvinimo nustatymų dalyje (failas *common.py*). Reikalingi pakeitimai nurodyti 64-65 pav.

```
bn_4 = self.batch_normal(vgg.conv4_4, "bn_4", is_training)
b_conv4 = self.conv_layer(bn_4, "b_conv4", is_training, bn=False)

if debug:
    assert bn_4.get_shape().as_list()[1:] == [28, 28, 512]
    assert b_conv4.get_shape().as_list()[1:] == [28, 28, 256]

b_conv4_upscale = tf.image.resize_images(b_conv4, [56, 56], method=image_resize_method)
bn_3 = self.batch_normal(vgg.conv3_4, "bn_3", is_training)
b_conv3_input = tf.add(bn_3, b_conv4_upscale, name="b_conv3_input")
b_conv3 = self.conv_layer(b_conv3_input, "b_conv3", is_training)
```

**64 pav. Dėl naujų atsiradusių sluoksnių, hiperstulpo naudojami konvoliuciniai sluoksniai keičiami į naujai atsiradusius.**

```
vgg = vgg19.Vgg19()
```

**65 pav. Vietoje *VGG16* kode užkraunamas *VGG19* klasifikatorius su modeliu.**

Klasifikatorius *ResNet50* yra gan naujas objektų klasifikavimo dirbtinis neuroninis tinklas, pirmą kartą pristatytas 2015 metais, mokslininkų *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun*, moksliniame darbe „*Deep Residual Learning for Image Recognitio*n“ [HXR+15]. Klasifikatorius yra 50 sluoksnių gylio, ir kaip ir kiti tinklai, jo įvairius sprendimus galima rasti atviro kodo[15] su prieš tai apmokytu modeliu. Šio tinklo įdiegimas sudėtingesnis, nes reikalingas keisti ir pats klasifikatoriaus kodas ir hiperstulpo kodas, tam kad veiktų bendra sistema. Apie reikalingus padaryti pakeitimus galima pamatyti 66–68 pav.

```
def build_model(self, image):

    self.rgb = tf.placeholder("float32", [None, None, None, 3])
    bgr = self.normal_img(image)
    self.conv1 = self.conv_block_1(bgr)
    self.conv2 = self._build_box(self.conv1, 3, '2')
    self.conv3 = self._build_box(self.conv2, 4, '3')
    self.conv4 = self._build_box(self.conv3, 6, '4')
    self.conv5 = self._build_box(self.conv4, 3, '5')
    self.pool5 = tf.nn.avg_pool(self.conv5, [1,7,7,1], [1,1,1,1], 'SAME', name='pool5')
    self.pool5_resize = tf.reshape(self.pool5, [-1,2048])
    self.shortcut = self.pool5
    self.fc1000 = self.fc_1000(self.pool5_resize)

    self.prob = tf.nn.softmax(self.fc1000, name = 'softmax_prob')
```

**66 pav. *ResNet50* kode nereikalinga atskira sesija tinklui, taip pat reikalingas naujas parametras – paveikslėlis, kuris turi pereiti per konvoliucijas tinkle.**

[15] https://github.com/BigWZhu/ResNet50/blob/master/ResNet.py

```
weights = {
    'b_conv4': tf.Variable(tf.truncated_normal([1, 1, 2048, 512], stddev=0.01), trainable=True),
    'b_conv3': tf.Variable(tf.truncated_normal([3, 3, 512, 256], stddev=0.01), trainable=True),
    'b_conv2': tf.Variable(tf.truncated_normal([3, 3, 256, 64], stddev=0.01), trainable=True),
    'b_conv1': tf.Variable(tf.truncated_normal([3, 3, 64, 3], stddev=0.01), trainable=True),
    'b_conv0': tf.Variable(tf.truncated_normal([3, 3, 3, 3], stddev=0.01), trainable=True),
    'output_conv': tf.Variable(tf.truncated_normal([3, 3, 3, 2], stddev=0.01), trainable=True),
}
```

**67 pav. Pagrindiniame hiperstulpe reikalingos pakeisti sluoksnių dimensijos, kitu atveju sluoksnių dydžiai tarp klasifikatoriaus ir hiperstulpo nesutaps ir tinklas negalės veikti**

```
bn_4 = self.batch_normal(resnet1.conv5, "bn_4", is_training)
b_conv4 = self.conv_layer(bn_4, "b_conv4", is_training, bn=False)

if debug:
    assert bn_4.get_shape().as_list()[1:] == [28, 28, 512]
    assert b_conv4.get_shape().as_list()[1:] == [28, 28, 256]

b_conv4_upscale = tf.image.resize_images(b_conv4, [28, 28], method=image_resize_method)
print(b_conv4_upscale)
bn_3 = self.batch_normal(resnet1.conv3, "bn_3", is_training)
b_conv3_input = tf.add(bn_3, b_conv4_upscale, name="b_conv3_input")
b_conv3 = self.conv_layer(b_conv3_input, "b_conv3", is_training)

if debug:
    assert b_conv4_upscale.get_shape().as_list()[1:] == [56, 56, 256]
    assert bn_3.get_shape().as_list()[1:] == [56, 56, 256]
    assert b_conv3_input.get_shape().as_list()[1:] == [56, 56, 256]
    assert b_conv3.get_shape().as_list()[1:] == [56, 56, 128]

b_conv3_upscale = tf.image.resize_images(b_conv3, [56, 56], method=image_resize_method)
bn_2 = self.batch_normal(resnet1.conv2, "bn_2", is_training)
b_conv2_input = tf.add(bn_2, b_conv3_upscale, name="b_conv2_input")
b_conv2 = self.conv_layer(b_conv2_input, "b_conv2", is_training)

if debug:
    assert b_conv3_upscale.get_shape().as_list()[1:] == [112, 112, 128]
    assert bn_2.get_shape().as_list()[1:] == [112, 112, 128]
    assert b_conv2_input.get_shape().as_list()[1:] == [112, 112, 128]
    assert b_conv2.get_shape().as_list()[1:] == [112, 112, 64]

b_conv2_upscale = tf.image.resize_images(b_conv2, [56, 56], method=image_resize_method)
bn_1 = self.batch_normal(resnet1.conv1, "bn_1", is_training)
b_conv1_input = tf.add(bn_1, b_conv2_upscale, name="b_conv1_input")
b_conv1 = self.conv_layer(b_conv1_input, "b_conv1", is_training)

if debug:
    assert b_conv2_upscale.get_shape().as_list()[1:] == [224, 224, 64]
    assert bn_1.get_shape().as_list()[1:] == [224, 224, 64]
    assert b_conv1_input.get_shape().as_list()[1:] == [224, 224, 64]
    assert b_conv1.get_shape().as_list()[1:] == [224, 224, 3]

b_conv1_upscale = tf.image.resize_images(b_conv1, [224, 224], method=image_resize_method)
bn_0 = self.batch_normal(input_data, "bn_0", is_training)
b_conv0_input = tf.add(bn_0, b_conv1_upscale, name="b_conv0_input")
b_conv0 = self.conv_layer(b_conv0_input, "b_conv0", is_training)

if debug:
    assert bn_0.get_shape().as_list()[1:] == [224, 224, 3]
    assert b_conv0_input.get_shape().as_list()[1:] == [224, 224, 3]
    assert b_conv0.get_shape().as_list()[1:] == [224, 224, 3]
```

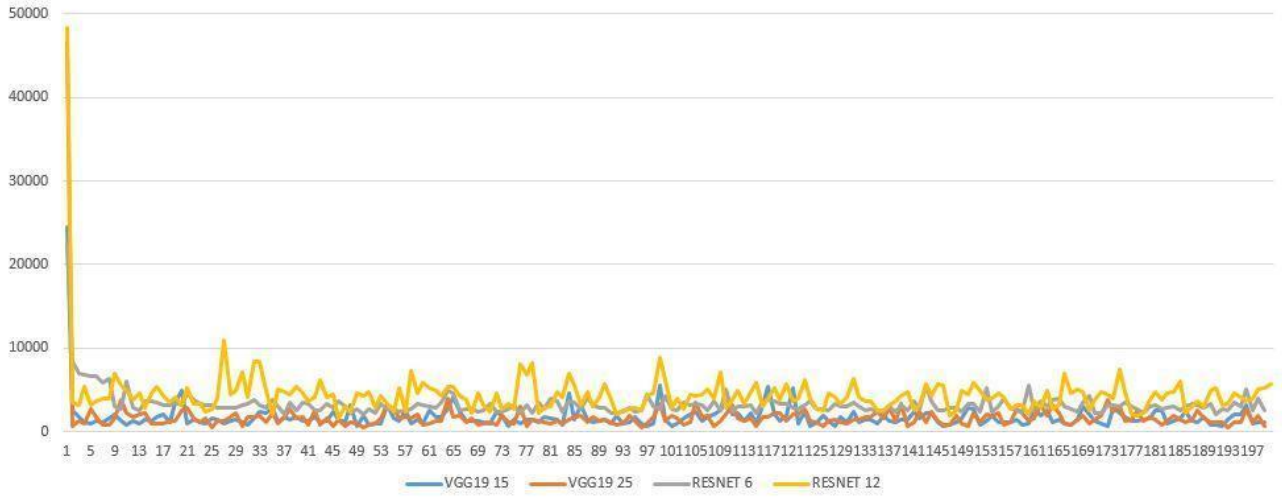
**68 pav. Pagrindiniame hiperstulpe reikalingi pridėti nauji konvoliuciniai sluoksniai iš *ResNet50*, taip pat papildomas išdidinimas b_conv1_upscale tam, kad išvestis liktų tokia pati kaip ir prieš tai.**

Paruošus dirbtinius neuroninius tinklus, galima tęsti toliau tinklų mokymą tam tikras nustatytais parametrais ir tikrinti rezultatus.

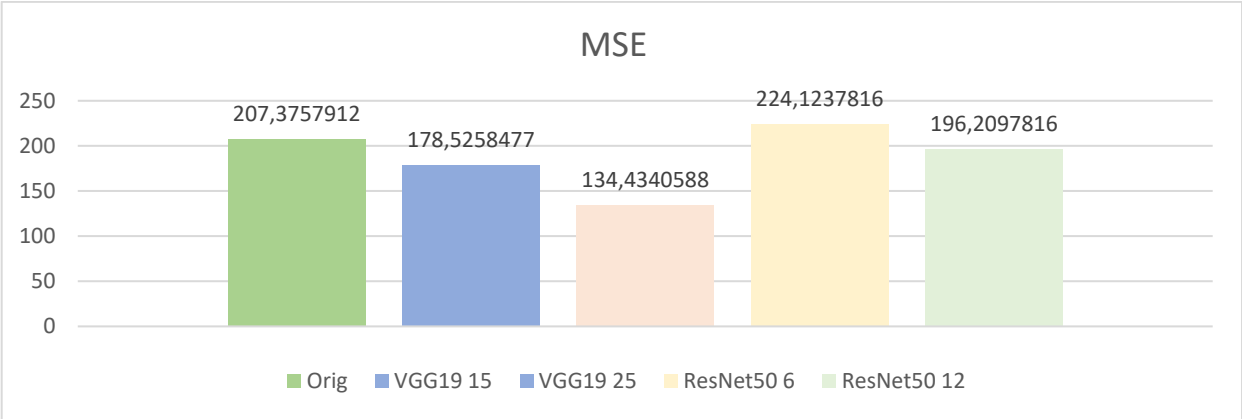
- Grupės dydis – (6, 12 su *ResNet*, 15, 25 su *VGG19*) – dydis apribojamas iki 12 dėl *ResNet50* ir turimos techninės įrangos atminties ir skaičiavimo limitų.
- Validacijos dažnis – 2000 iteracijų.
- Maišymo dažnis – 2000 iteracijų.
- Iteracijų skaičius – 20 000.
- Neuroninio tinklo architektūra – klasifikatorius pakeistas *VGG16* tinklas į *VGG19* arba *ResNet50*.

9 lentelė – Pakeistos architektūros mokymo laikai

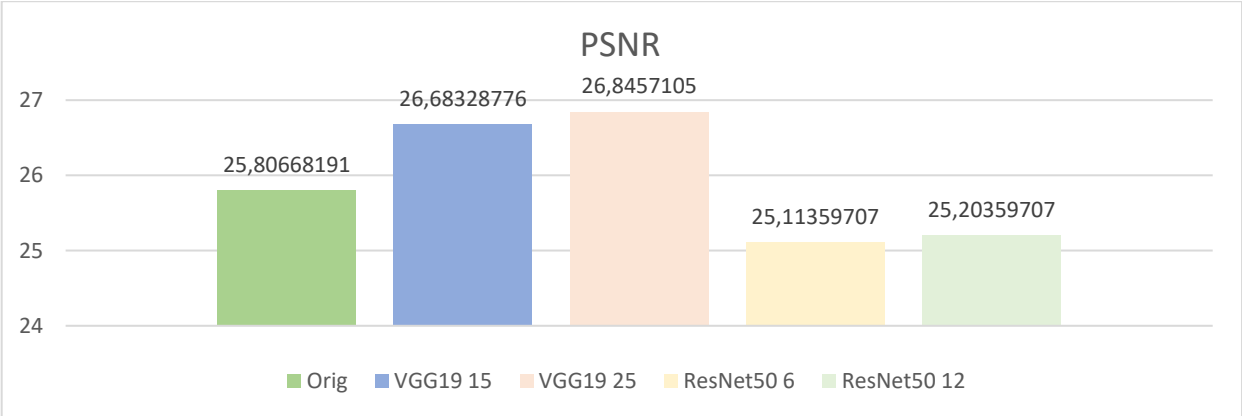
| | |
|---|---|
| Grupės dydis 15 *VGG19* | 4val. 6 min. 5s |
| Grupės dydis 25 *VGG19* | 7val. 57 min. 3s |
| Grupės dydis 6 *ResNet50* | 2val. 11 min. 12s |
| Grupės dydis 12 *ResNet50* | 5val. 28 min. 3s |



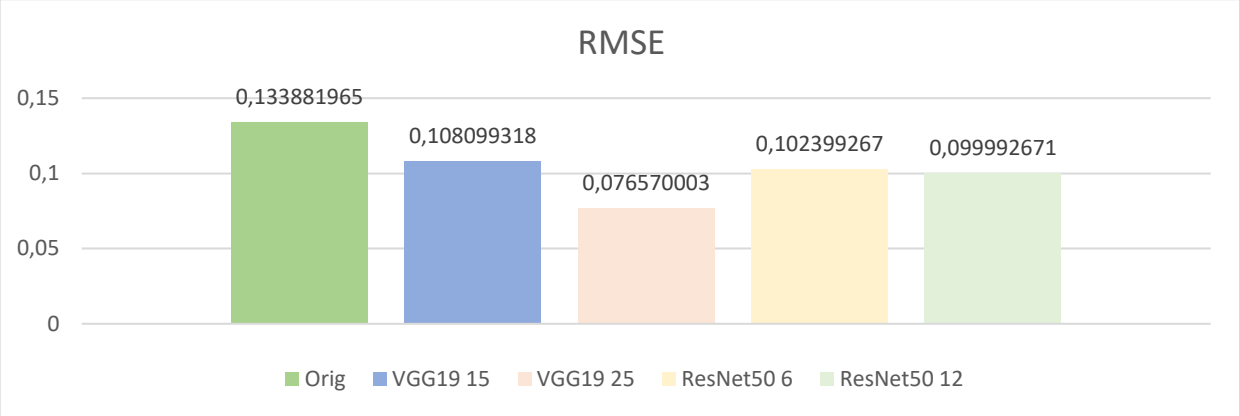

**69 pav. Skirtingų architektūrų įtaka praradimų funkcijos priklausomybei nuo mokymo laiko (y ašis – praradimų funkcijos rezultatas, x ašis – iteracijos žingsnis, 1 žingsnis – 100 iteracijų).**



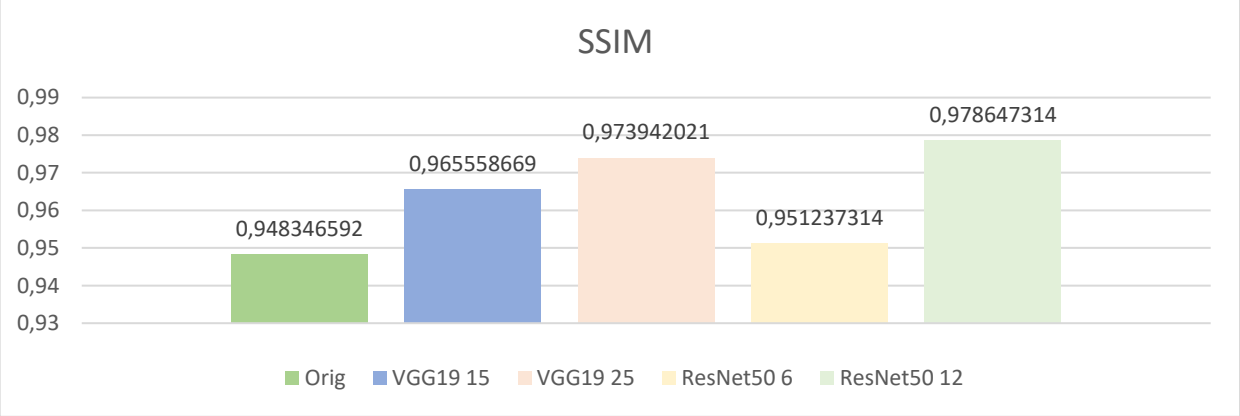

**70 pav. Įvairių pakeistų klasifikatorių vidutinis MSE rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



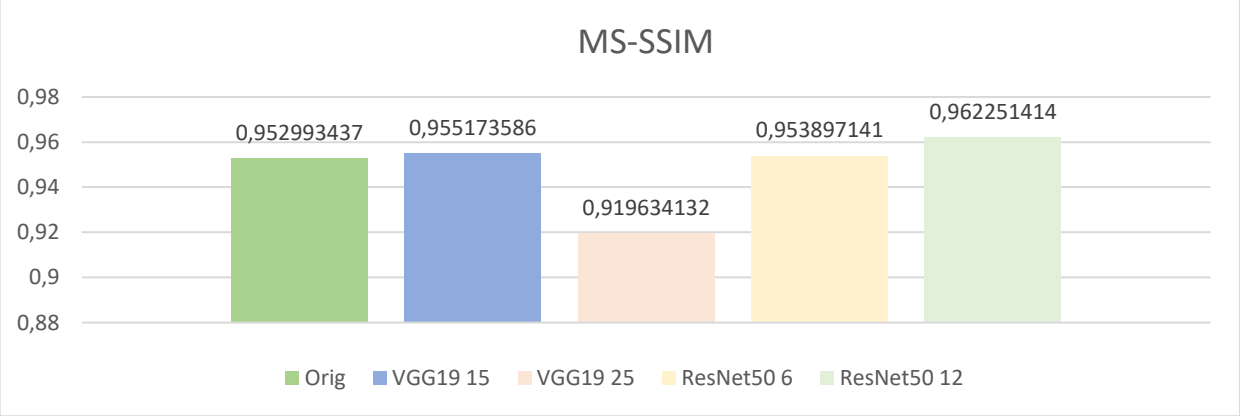

**71 pav. Įvairių pakeistų klasifikatorių vidutinis PSNR rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**72 pav. Įvairių pakeistų klasifikatorių vidutinis RMSE rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**



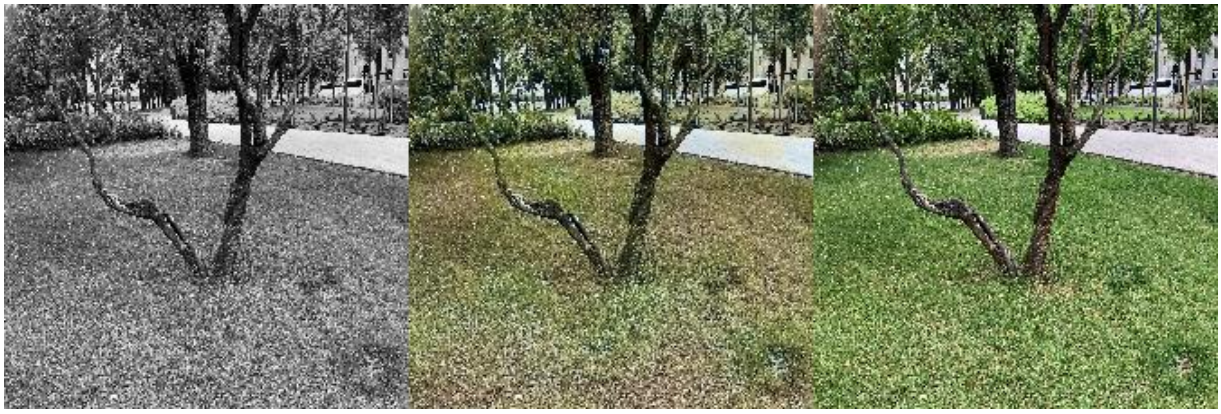

**73 pav. Įvairių pakeistų klasifikatorių vidutinis SSIM rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**




**74 pav. Įvairių pakeistų klasifikatorių vidutinis MS-SSIM rezultatas, palygintas su originalaus neuroninio tinklo rezultatu.**

Naujų klasifikatorių su architektūra analizės rezultatas – pavyko gauti iš dalies geresnius rezultatus beveik visuose atvejuose, lyginant su pradiniais originalaus tinklo duomenimis. Abu nauji klasifikatoriai pasirodė panašiai, o lyginant metrikomis – pagal *MSE, PSNR* ir *RMSE* geresni rezultatai buvo *VGG19* tinklo, o pagal *SSIM* ir *MS-SSIM* – geresni buvo *ResNet50* pakeisto tinklo. Dar vienas stiprus skirtumas tarp abiejų klasifikatorių – mokymo laikas. *ResNet50*, net ir mažesniais grupių kiekiais mokymo laikas yra gerokai didesnis negu *VGG19*.

Todėl šioje vietoje kitas klasifikatorius laiko pranašumą. Šios analizės geriausią ir blogiausią rezultatą galima pamatyti 75–76 pav.



**75 pav. Blogiausias rezultatas – MSE = 301,0325, ResNet50 12 grupės dydžio bandymas.**

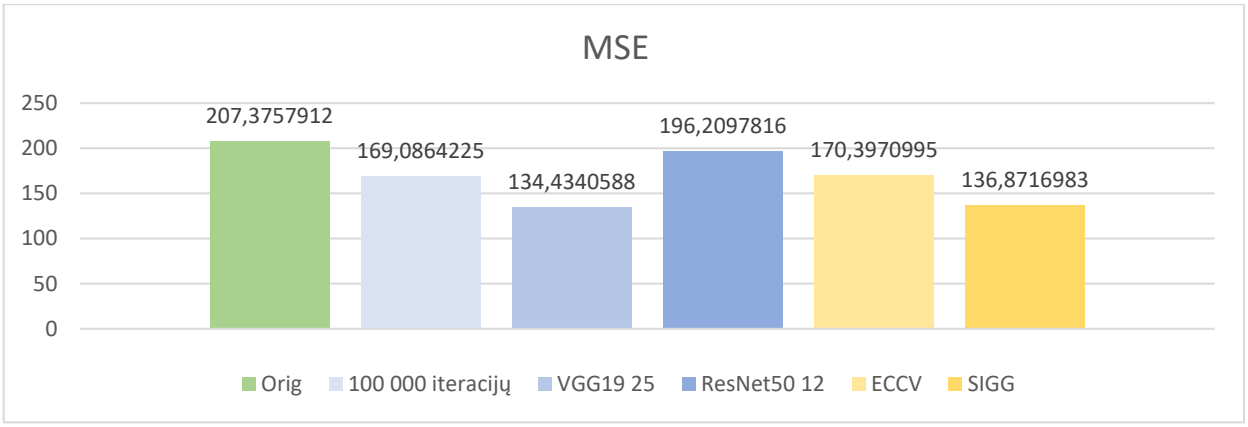
**76 pav. Geriausias rezultatas – MSE = 110,2598, ResNet50 12 grupės dydžio bandymas.**

## 8.6. Palyginimas su kitais tinklais

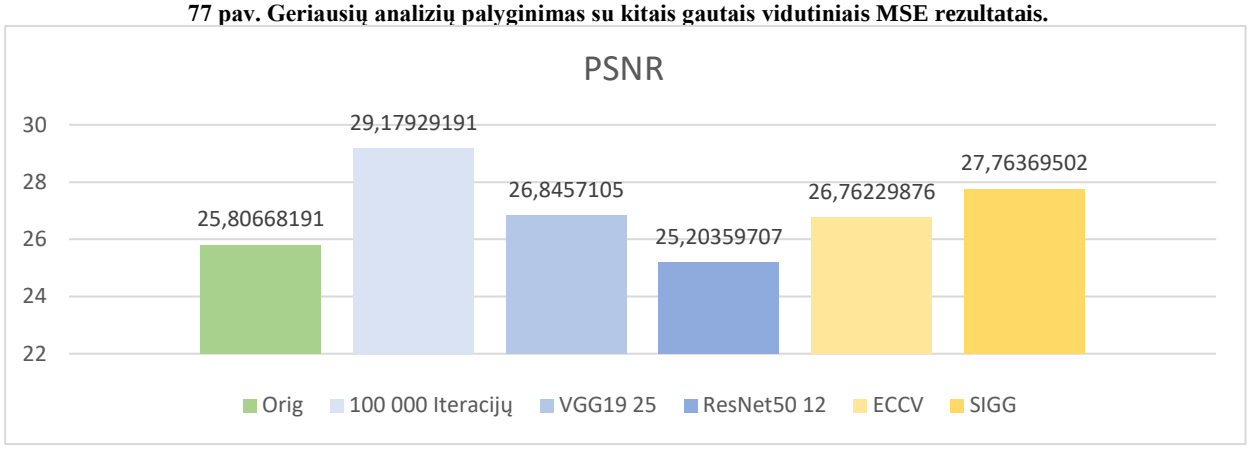
Atlikus analizę su įvairiais parametrais, taip pat išbandžius kitokias tinklo architektūras, paskutinis žingsnis yra įvertinti rezultatus, palyginant juos su kitais, mokslininkų kurtais ir išbandytais tinklais. Tokiu atveju bus gautas aiškus vaizdas, kokį bendrą rezultatą pavyko gauti. Palyginti rezultatus bus naudojamas dirbtinis neuroninis tinklas „*Colorful Image Colorization*“, sukurto pagal mokslininkų *Richard Zhang, Phillip Isola, Alexei A. Efros* mokslinį darbą „*Colorful Image Colorization*“ [RPA16]. Pats neuroninis tinklas kartu su jau apmokytais modeliais yra atviro kodo[16] ir dėl to tinka palyginimui. Be to, neuroninis tinklas vienu metu naudoja du skirtingus spalvinimo modelius – *SIGGRAPH17* ir *ECCV16,* dėl to atsiranda galimybė palyginti su dvejais skirtingais modeliais. Taisyklės palyginimui – spalvinimas turi būti vykdomas tiems patiems 10 analizės paveikslėlių, ir palyginimas vyksta tomis pačiomis

[16] https://github.com/richzhang/colorization

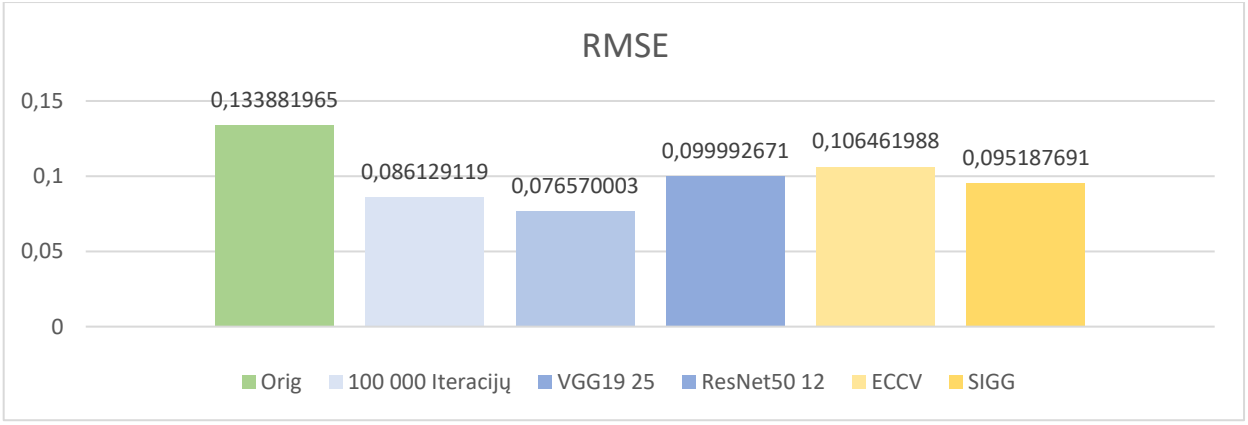
metrikomis. Palyginimų bandymams bus naudojami rezultatai iš geriausius rezultatus parodžiusių bandymų.



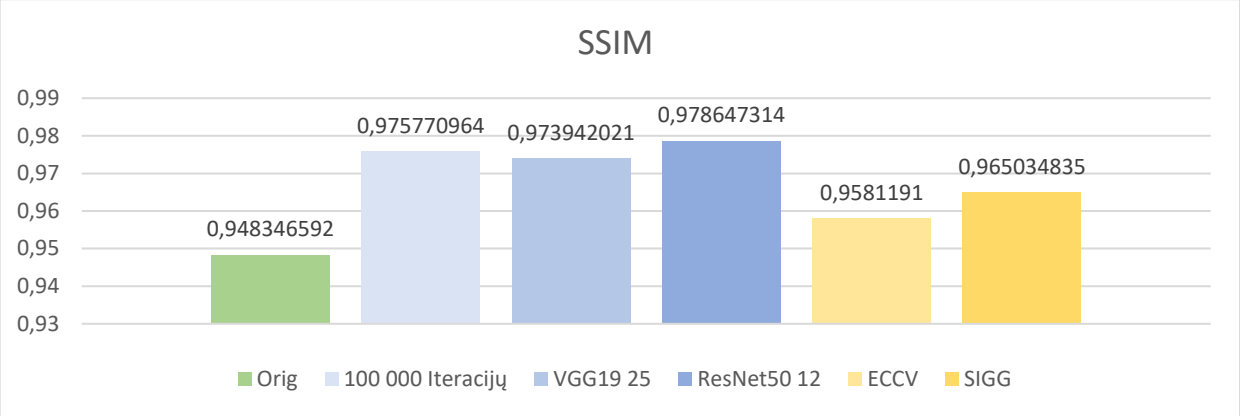

**77 pav. Geriausių analizių palyginimas su kitais gautais vidutiniais MSE rezultatais.**



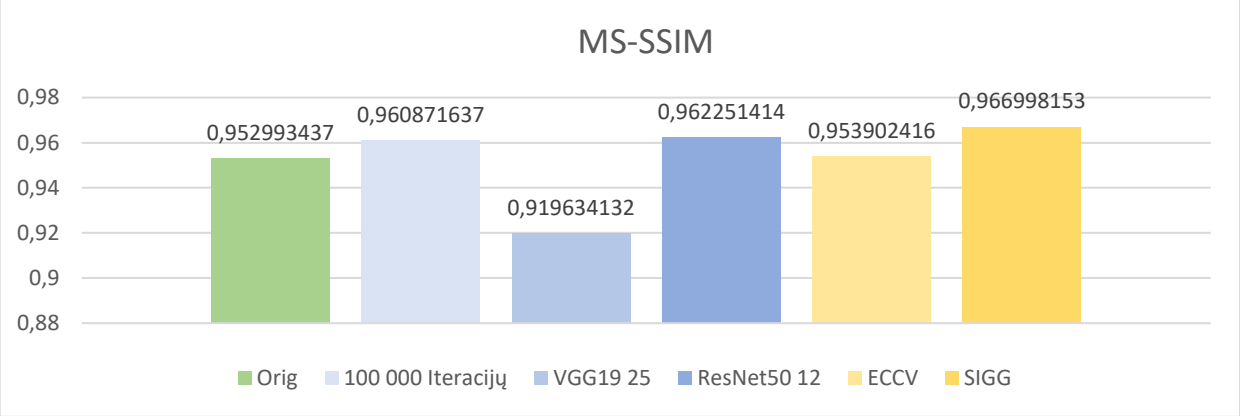

**78 pav. Geriausių analizių palyginimas su kitais gautais vidutiniais PSNR rezultatais.**




**79 pav. Geriausių analizių palyginimas su kitais gautais vidutiniais RMSE rezultatais.**




**80 pav. Geriausių analizių palyginimas su kitais gautais vidutiniais SSIM rezultatais.**




**81 pav. Geriausių analizių palyginimas su kitais gautais vidutiniais MS-SSIM rezultatais.**

Rezultatą iš dalies pavyko pasiekti – keičiant dirbtinio neuroninio tinklo parametrus ir keičiant architektūrą galima pagerinti tinklo spalvinimo rezultatus – pradinis tikslas pagerinti originalų gautą rezultatą pavyko visomis galutinėmis metrikomis – matematiškai nuspalvinti paveikslėliai yra arčiau originalaus paveikslėlio spalvų. Taip pat galutiniai rezultatai gali lygintis ir su kitais neuroniniais tinklais – pavyzdžiui *VGG19* klasifikatoriaus tinklui pavyko nedidelėmis paklaidomis pasiekti vidutiniškai truputį geresnį rezultatą negu *ECCV* ir *SIGG* kurtais spalvinimo neuroniniais tinklais. Geriausius šios analizės rezultatus galima pamatyti 82–83 pav.

**82 pav. Blogiausias rezultatas – MSE = 315,5543, 100 000 iteracijų bandymas.**

**83 pav. Geriausias rezultatas – MSE = 56,0280, SIGG bandymas.**

# 9. Rezultatai ir išvados

Buvo atliktas išsami dirbtinio neuroninio tinklo analizė ir buvo įvertintos jo paveikslėlių spalvinimo kokybės galimybės. Buvo atliktas gautų analizės rezultatų vertinimas keliais skirtingais kriterijais ir jų tarpusavyje palyginimas. Buvo palyginti gauti geriausi rezultatai su kitų dirbtinių neuroninių tinklų rezultatais, gautais tomis pačiomis sąlygomis ir jie buvo tarpusavyje įvertinti. Darbo metu pavyko įsitikinti ir parodyti, kad parametrų keitimas, taip pat architektūros keitimas keičiant skirtingus modulius gali turėti reikšmės norint gauti geresnius rezultatus arba siekiant pagerinti kitus. Siekti tikslai ir uždaviniai yra pasiekti ir rezultatai įvertinti – gerinant dirbtinius neuroninis tinklus svarbu ne tik jų struktūra, bet ir kokiais parametrais vyksta pats procesas. Taigi, įvertinus rezultatus, galima padaryti šias išvadas:

1. Buvo nustatyta riba, ties kuria verta sustabdyti tinklo mokymą. Atlikti tyrimai parodė, kad apie 20 000 yra iteracijų skaičiaus riba, kada paveikslėlių kokybė didėja nežymiai. Tokią tendenciją apie iteracijas parodė apskaičiuotos paveikslėlių kokybės *MSE, PSNR, RMSE* ir *SSIM* metrikos.
2. Atlikus tyrimą, nustatyta, kad padidinus grupės dydį, kartu panaudojus jau prieš tai nustatytą mokymo iteracijų skaičių, galima gauti vidutiniškai geresnį paveikslėlių spalvinimo kokybės rezultatą, tuo pačiu metu ir sutaupant mokymo laiko, nes išaugant grupės dydžiui stipriai išauga mokymo laikas.
3. Į esamus neuroninius tinklus galima integruoti kitus, jau sukurtus sprendinius ir taip pagerinti galutinius gaunamus rezultatus. Tokį rezultatą parodė pakeitus senesnį *VGG* tinklą naujesniu ir rezultatai buvo atspindėti metrikose *MSE* (nuo 207,375 iki 134,434), *PSNR* (nuo 25,806 iki 26,845), *RMSE* (nuo 0,133 iki 0,076) ir *SSIM* (nuo 0,948 iki 0,973).
4. Atlikus analizę, parodyta, kad pakeitus originalų neuroninį tinklą ir panaudojus naujus iteracijų ir grupės dydžio parametrus, jis savo spalvinimo kokybe nenusileidžia kitam neuroniniam tinklui „*Colorful image colorization*“ su dvejais skirtingais naudojamais modeliais, kaip tą parodė metrikos *MSE* (nuo 134,434 *VGG19* iki 136,871 *SIGGRAPH17*), *PSNR* (nuo 25,203 *ResNet50* iki 26,766 *ECCV16*), *RMSE* (nuo 0,0765 *VGG19* iki 0,095 *SIGGRAPH17*) ir *SSIM* (nuo 0.978 *ResNet50* iki 0,965 *SIGGRAPH17* ).

Atliktas darbas ir analizė galėtų pasitarnauti mokslininkams, kurie dirba dirbtinių neuroninių tinklų srityje kurdami metodus, kaip pasinaudojant dirbtiniais neuroniniais tinklais gauti kuo geresnį rezultatą ir kaip gautus rezultatus vertinti.

# 10. Šaltinių sąrašas


[SZ15] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. [žiūrėta 2021-05-09]. Prieiga per Internetą: <https://arxiv.org/pdf/1409.1556.pdf>

[HAG15] Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 447-456). [žiūrėta 2021-05-08]. Prieiga per Internetą: < https://arxiv.org/pdf/1411.5752.pdf>

[WSB03] Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003* (Vol. 2, pp. 1398-1402). Ieee. [žiūrėta 2021-04-12]. Prieiga per Internetą: < https://www.cns.nyu.edu/pub/eero/wang03b.pdf>

[CQ19] Lei, C., & Chen, Q. (2019). Fully automatic video colorization with self-regularization and diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3753-3761). [žiūrėta 2020-11-17]. Prieiga per Internetą:< https://arxiv.org/pdf/1908.01311.pdf >

[HXR+15] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). [žiūrėta 2021-05-16]. Prieiga per Internetą: <https://arxiv.org/pdf/1512.03385.pdf>

[ADY] Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. In *ACM SIGGRAPH 2004 Papers* (pp. 689-694). [žiūrėta 2020-09-01]. Prieiga per Internetą: <http://homepages.inf.ed.ac.uk/ksubr/Files/Papers/p689-levin.pdf>

[HKM19] Thasarathan, H., Nazeri, K., & Ebrahimi, M. (2019, May). Automatic temporally coherent video colorization. In *2019 16th Conference on Computer and Robot Vision (CRV)* (pp. 189-194). IEEE. [žiūrėta 2020-09-01]. Prieiga per Internetą: < https://arxiv.org/pdf/1904.09527.pdf >

[VK16] Bagaria, V. K., & Tatwawadi, K. CS231N Project: (2016) Coloring black and white world using Deep Neural Nets. [žiūrėta 2020-09-01]. Prieiga per Internetą: <http://cs231n.stanford.edu/reports/2016/pdfs/205_Report.pdf>

[RAD+12] Gupta, R. K., Chia, A. Y. S., Rajan, D., Ng, E. S., & Zhiyong, H. (2012, October). Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia* (pp. 369-378). [žiūrėta 2020-09-01]. Prieiga per Internetą:

<https://people.cs.clemson.edu/~jzwang/1501863/mm2012/p369-gupta.pdf >

[BFZ+14] Li, B., Zhao, F., Su, Z., Liang, X., Lai, Y. K., & Rosin, P. L. (2017). Example-based image colorization using locality consistent sparse representation. IEEE Transactions on Image Processing, 26(11), 5188-5202. [žiūrėta 2020-09-01]. Prieiga per Internetą: <http://orca.cf.ac.uk/102840/1/TIP_Sparse_Colorization.pdf>

[RJP+17] Zhang, R., Zhu, J. Y., Isola, P., Geng, X., Lin, A. S., Yu, T., & Efros, A. A. (2017). Real-time user-guided image colorization with learned deep priors. arXiv preprint arXiv:1705.02999.[žiūrėta 2020-09-01]. Prieiga per Internetą: < https://arxiv.org/pdf/1705.02999.pdf >

[MLA+20] Ho, M. M., Zhang, L., & Zhou, J. (2020). Semantic-driven Colorization. arXiv preprint arXiv:2006.07587.[žiūrėta 2020-09-01]. Prieiga per Internetą: < https://arxiv.org/pdf/2006.07587.pdf >

[SRY+11] Chia, A. Y. S., Zhuo, S., Gupta, R. K., Tai, Y. W., Cho, S. Y., Tan, P., & Lin, S. (2011). Semantic colorization with internet images. ACM Transactions on Graphics (TOG), 30(6), 1-8. [žiūrėta 2020-09-01]. Prieiga per Internetą: <https://www.researchgate.net/profile/Raj_Gupta18/publication/314769103_Semantic_colorization_with_internet_images/links/592e454caca272fc55b5b782/Semantic-colorization-with-internet-images.pdf >

[JLC+18] Zhao, J., Liu, L., Snoek, C. G., Han, J., & Shao, L. (2018). Pixel-level semantics guided image colorization. arXiv preprint arXiv:1808.01597. [žiūrėta 2020-09-01]. Prieiga per Internetą:< https://arxiv.org/pdf/1808.01597.pdf >

[KEM18] Nazeri, K., Ng, E., & Ebrahimi, M. (2018). Image colorization using generative adversarial networks. In *International conference on articulated motion and deformable objects* (pp. 85-94). Springer, Cham. [žiūrėta 2020-09-01]. Prieiga per Internetą: <https://arxiv.org/pdf/1803.05400.pdf >

[DT17] Varga, D., & Szirányi, T. (2017). Twin deep convolutional neural network for example-based image colorization. In *International Conference on Computer Analysis of Images and Patterns* (pp. 184-195). Springer, Cham. [žiūrėta 2021-02-24]. Prieiga per Internetą: <https://eprints.sztaki.hu/9190/1/Varga_184_3270605_ny.pdf>

[TMK02] T. Welsh, M. Ashikhmin, K. Mueller (2002) „Transferring Color to Greyscale Images“. Center for Visual Computing, Computer Science Department, SUNY at Stony Brook. [žiūrėta 2018-12-13]. Prieiga per Internetą:

<https://www.researchgate.net/publication/220183710_Transferring_Color_to_Greyscale_Images>

[JYL+17] Liao, J., Yao, Y., Yuan, L., Hua, G., & Kang, S. B. (2017). Visual attribute transfer through deep image analogy. arXiv preprint arXiv:1705.01088. [žiūrėta 2018-12-13]. Prieiga per Internetą: <https://arxiv.org/pdf/1705.01088.pdf>

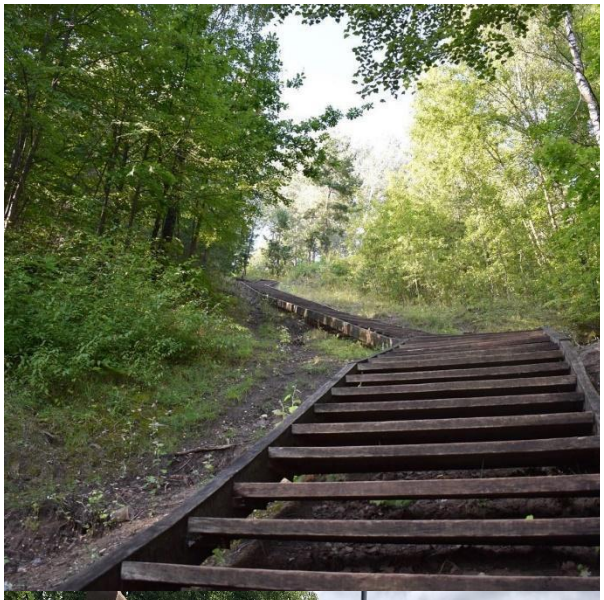
[RPA16] Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision* (pp. 649-666). Springer, Cham. [žiūrėta 2019-01-12]. Prieiga per Internetą: <https://arxiv.org/pdf/1603.08511.pdf>

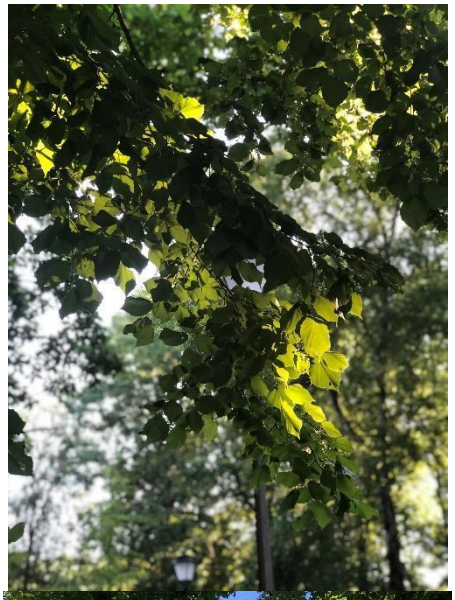
[RP18] Rachmadi, R. F., & Purnama, I. (2015). Vehicle color recognition using convolutional neural network. *arXiv preprint arXiv:1510.07391*. [žiūrėta 2021-05-16]. Prieiga per Internetą: <https://arxiv.org/pdf/1510.07391.pdf>

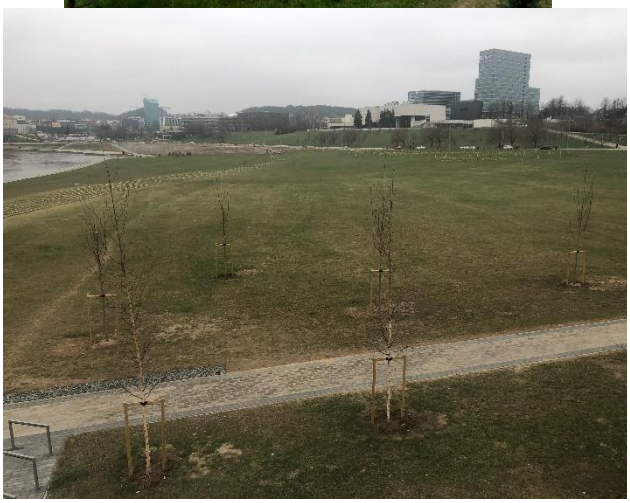
# 11. Priedai

## 11.1. Originalios analizės nuotraukos

Visos analizei naudotos nuotraukos buvo nufotografuotos ir sukurtos magistro baigiamojo darbo autoriaus.







## 11.2. Naudota programinė įranga ir *Python* paketai

1. Programinė įranga:
   1.1. Anaconda Navigator 1.10.0
   1.2. Spyder 4.2.5
2. Python paketai:
   2.1. Sewar 0.4.4
   2.2. Scipy 1.5.2
   2.3. Scikit-image 0.17.2
   2.4. Tensorflow 1.8.0
   2.5. Tensorflow-gpu 1.8.0
   2.6. Numpy 1.19.5
   2.7. Opencv 4.5.1
   2.8. Cudnn 7.6.5
   2.9. cudatoolkit 9.0
   2.10. matplotlib 3.3.4

## 11.3. Naudota kompiuterinė įranga

- Operacinė sistema – Windows 10
- Kompiuteris MSI GE72 7RD Apache
- Procesorius Intel Core i7-7700HQ 2.80GHz
- RAM atmintis – 16Gb
- GPU procesoriai Intel HD Graphics 630 8Gb, NVIDIA GeForce GTX 1050 12Gb